

STDArm: Transferring Visuomotor Policies From Static Data Training to Dynamic Robot Manipulation

Yifan Duan^{*1}, Heng Li^{*1}, Yilong Wu^{*1}, Wenhao Yu², Xinran Zhang¹, Yedong Shen¹,
Jianmin Ji¹ and Yanyong Zhang^{†3}

¹School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

²Institute of Advanced Technology, University of Science and Technology of China, Hefei, China

³School of Artificial Intelligence and Data Science, University of Science and Technology of China, Hefei, China

^{*}Equal Contribution [†]Corresponding Author

Project page: <https://yjsx.top/stdarm>

Abstract—Recent advances in mobile robotic platforms like quadruped robots and drones have spurred a demand for deploying visuomotor policies in increasingly dynamic environments. However, the collection of high-quality training data, the impact of platform motion and processing delays, and limited onboard computing resources pose significant barriers to existing solutions. In this work, we present *STDArm*, a system that directly transfers policies trained under static conditions to dynamic platforms without extensive modifications.

The core of *STDArm* is a real-time action correction framework consisting of: (1) an action manager to boost control frequency and maintain temporal consistency, (2) a stabilizer with a lightweight prediction network to compensate for motion disturbances, and (3) an online latency estimation module for calibrating system parameters. In this way, *STDArm* achieves centimeter-level precision in mobile manipulation tasks.

We conduct comprehensive evaluations of the proposed *STDArm* on two types of robotic arms, four types of mobile platforms, and three tasks. Experimental results indicate that the *STDArm* enables real-time compensation for platform motion disturbances while preserving the original policy’s manipulation capabilities, achieving centimeter-level operational precision during robot motion.

I. INTRODUCTION

Visuomotor policies, which directly map visual observations to robot actions, have emerged as an effective paradigm for complex task acquisition [1]. By integrating end-to-end imitation learning frameworks trained on human demonstrations, visuomotor policies eliminate the need for task-specific modular pipeline design, translating raw sensor input into robot actions without intermediate representations [2]. Recent advances have enabled robotic systems with remarkable capabilities such as battery insertion [3], food preparation [4], and textile handling [5], etc.

The acquisition of high-quality human demonstrations, typically achieved through teleoperation, plays a pivotal role in the successful skill acquisition using imitation learning-based visuomotor policies [6]. This process is relatively simple with stationary robotic systems, leading to common experimental

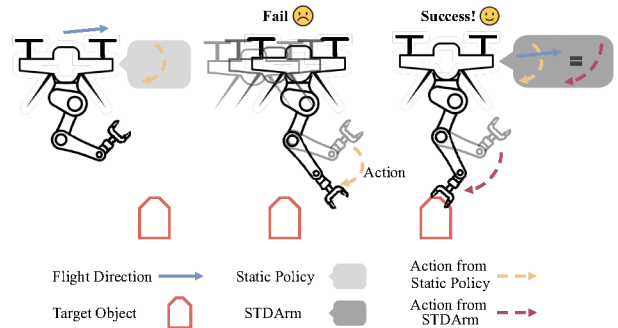


Fig. 1. When deploying a policy trained on static data on computationally constrained edge devices, action drift may occur due to delays like inference and execution latency. Our system, *STDArm*, addresses this by direct rectification of actions, enabling reliable task completion without policy modifications.

setups that use table-mounted manipulators. However, significant challenges arise when working with dynamic robot platforms, such as drones, humanoid or quadruped [7, 8]. These robots exhibit both active movement for navigation and wobble caused by their instability. The inherent mobility of these platforms requires skilled operators who can maintain stable action outputs while compensating for base movement disturbances during data collection. This transforms what would otherwise be a straightforward data collection step into potentially the most challenging part of the entire process.

Beyond the challenges in data collection, directly deploying policies trained in stationary conditions on dynamic platforms is ineffective. First, performing policy inference on low-computational platforms carried by small robots often results in significant processing delays, causing the generated actions to become outdated due to the robot’s motion, as shown in Fig. 1. Second, current visuomotor policies [3, 4, 9] commonly employ action chunking to promote temporal action consistency, which involves generating multiple actions in a single inference for sequential execution. This approach further reduces the robot’s responsiveness to environmental changes.

Lastly, even if above challenges are overcome, policies trained on specific mobile robot struggle to generalize across different robotic systems due to their distinct locomotion characteristics.

To address the aforementioned challenges, we develop *STDArm*, which transfers visuomotor policy from static data training to dynamic platforms with negligible cost, equipping various mobile robots with robust capabilities inherited from imitation learning. Our key insight is that, in addition to visual and arm pose inputs, *STDArm* incorporates additional high-frequency observations of the robot’s position and employs a lightweight network to perform real-time action corrections based on the robot’s movement. By increasing the control frequency, the system achieves swift responsiveness to changes in robot’s state. Specifically, an action manager module is first designed to decouple the policy network’s asynchronous inference from the real-time control loop, while interpolating the policy outputs to higher frequency. This establishes high-frequency control as the foundation of the entire system. Building on this, *STDArm* combines high-frequency ego-motion tracking through a visual SLAM system – operating at 10× the base policy’s observation rate – with a lightweight neural network that bridges the perception-action latency gap. This integrated solution enables real-time compensation for platform motion disturbances while maintaining the original policy’s manipulation capabilities, achieving centimeter-level operational precision during mobile manipulation tasks. Finally, to align *STDArm*’s optimization objectives with task execution goals, we implement an action-stabilizing warm-up routine prior to mission execution, enabling online calibration of system parameters through controlled motion sequences.

To validate *STDArm*’s effectiveness, we conduct experiments across two distinct robotic arm configurations and four mobile platforms, including drones. Two canonical visuomotor policies, diffusion policy (DP) [4] and ACT [3], serve as the foundational policy networks. We design three manipulation tasks with progressively stringent precision requirements to evaluate *STDArm*’s manipulation stability during robot’s movement. The most challenging task demands end-effector positioning accuracy within 4 cm. Experimental results demonstrate that while the native policy achieves near-zero success rates during the robot’s movement, *STDArm* attains success rates comparable to those of the native policy under ideal static conditions. These experimental configurations validate that *STDArm* enables cost-effective migration of static-trained policy networks to diverse dynamic platforms.

