2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Viper: Verifiable Imitation Learning Policy for Efficient Robotic Manipulation

Xianfeng Cheng[1], Qing Gao[1*], Guangyu Chen[1], Rui Xiong[1], Junjie Hu[2], Yulan Guo[1] and Zhaojie Ju[3]

***Abstract*— Imitation learning (IL) presents a promising paradigm for enabling embodied robots to efficiently acquire human-like manipulation skills. However, prevailing methods face a persistent trade-off between motion precision and computational tractability. To resolve this fundamental challenge, this paper introduces Viper, a framework for Verifiable Imitation learning Policy for Efficient Robotic manipulation. Viper integrates principles of Nonlinear Model Predictive Control (NMPC) within a learning-based model. Grounded in an NMPC-style closed-loop architecture, the proposed method unifies the modeling of nonlinear system dynamics with online, multi-horizon optimization of state-action predictions, while intrinsically embedding physical constraints. This co-design enables both smooth trajectory generation and fast execution. Furthermore, a theoretical stability analysis for the Viper framework is provided. Extensive evaluations, from simulated benchmarks to real-world manipulation tasks, demonstrate that Viper effectively reconciles the competing demands of precision and speed inherent in existing robotic IL paradigms. Project page: `https://cheng122.github.io/Viper`**

I. Introduction

Robotic imitation policy learning from demonstrations, formulated as the supervised regression task, establishes direct state-to-action mappings on expert behaviors. This paradigm has demonstrated remarkable efficacy across diverse robotic applications, enabling autonomous acquisition of complex control policies while reducing reliance on elaborate manual programming [1], [2]. In robotic IL, execution speed and precision serve as critical performance metrics that directly determine system viability, as they govern both operational efficiency and practical effectiveness [3].

As shown in Fig. 1 (a), existing classic methods frame robotic IL as an action generation task, addressing the challenges of robotic policy learning through various generative techniques. Action Chunking with Transformers (ACT), a representative of the autoregressive modeling paradigm [4]–[6], provides a simple yet efficient action generation scheme through a CVAE-based architecture. Despite the flexible design and mature exploration under constrained computational resources, it struggles to maintain temporal consistency over long-horizon tasks due to the compounding of prediction errors. In contrast, diffusion-based methods [7]–[9] offer an alternative for generating coherent action sequences. Specifically, Diffusion Policy (DP) employs iterative gradient optimization on action-distribution score functions. This approach significantly enhances multi-modal behavior generation and stabilizes control in high-dimensional action spaces through probability density modeling. However, the requisite multi-step sequential denoising process introduces substantial computational overhead, compromising real-time responsiveness and operational efficiency. The ACT and DP paradigms exhibit distinct advantages and limitations, which are often difficult to reconcile in practice. This raises a critical question: Can we design a novel paradigm for robotic IL that maintains high task success rate, while simultaneously achieving rapid responsiveness?

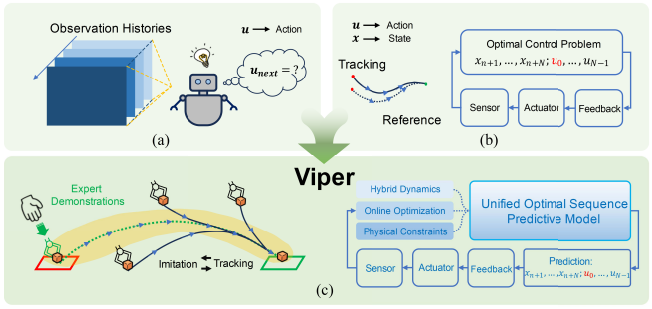

Fig. 1. Panel (a) shows the generative paradigm for IL, which directly maps historical states to actions. Panel (b) illustrates the NMPC closed-loop architecture, which leverages the current state to perform online optimization over a receding horizon, yielding an optimal action u_0. Panel (c) presents our Viper framework that integrates key insights from NMPC into IL, enabling imitation and tracking of expert trajectories.