Overall, our contributions are as follows:

- We present an efficient system, *STDArm*, enabling direct deployment of static-trained policy networks to mobile robotic platforms with negligible cost, while preserving the policy’s original performance.
- The designed action manager module is incorporated to enhance robotic decision-making and control frequencies, forming the foundational architecture of *STDArm*.
- By predicting the robot’s state and estimating system latency online, we precisely maintain the stability of manipulation during the robot’s movement.

- To validate our approach, we conduct experiments using two types of robotic arms across three different tasks. Furthermore, we deploy *STDArm* on a drone in a real-world scenario to validate its effectiveness.

II. RELATED WORK

A. Imitation Learning

Imitation learning enables robots or agents to rapidly acquire complex behaviors by leveraging expert-provided demonstrations. In recent years, this approach has been widely applied to tasks such as robot manipulation [3, 4, 10, 11, 12, 13, 14], navigation [15, 16], and mobile manipulation [17, 7, 8]. Diffusion Policy [4] leverages the robust multi-modal modeling capabilities of diffusion models to better address the challenges posed by the diversity in robot manipulation demonstrations. π_0 [11] integrates a pre-trained multimodal language model with a flow-matching-based diffusion action head to achieve efficient and generalized robotic policies. CARP [13] aims to design an autoregressive generation policy that extracts multi-scale representations of the entire robot action sequence, generating action sequence predictions through a coarse-to-fine autoregressive process. The efficient collection of high-quality expert data is one of the crucial prerequisites for the effectiveness of the aforementioned imitation learning methods. However, unstable and dynamic robotic platforms, like drones and legged robots, introduce significant challenges to the data collection process. To address this, *STDArm* introduces an efficient system that allows policies trained on data collected from static platforms to be seamlessly transferred for inference on unstable and dynamic robotic platforms.

The dynamically unstable platform also imposes requirements on the inference frequency of the policy, as a higher inference speed can better adapt to the dynamic changes in the task settings. In particular, the recently popular diffusion policy faces challenges in improving inference frequency due to its need for multiple iterative denoising steps. Consistency Policy [18, 19] attempts to use consistency distillation to enable a student network to learn a direct mapping over longer step spans on the probability flow ODE. AdaFlow [20], based on flow-based generative models, adaptively adjusts the computational cost during inference according to the variance of the state, thereby significantly improving inference efficiency while maintaining decision diversity. The above methods all attempt to reduce the number of denoising steps in diffusion policy to increase the overall policy inference frequency. Meanwhile, BID [21] creates action chunks at each observation stamp and uses two loss functions to select chunks that try achieving both long-term consistency and short-term reactivity. However, *STDArm* adopts a more direct approach by using an action manager and interpolated action compensation, ensuring swift responses to state changes even on low-computational platforms.

Theoretically, the system is designed to be compatible with a wide range of learning-based algorithms, ensuring flexibility and broad applicability.

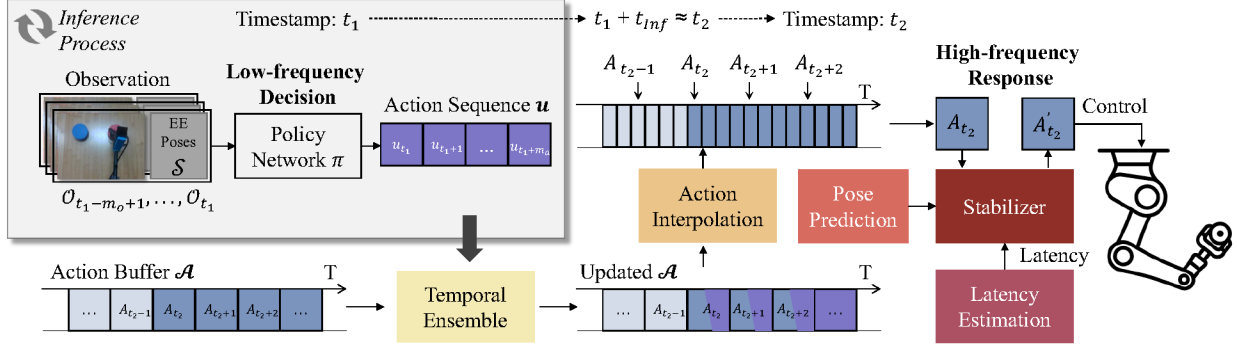


Fig. 2. Pipeline of *STDArm*. Given images and joint state observations, our system first generates a low-frequency action sequence via a policy network. These actions are then passed to action manager, which maintains an action buffer using temporal ensemble and generates high-frequency actions through interpolation. Subsequently, a stabilizer refines the action based on estimated latency and real-time pose predictions to compensate for the motion of the robot platform. Finally, the stabilized action is sent to the robotic arm for precise and stable manipulation.

B. Mobile Manipulation

Early methods for mobile manipulation have shown notable success in tasks like grasping, including ground-based mobile manipulation [22, 23, 24, 25, 26, 27, 28] and unmanned aerial manipulation (UAM) [29, 30, 31]. However, these approaches often depend on human engineered designs tailored to specific scenarios or require extensive data collection for training, which restricts their adaptability across diverse engineering applications. To mitigate these limitations, MobileALOHA [17] introduces a cost-effective wheeled dual-arm system capable of remote operation to streamline data collection and relies on ACT [3] for policy learning and deployment. Despite its contributions, MobileALOHA is primarily designed for static operations within similar environmental conditions, limiting its broader applicability. Recognizing these constraints, UMI on Legs [7] focuses on efficient data collection and Diffusion Policy [4] deployment across a variety of dynamic legged robots. Its performance is validated on platforms such as quadrupeds, demonstrating increased adaptability. However, when applied to highly dynamic platforms like drones, manipulation faces heightened challenges in achieving precise and robust control due to the dynamic and unstable nature of their bases. Addressing these challenges, our *STDArm* builds on the strengths of previous methods, enabling efficient data collection and stable deployment of imitation learning for manipulation operating on more challenging platforms, including drones. This advancement underscores the potential for enhanced control and adaptability in complex and dynamic environments.

III. STDARM

As shown in Fig. 2, we present the *STDArm* architecture through its workflow, tracing the data flow from RGB image inputs and joint state measurements to final action execution. We initially employ a foundational policy network to generate action sequence for low-frequency decision-making, as detailed in Sec. III-A. Then, an action manager module enhances control frequency through temporal ensemble and action interpolation strategy, as discussed in Sec. III-B. To

enable real-time compensation for platform motion disturbances, the stabilizer module, detailed in Sec. III-C, facilitates the conversion of preliminary actions into the final executed actions. Finally, we design an action-stabilizing warm-up routine before mission execution in Sec. III-D, to perform online parameter estimation essential for the stabilizer module.

It is worth mentioning that all these modules are deployed on edge devices without the assistance of external platforms for computation.

A. Policy Network

The policy network π takes m_o steps of observations $\mathcal{O}$, including images $\mathcal{I}$ and end-effector poses $\mathcal{S}$ as input. Based on these observations, the network predicts the action sequence $\mathbf{u}$ for the next m_a time steps. Both the observation steps m_o and prediction steps m_a are predefined based on prior experience. Denoting the action sequence as $\mathbf{u}_{t+1:t+m_a} = \{u_{t+1}, \dots, u_{t+m_a}\}$, this process can be formulated as:

$$\mathbf{u}_{t+1:t+m_a} = \pi(\mathcal{I}_{t-m_o+1:t}, \mathcal{S}_{t-m_o+1:t}), \quad (1)$$

where u is defined as the end-effector's position, which will be used to control the robotic arm. In order to avoid prolonged policy inference blocking the entire system, the policy network π is assigned a dedicated subprocess. Following each inference cycle, the policy network immediately retrieves the latest observations to ensure a continuous generation of action sequences. This approach enhances the decision-making frequency of the policy network, which is beneficial for handling dynamic scenarios.

B. Action Manager

To effectively select the appropriate actions for execution from the continuously generated action sequences, we design an action manager module to handle the output of the policy network. First, the action manager maintains an action buffer $\mathcal{A} = \{A_t, t \in \mathbb{N}\}$, which sequentially stores both recently executed actions and predicted actions awaiting execution. We define A_t to be in the same form as u . A temporal ensemble algorithm is then employed to merge the newly generated action sequences $\mathbf{u}$ with the contents of the action buffer $\mathcal{A}$.

Finally, action interpolation is applied to enhance the control frequency of the robotic arm. The details are as follows:

1) *Temporal Ensemble*: As illustrated in Fig. 2, π reads the most recent observation $\mathcal{O}_{t_1-m_o+1:t_1}$ at time t_1 , and after an inference duration of t_{Inf} , generates an action sequence $\mathbf{u}_{t_1+1:t_1+m_a}$. The action manager at time $t_2 \approx t_1 + t_{\text{Inf}}$ must merge $\mathbf{u}_{t_1+1:t_1+m_a}$ with current action buffer $\mathcal{A}$, where “ $\approx$ ” indicates that t_2 is the first execution moment after obtaining the action sequence generated from the observation at t_1 . A critical issue of merging lies in determining the correspondence between actions in $\mathbf{u}_{t_1+1:t_1+m_a}$ and those already in $\mathcal{A}$. Aligning them by timestamp is a straightforward approach, treating the action at t_2 in $\mathbf{u}_{t_1+1:t_1+m_a}$ as corresponding to the action at t_2 in $\mathcal{A}$. However, in practice, we observe that due to inconsistencies in policy inference, actions predicted for the same timestamp may not match, potentially leading to action backtracking, i.e., repeatedly executing previously completed actions. To address this, we use timestamp alignment as an initial solution and select the appropriate match by comparing the similarity of actions as follows:

$$t_u = \arg \min_t \frac{\sum_{k \in \mathbb{K}} d(u_{t+k}, A_{t_2+k})}{|\mathbb{K}|}, \quad (2)$$

$$t \in [t_2 - m_s, t_2 + m_s],$$

where $d()$ denotes the Euclidean distance, $\mathbb{K}$ represents the overlapping range between $\mathbf{u}_{t_1+1:t_1+m_a}$ and $\mathcal{A}$, $|\mathbb{K}|$ is the cardinality of $\mathbb{K}$, and m_s is the predefined search range. The resulting t_u indicates that $\mathbf{u}_{t_u:t_u+k}$ corresponds to $\mathcal{A}_{t_2:t_2+k}$.

After establishing the correspondence, inspired by ACT [3], we employ temporal ensemble to further enhance the smoothness of the actions. Specifically, the action buffer $\mathcal{A}$ is updated as follows:

$$A_{t_2+k} = \begin{cases} w_t^{\text{old}} \cdot A_{t_2+k} + w_t^{\text{new}} \cdot u_{t_u+k}, & 0 \leq k \leq |\mathcal{A}| - t_2, \\ u_{t_u+k}, & k > |\mathcal{A}| - t_2, \end{cases} \quad (3)$$

where the weights w_t^{new} of $\mathbf{u}_{t_1+1:t_1+m_a}$ are computed following an exponential distribution parameterized by the hyperparameter α . For the action manager, the weight of each time step is initialized to 0. After performing temporal ensemble, the weight of each time step is updated by adding the corresponding weight of the predicted action, i.e.:

$$w_{t+i}^{\text{new}} = \exp(-\alpha i), i \in [1, m_a], \quad (4)$$

$$w_t^{\text{old}} = w_t^{\text{old}} + w_t^{\text{new}}. \quad (5)$$

2) *Action Interpolation*: Additionally, considering the relatively low frequency of the policy network’s action sequence output compared to the high frequency required for actual action execution, we perform linear interpolation for actions at finer time step τ . Specifically, for τ located between t and $t+1$, the interpolated action can be expressed as:

$$A_\tau = (\tau - t)A_t + (t + 1 - \tau)A_{t+1}. \quad (6)$$

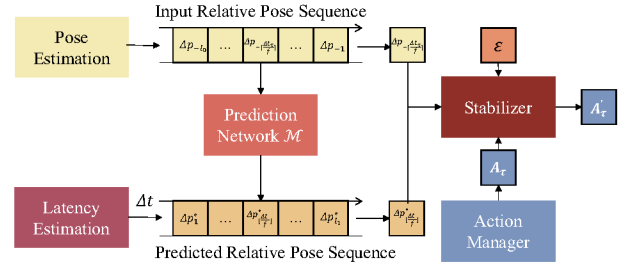


Fig. 3. By utilizing the pose at action generation, the predicted pose, and the extrinsic parameters between the robotic arm and the pose estimator as inputs, the stabilizer compensates for each action output by the action manager, whether derived from the policy network or interpolation, thereby counteracting platform motion.