In the field of control theory, NMPC is an optimization-based feedback control methodology (Fig. 1 (b)) for nonlinear systems [10], distinguished by its theoretical foundations, exceptional performances and extensive applications. It is primarily employed to solve "tracking" problem, which refers to constructing a system whose evolving states follow a reference trajectory as closely as possible. From a macroscopic perspective, a strong conceptual correspondence exists between "tracking" and "imitation", with "imitation" viewed as a form of intelligent "tracking". This insight motivates our exploration of structuring IL policies around the core principles of NMPC, targeting improvements in both task

This work was supported in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2025A1515011954, 2023A1515110074, in part by the Shenzhen Science and Technology Program under Grant ZDCY20250901100201002.

*Corresponding Author: gaoqing2@mail.sysu.edu.cn

[1]Xianfeng Cheng, Qing Gao, Guangyu Chen, Rui Xiong and Yulan Guo are with School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen 518107, China. (e-mail: chengxf6@mail2.sysu.edu.cn; gaoqing2@mail.sysu.edu.cn; chengy285@mail2.sysu.edu.cn; xiongr9@mail2.sysu.edu.cn; guoyulan-AT-sysu.edu.cn)

[2]Junjie Hu is with School of Artificial Intelligence, The Chinese University of Hong Kong, Shenzhen 518100, China. (e-mail: hujunjie@cuhk.edu.cn)

[3]Zhaojie Ju is with School of Computing, University of Portsmouth, Portsmouth PO13HE, UK. (e-mail: Zhaojie.Ju@port.ac.uk)

success rate and inference speed. Furthermore, the potential of adapting stability analysis techniques from NMPC is investigated to assess the stability of the proposed policy.

Based on aforementioned viewpoints, we propose Viper (Verifiable Imitation learning Policy for Efficient Robotic manipulation), as depicted in the Fig. 1 (c). Viper interprets IL from demonstrations as the intelligent tracking of expert trajectories. This is accomplished by designing a unified predictive model coupled with a suitable feedback, forming a closed-loop system inspired by the NMPC framework. The unified model is responsible for learning nonlinear dynamics, calculating optimal sequences, and respecting physical constraints. A key feature is a meticulously designed loss function, which facilitates an attempt of investigation into the system's stability of Viper via Lyapunov functions. Through extensive simulations and real-world experiments, complemented by visualizations and stability analysis, the Viper architecture is demonstrated to be both efficient and reliable.

In conclusion, our primary contributions are as follows:

- **An NMPC-Inspired IL Framework:** We introduce a paradigm that achieves efficient robotic IL under the guidance of NMPC framework. By leveraging a carefully designed unified predictive model to forecast optimal sequences and incorporating real-time feedback, Viper realizes fast, precise, and constraint-aware manipulation.
- **Interpretable Motion Stability:** We provide a potential theoretical lens for analyzing the robotic IL problem. The closed-loop system, formed by integrating the unified predictive model with a feedback, can be proven practically stability using a designed Lyapunov function. This process qualitatively and quantitatively establishes the interpretability and reliability of the Viper paradigm.
- **Comprehensive Simulation and Real-World Experiments:** Extensive experiments validate the effectiveness and robustness of Viper across a range of simulated and real-world robotic manipulation tasks.

II. Related Works

A. Robotic Imitation Learning

IL enables robots to acquire skills by mimicking expert demonstrations, circumventing the need for manual programming or intricate reward engineering [11], [12]. As a foundational IL approach, Behavioral Cloning (BC) [13] formulates IL as a supervised learning problem that directly maps observations to actions [1]. Despite its simplicity and success across various tasks, including manipulation [5], [7], [14]–[18], autonomous driving [19], [20], and automatic navigation [21], BC remains susceptible to compounding errors and covariate shift [22], [23].