The continuous merging of action sequences into the action buffer ensures that interpolation is always performed internally, avoiding the risks associated with external extrapolation.

C. Prediction Compensation Based Stabilizer

After obtaining high-frequency actions in real time through the action manager, we still need to address the impact of the robot’s motion on manipulation, such as the robot’s active movement or the wobble caused by its instability. The most direct approach involves monitoring the robot’s motion and using this data to refine its arm actions. However, considering computation, communication and control delays, this method adjusts the robotic arm’s action using outdated robot position data, which significantly reduces the success rate of high-precision manipulation tasks. Therefore, we design a prediction compensation based stabilizer, which predicts the platform’s motion over a short future period and modifies the actions to counteract the motion effectively.

As shown in Fig. 3, we first implement a pose prediction module incorporating two parts: a SLAM module for pose estimation and a lightweight prediction network. In terms of data flow, we employ a real-time, high-frequency visual SLAM using stereo camera with an IMU¹ to obtain a high-frequency sequence of poses. Then, for each prediction in prediction network $\mathcal{M}$, we use the current pose p_0 as the base coordinate system and calculate the sequence of high-frequency relative pose $\Delta P_{-l_0:-1} = \{\Delta p_{-l_0}, \dots, \Delta p_{-1}\}$ from the past l_0 frames, where $\Delta p_{-i} = p_0^{-1} \cdot p_{-i}$ and $\Delta P_{-l_0:-1}$ serves as the input to the prediction network. The prediction network directly outputs the future l_1 frames’ high-frequency relative pose sequence $\Delta P_{1:l_1}^* = \{\Delta p_1^*, \dots, \Delta p_{l_1}^*\}$, which predicts the future pose by $p_i^* = p_0 \cdot \Delta p_i^*$.

Considering the limitation of computational resources and the requirements for system real-time performance, we implement a lightweight prediction network $\mathcal{M}$. Specifically, we parallelize an LSTM [32] and a GRU [33], which process the same input. We concatenate their corresponding temporal outputs along the feature dimension and finally pass them through a fully connected layer to produce the prediction

¹<https://www.intelrealsense.com/visual-inertial-tracking-case-study/>

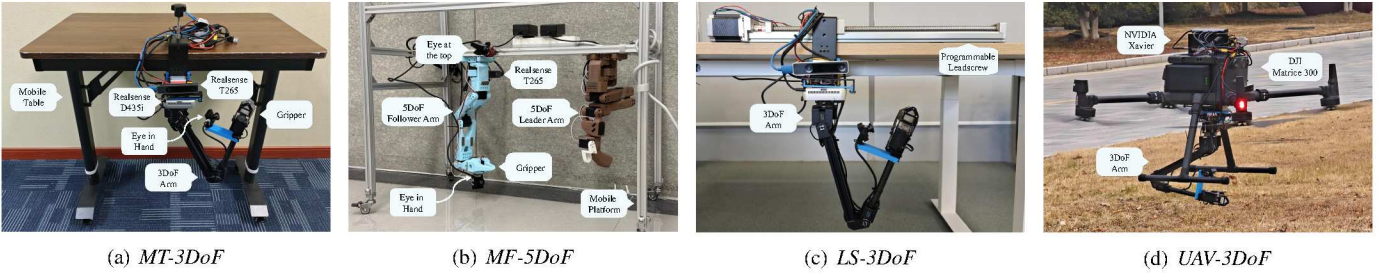


Fig. 4. Four experimental configurations we design. Each is equipped with two cameras for observation inputs and a T265 camera for pose estimation. The robotic arms operate independently of the robot platforms, with *MT-3DoF*, *LS-3DoF* and *UAV-3DoF* sharing the same arm.

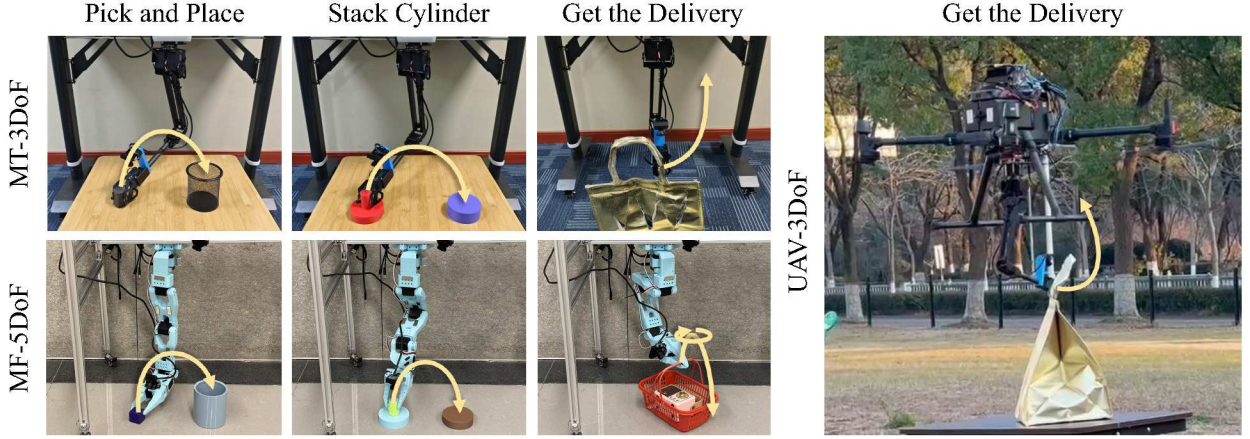


Fig. 5. We design three tasks of varying difficulty levels and conduct experiments across three experimental configurations.

results, that is:

$$\Delta P_{1:l_1}^* = \mathcal{M}(\Delta P_{-l_0:-1}), \quad (7)$$

$$\mathcal{M}(x) = \text{FC}(\text{concat}(\text{LSTM}(x), \text{GRU}(x))). \quad (8)$$

Ultimately, the action offset caused by robot motion for the target action A_τ can be expressed as:

$$\delta(\tau, \Delta t, \mathcal{E}) = \mathcal{E}^{-1} \cdot (\Delta p_{[-\Delta t_\tau/f]}^{-1} \cdot \Delta p_{[\Delta t/f]}^*)^{-1} \cdot \mathcal{E}, \quad (9)$$

where $\mathcal{E}$ denotes the extrinsics between the visual SLAM coordinate system and the arm coordinate system, manually pre-calibrated; Δt_τ represents the time interval between the generation of A_τ by policy network π and its execution; f indicates the output frequency of the SLAM system. Δt , which will be estimated online in Sec. III-D, is the total latency spanning from pose estimation to action execution, incorporating processing time for pose estimation, pose prediction, data transmission, and action execution. The compensated action ultimately executed by the robot becomes

$$A'_\tau = \delta(\tau, \Delta t, \mathcal{E}) \cdot A_\tau. \quad (10)$$

D. Latency Estimation

To align *STDArm*'s optimization objectives with task execution goals, we implement a pre-mission warm-up routine to perform online calibration of system latency, thereby enabling the stabilizer module to accurately adjust the actions. To achieve this, the stabilizer module is employed to help the

robot do the end-effector stabilization task, enabling quantitative assessment of various latencies' impact on operational stability. The gripper maintains hold of a red marker while a fixed monocular third-view camera tracks its positional variance during robotic arm base moving. We conduct a linear search for the latency starting from zero, identifying the latency that minimizes the marker's movement as the system delay. To ensure that variations of the robot's movement don't affect the correctness of the latency estimation, we adopt the velocity ratio between the marker's pixel-plane motion and the VSLAM-estimated platform motion as the evaluation metric.

IV. IMPLEMENTATION

A. Hardware

Our *STDArm* is a modular mobile manipulation system, which consists of three core components: 1) a reconfigurable mobile base platform, 2) a multi-DoF robot arm, and 3) a computing platform. For imitation learning policies inputs, the arm is equipped with two RGB cameras: one near the end effector to monitor interaction dynamics and another at the arm base to observe objects of interest. A forward-facing camera is additionally integrated to support real-time pose estimation.

To validate system versatility, as illustrated in Fig. 4, we implement four experimental configurations:

- **MT-3DoF**: Mobile Table with 3-DoF arm²

²<https://www.waveshare.com/product/robotics/robot-arm-control/robot-arm/roarm-m2-s.htm>

- **MF-5DoF**: Mobile aluminum frame with 5-DoF arm ³
- **LS-3DoF**: Programmable Leadscrew with 3-DoF arm
- **UAV-3DoF**: DJI Matrice 300⁴ with 3-DoF arm

All configurations utilize an NVIDIA Jetson AGX Xavier Developer Kit ⁵ as the computing platform. Specially, all components of *UAV-3DoF* except the drone are powered via USB interfaces from the drone.

B. Tasks Design

As shown in the Fig. 5, we design three tasks with varying requirements for operational precision as follows:

- **Pick and Place**. We first conduct a classic task in robot manipulation, where the robot needs to pick up a block and throw it into a small container. The block is approximately 3 cm wide, and the pen holder has a diameter of about 8 cm.
- **Stack Cylinder**. We attempt to stack two cylinders, each with a diameter of 8 cm. This requires the robotic arm to have a control precision within 4 cm, as any deviation beyond 4 cm results in a failed stack. To facilitate gripping, we attach a cylindrical handle with a diameter of 2.5 cm to one of the cylinders.
- **Get the Delivery**. Finally, we attempt to complete a real-world task—picking up takeout. The robot needs to extend its gripper near the handle of a bag or basket, grasp the bag, and lift it. We successfully perform this task using a drone as well.

These tasks are validated on *MT-3DoF* and *MF-5DoF*. However, on *UAV-3DoF*, only the **Get the Delivery** task is tested. This limitation is that the wind generated by the drone can blow away the target objects of the other two tasks.

V. EXPERIMENTS

A. Experimental Setup

For each task described in Sec.IV-B, we gather 50 sequences of stationary state data to train the policy network, with target objects randomly placed within the robotic arm’s reachable workspace. Additionally, for each configuration outlined in Sec.IV-A, we collect approximately five minutes of motion data to train the pose prediction network.

In the case of *MT-3DoF* and *MF-5DoF*, we simulate mobile scenarios by manually shaking the platforms. To ensure fairness, we employ a double-blind protocol where the operator shaking the platform is unaware of the specific algorithm being tested and receives no visual feedback during the process, focusing solely on generating random perturbations. This methodology not only ensures impartiality but also inherently demonstrates the robustness of our approach by leveraging the natural variability of human motion.

For *LS-3DoF*, a programmable leadscrew moves the arm base at 12 cm/s to provide perturbations. Since the motion

is precisely controllable, *LS-3DoF* provides reproducible experimental conditions, but its regular motion makes trajectory prediction straightforward. For this reason, we only employ it for ablation studies

Regarding *UAV-3DoF*, the aerial platform’s inherent instability arises from its dynamic manipulation environment, including self-induced aerodynamic disturbances and limited positional precision during hovering. We leverage these natural instability characteristics rather than introducing artificial perturbations. Notably, to accommodate the manipulation’s operational range constraints and the drone’s finite positional holding capacity, an operator manually maintains the relative positioning between the drone and the delivery target. This operational paradigm is adopted because that fully autonomous drone navigation exceeds the scope of this study.

As for the specific network architecture, we employ ResNet-18 [34] as vision backbone of the policy network, with input images of shape $3 \times 480 \times 640$. The observation and prediction horizons are configured differently for each method: For DP, we set the observation horizon to 2 and action prediction horizon to 8 at 5Hz, utilizing a transformer [35] combined with DDIM [36] for denoising with 20 steps during inference. For ACT, we configure the observation horizon to 1 and action prediction horizon to 100 at 30Hz. In the action manager, actions are interpolated to 50Hz, and the temporal ensemble parameter α is set to 1. The pose prediction network takes a 1s pose sequence as input and generates 0.5s predictions, where pose frequency is 200Hz. This configuration ensures optimal real-time performance under constraints including inference latency, communication latency, and edge device performance variability. The code framework is adapted from the lerobot codebase [37].

B. Experiment Results

The experimental results presented in Tab. I demonstrate the efficacy of *STDarm* in enabling policy networks trained under static conditions to operate effectively on dynamically unstable platforms. Under static configurations, baseline methods (DP and ACT) achieve satisfactory performance across all tasks, *MF-5DoF* attaining perfect scores in all tasks with ACT. However, dynamic scenarios reveal severe performance degradation for baseline methods, particularly for precision-demanding tasks like Pick and Place (up to 0% success rate) and Stack Cylinder (up to 13.3% success rate).

The integration of *STDarm* yields significant improvements, with 46.7-53.3% relative performance recovery in dynamic Pick and Place and Stack Cylinder. Moreover, *MT-3DoF* and *MF-5DoF* achieved improvements of 73.3% and 60% in Get the delivery, respectively. Notably, the aerial platform (*UAV-3DoF*) exhibits distinct behavioral characteristics, where baseline methods achieve only 13.3% success rate in dynamic Get the Delivery, while *STDarm*-enhanced policies demonstrate a 200% performance improvement (40% success rate). This performance gap highlights the method’s adaptability to different instability sources, ranging from manual platform perturbations (*MT-3DoF*/*MF-5DoF*) to inherent aerodynamic

³<https://github.com/TheRobotStudio/SO-ARM100>

⁴<https://www.dji.com/support/product/matrice-300>

⁵<https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/>

TABLE I
SUCCESS RATE OF OUR CONFIGURATIONS AND TASKS. THE BEST IN DYNAMIC IS IN BOLD.

Operation Mode	Configurations	Method	Pick and Place	Stack Cylinder	Get the Delivery
Static	<i>MT-3DoF</i>	DP	73.3%	66.7%	100%
		DP+ <i>STDArm</i>	73.3%	80%	100%
	<i>MF-5DoF</i>	ACT	93.3%	100%	100%
		ACT+ <i>STDArm</i>	100%	100%	100%
Dynamic	<i>MT-3DoF</i>	DP	0%	0%	20%
		DP+ <i>STDArm</i>	53.3%	53.3%	93.3%
	<i>MF-5DoF</i>	ACT	6.7%	13.3%	26.7%
		ACT+ <i>STDArm</i>	53.3%	66.7%	86.7%
	<i>UAV-3DoF</i>	DP	\	\	13.3%
		DP+ <i>STDArm</i>	\	\	40%

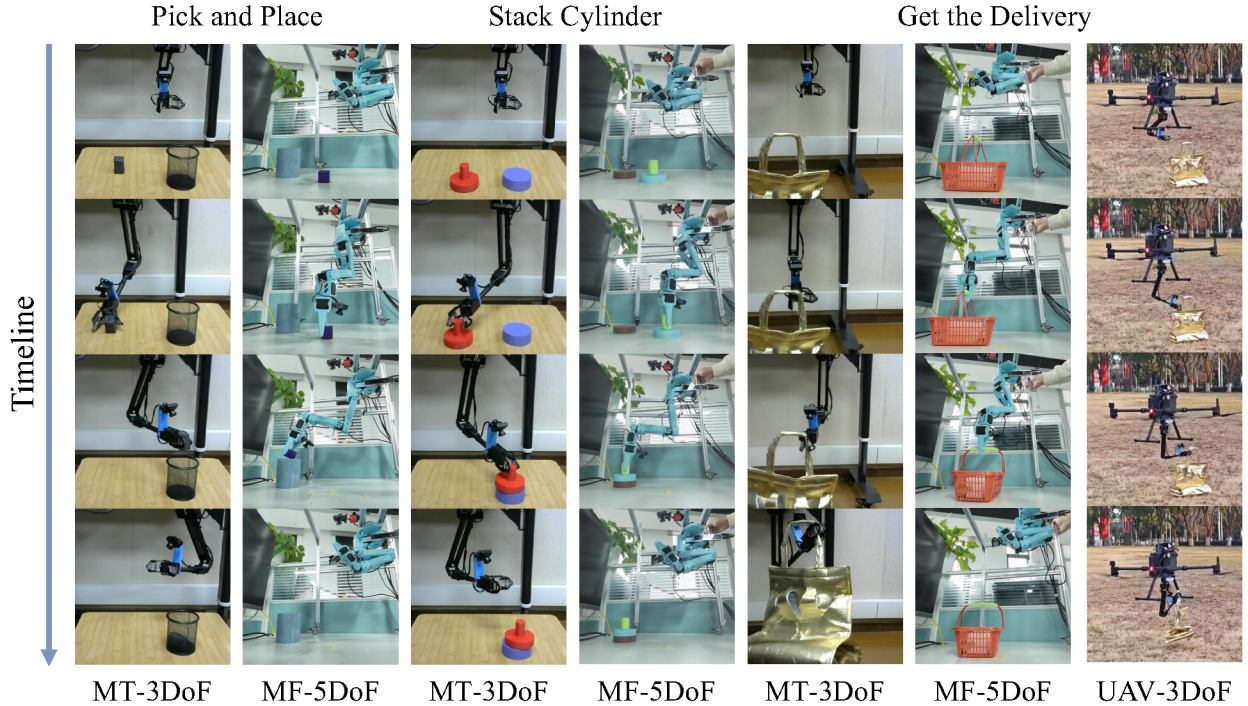


Fig. 6. Temporal task progression during experiments. Beyond task completion metrics, the motion of the platforms also worth noting. Videos on the project page visually demonstrate that the *STDArm* enables real-time compensation for platform motion disturbances while preserving the original policy’s manipulation capabilities.

uncertainties (*UAV-3DoF*). Fig. 6 illustrates the robotic arm’s task execution process during platform movement.

It should be noted that we use two different robotic arms and two distinct policies to demonstrate *STDArm*’s plug-and-play capability. Consequently, we did not conduct experiments to compare different policies (e.g., DP vs. ACT) on the same configurations.

C. Ablation Study

To account for the potential controversy surrounding the manual influence of shaking the platform, we conduct two separate ablation studies in the Get the Delivery task. The first study employs *MT-3DoF* and *MF-5DoF* to evaluate the incremental contributions of the components of *STDArm*. The second study further systematically evaluates each component’s impact across various conditions using *LS-3DoF*.

As shown on Fig. 7, starting from a baseline static policy that directly employs the trained DP or ACT model without dynamic adaptation, we observe the lowest success rates

(20% on *MT-3DoF* and 26.7% on *MF-5DoF*) due to its inability to handle dynamic state changes. Introducing the action manager improves performance by 20% and 26.7% respectively, though inference latency still limit operational stability. Subsequent integration of the stabilizer module’s pose prediction mechanism with “latency=0s” or “latency=0.5s” does not noticeable improvement compare to the action manager baseline. Addressing system latency through our latency estimation module, *STDArm* with all components achieves peak success rates of 93.3% on *MT-3DoF* and 86.7% on *MF-5DoF* across 15 experimental trials, establishing 73.3%/60% absolute improvement over the initial static policy. This progressive performance gain confirms the necessity of our approach to achieve precise robotic manipulation in dynamic environments.

For the second ablation study detailed in Tab. II, each group involves 15 trials using DP as the base policy. Results underscore three key elements: temporal ensemble (No.0 vs. No.1 & No.2 vs. No.6), latency estimation (No.5 vs. No.7), and

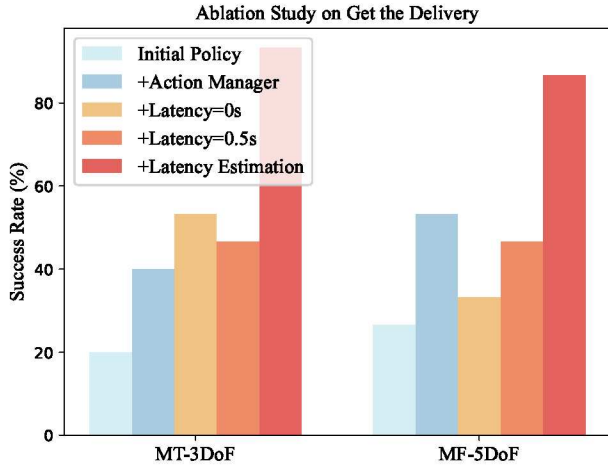


Fig. 7. Ablation Study with *MT-3DoF* and *MF-5DoF* on Get the Delivery

TABLE II
ABLATION STUDY WITH *LS-3DoF* ON GET THE DELIVERY.

No.	Action Manager		Latency=0	Latency Estimation	Success Rate(%)
	Temporal Ensemble	Action Interpolation			
0					0
1	✓				20
2				✓	40
3	✓	✓			26.6
4	✓		✓		33.3
5	✓	✓	✓		33.3
6	✓			✓	53.3
7	✓	✓		✓	80

action interpolation (No.6 vs. No.7). Ablations demonstrate that continuous decision-making and high-frequency control generated by the action manager, along with latency-aware action compensation, are essential components.

D. Accuracy of End Hold

To more intuitively demonstrate the contribution of our method to end-effector stability, we conduct an end-effector holding task on *MT-3DoF* to visualize performance. As depicted in Fig. 8(a), the robot’s base is swayed left and right along with the table for 30 seconds while tracking the movement trajectory of a red ball held in the gripper. As shown in Fig. 8(b), without any corrective actions, the distribution of the red ball mirrors the movement of the manipulation, exhibiting a linear pattern. When employing the “latency=0s” correction, the situation improved, as seen in Fig. 8(c). However, due to the lack of prediction of the base’s movement, the corrections to the manipulation’s actions are delayed, resulting in a relatively large range of end-effector movement. Finally, with the integration of our pose prediction module, the delay from pose changes to manipulation control is effectively mitigated, as illustrated in Fig. 8(d). This advancement significantly enhanced the stability of the end-effector.

VI. CONCLUSION

In this work, we introduce *STDArm*, a system that effectively transfers visuomotor policies trained under static data to dynamic robotic platforms. By addressing key challenges

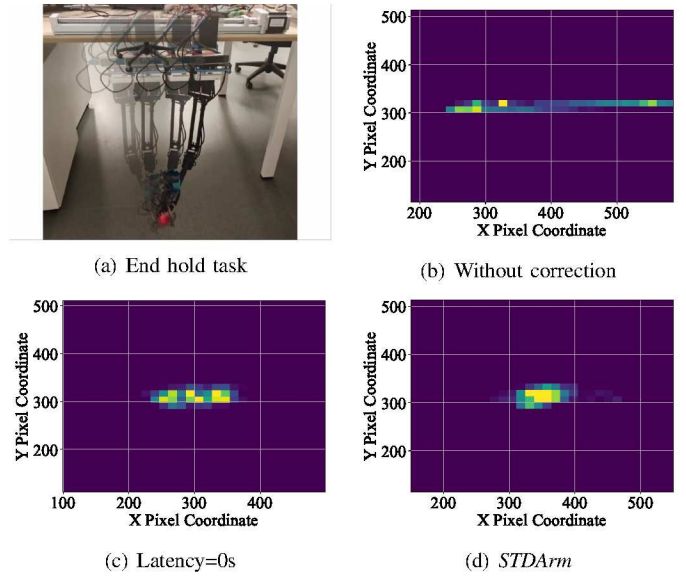


Fig. 8. The visualization of the end hold task and the distribution heatmaps of the red marker.

such as decision-making latency, platform-induced motion disturbances, and asynchronous control loops, *STDArm* significantly enhances manipulation stability during robot movement. Our design integrates an action manager for high-frequency control, a lightweight prediction network to bridge perception-action delays, and an end-effector stabilizer that ensures precise execution independent of the robot’s motion.

Through extensive evaluations across multiple robotic arms, mobile platforms, and manipulation tasks, we demonstrated that *STDArm* achieves centimeter-level precision, even in highly dynamic conditions. Our results confirm that *STDArm* enables real-time compensation for motion disturbances, achieving success rates comparable to static-trained policies under ideal conditions. This validates its effectiveness as a cost-efficient and generalizable solution for deploying visuomotor policies across diverse mobile robots.

Limitations and future work A key assumption of all the contributions presented in this paper is that the foundational policy network is functional. However, during a robot’s movement, inevitable changes in the camera’s perspective, the background of the target object, and other factors pose significant challenges to the policy network’s generalization. Our method does not address this issue. In the future, we will focus on specifically enhancing the generalization capability of the policy network to improve its performance in challenging dynamic scenarios.

ACKNOWLEDGMENTS

We gratefully acknowledge Prof. Chong Li from OUC and Yu Liu from USTC for their expert advice on control theory. This work was supported by the National Natural Science Foundation of China (No. 62332016) and the Key Research Program of Frontier Sciences, CAS (No. ZDBS-LY-JSC001).

REFERENCES

- [1] Julen Urain, Ajay Mandlekar, Yilun Du, Mahi Shafiullah, Danfei Xu, Katerina Fragkiadaki, Georgia Chalvatzaki, and Jan Peters. Deep generative models in robotics: A survey on learning from multimodal demonstrations. *arXiv preprint arXiv:2408.04380*, 2024.
- [2] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.
- [3] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *Robotics: Science and Systems (RSS)*, 2023.
- [4] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.
- [5] Thomas Weng, Sujay Man Bajracharya, Yufei Wang, Khush Agrawal, and David Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In *Conference on Robot Learning*, pages 192–202. PMLR, 2022.
- [6] Mingyo Seo, Steve Han, Kyutae Sim, Seung Hyeon Bang, Carlos Gonzalez, Luis Sentis, and Yuke Zhu. Deep imitation learning for humanoid loco-manipulation through human teleoperation. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE, 2023.
- [7] Huy Ha, Yihuai Gao, Zipeng Fu, Jie Tan, and Shuran Song. Umi on legs: Making manipulation policies mobile with manipulation-centric whole-body controllers. *arXiv preprint arXiv:2407.10353*, 2024.
- [8] Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetstein, and Chelsea Finn. Humanplus: Humanoid shadowing and imitation from humans. *arXiv preprint arXiv:2406.10454*, 2024.
- [9] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Robotics: Science and Systems (RSS)*, 2024.
- [10] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.
- [11] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0 : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.
- [12] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.
- [13] Zhefei Gong, Pengxiang Ding, Shangke Lyu, Siteng Huang, Mingyang Sun, Wei Zhao, Zhaoxin Fan, and Donglin Wang. Carp: Visuomotor policy learning via coarse-to-fine autoregressive prediction. *arXiv preprint arXiv:2412.06782*, 2024.
- [14] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.
- [15] Wenhao Yu, Jie Peng, Huanyu Yang, Junrui Zhang, Yifan Duan, Jianmin Ji, and Yanyong Zhang. Ldp: A local diffusion planner for efficient robot navigation and collision avoidance. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5466–5472. IEEE, 2024.
- [16] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.
- [17] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.
- [18] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *Robotics: Science and Systems (RSS)*, 2024.
- [19] Guanxing Lu, Zifeng Gao, Tianxing Chen, Wenxun Dai, Ziwei Wang, and Yansong Tang. Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation. *arXiv preprint arXiv:2406.01586*, 2024.
- [20] Xixi Hu, Bo Liu, Xingchao Liu, and Qiang Liu. Adafflow: Imitation learning with variance-adaptive flow-based policies. *arXiv preprint arXiv:2402.04292*, 2024.
- [21] Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Maximilian Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via closed-loop resampling. *arXiv preprint arXiv:2408.17355*, 2024.
- [22] Max Bajracharya, James Borders, Dan Helmick, Thomas Kollar, Michael Laskey, John Leichty, Jeremy Ma, Umashankar Nagarajan, Akiyoshi Ochiai, Josh Petersen, et al. A mobile manipulation system for one-shot teaching of complex tasks in homes. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11039–11045. IEEE, 2020.
- [23] Joel Chestnutt, Manfred Lau, German Cheung, James Kuffner, Jessica Hodgins, and Takeo Kanade. Footstep planning for the honda asimo humanoid. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 629–634. IEEE, 2005.

- [24] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G Atkeson. Optimization based full body control for the atlas robot. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 120–127. IEEE, 2014.
- [25] Oussama Khatib, K Yokoi, K Chang, D Ruspini, R Holmberg, A Casal, and A Baader. Force strategies for cooperative tasks in multiple mobile manipulation systems. In *Robotics Research: The Seventh International Symposium*, pages 333–342. Springer, 1996.
- [26] Keenan A Wyrobek, Eric H Berger, HF Machiel Van der Loos, and J Kenneth Salisbury. Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. In *2008 IEEE International Conference on Robotics and Automation*, pages 2165–2170. IEEE, 2008.
- [27] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.
- [28] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023.
- [29] Yanjie Chen, Yangning Wu, Zhenguo Zhang, Zhiqiang Miao, Hang Zhong, Hui Zhang, and Yaonan Wang. Image-based visual servoing of unmanned aerial manipulators for tracking and grasping a moving target. *IEEE Transactions on Industrial Informatics*, 19(8): 8889–8899, 2022.
- [30] Huazi Cao, Yongqi Li, Cunjia Liu, and Shiyu Zhao. Eso-based robust and high-precision tracking control for aerial manipulation. *IEEE Transactions on Automation Science and Engineering*, 21(2):2139–2155, 2023.
- [31] Meng Wang, Zeshuai Chen, Kexin Guo, Xiang Yu, Youmin Zhang, Lei Guo, and Wei Wang. Millimeter-level pick and peg-in-hole task achieved by aerial manipulator. *IEEE Transactions on Robotics*, 2023.
- [32] S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.
- [33] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [35] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- [36] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.
- [37] Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. <https://github.com/huggingface/lerobot>, 2024.