To mitigate these limitations, early methods like DAgger [24] and synthetic correction techniques [25], [26] introduced interactive refinement, though often impractical in high-dimensional visual settings. Recent advancements have focused on enhancing the expressiveness and robustness of BC. Discretized policies [27], [28] capture multi-modal behaviors but suffer from exponential action space growth, while implicit models (e.g. energy-based methods [29]) offer greater flexibility at the cost of optimization stability. Diffusion-based policies [7], [30] improve precision but their multi-step inference incurs prohibitive computational costs for real-time applications. Other efforts, including voxel-based 3D representations [31], [32] and ACT [5], have improved spatial reasoning and error resilience.

However, a fundamental trade-off persists: High-accuracy models typically compromise on inference speed, while lightweight alternatives often struggle with precision. Although multimodal and 3D inputs improve generalization and accuracy of action execution, they also introduce considerable computational overhead.

To bridge this gap, we focuses on the architectural design and propose a novel IL framework that explicitly incorporates principles from NMPC, enabling fast inference and high task performance in complex manipulation settings.

B. Nonlinear Model Predictive Control

Rooted in optimal control theory [33], [34], NMPC has evolved into a cornerstone methodology for complex robotic systems through its distinctive receding-horizon optimization paradigm. Fundamentally, NMPC is an optimization-based feedback control methodology for nonlinear systems [10]. It ensures temporal consistency through multistep dynamic propagation, constrained trajectory optimization, and closed-loop replanning. Interestingly, a striking conceptual parallel exists between NMPC and IL: Both paradigms map the current state (and potentially its history) to a sequence of control actions. However, NMPC distinguishes itself through its rigorous model-based formulation and inherent theoretical guarantees, such as stability and constraint satisfaction. This parallel raises the intriguing possibility of adapting the well-established analytical tools from the NMPC framework to analyze and validate IL-based policies.

While recent years have witnessed continued theoretical refinements in NMPC [35], [36], alongside its growing adoption in autonomous driving [37], robotics [38], [39], and power electronics [40], [41], its integration with IL for robotic manipulation remains underexplored. Our work is the first time to conceptualize the IL problem through the lens of NMPC, yielding a dual contribution: First, a novel and performant method that integrates the structural advantages of NMPC is proposed. Second, we attempt to verify the stability of the proposed learning-based control algorithm via the analysis tools from NMPC.

III. Method

A. Basic NMPC Algorithm

Suppose a controlled process is given whose state $x(n) \in X = \mathbb{R}^d$ is measured at discrete time instants $t_n, n = 0, 1, 2, ...$ "Controlled" means that at each time instant, a control input $u(n) \in U = \mathbb{R}^m$ can be selected, which influences the future behavior of the system state. It's assumed that the next state x^+ depends on the current state x and

Algorithm 1 Basic NMPC for time varying reference x^{ref}

At each sampling time t_n, $n = 0, 1, 2 \ldots$

1) Measure the state $x(n) \in X$ of the system.
2) Set $x_0 = x(n)$, solve the Optimal Control Problem:

$$\begin{aligned} &\text{minimize} && J_N(n, x_0, u(\cdot)) := \\ &&& \sum_{k=0}^{N-1} \ell(n+k, \hat{x}(k), u(k)), \\ &\text{with respect to} && u(\cdot) \in \mathbb{U}^N(x_0), \\ &\text{subject to} && \hat{x}(0) = x(n) = x_0, \\ &&& \hat{x}(k+1) = f(\hat{x}(k), u(k)). \end{aligned}$$

3) Get the feedback value $u^*(0) \in U$ and use it in the next sampling period.

control value u. This is described using the following process model:

$$x^+ = f(x, u), \tag{1}$$

where $f : X \times U \to X$ represents a nonlinear mapping. X and U are arbitrary metric spaces.

From the current state $x(n)$ (denoted as x_0), for any given control sequence $u(0), u(1), ..., u(N-1)$, the predicted state trajectory $\hat{x}(1), \hat{x}(2), ..., \hat{x}(N)$ can be constructed by iterating (1):

$$\hat{x}(k+1) = f(\hat{x}(k), u(k)), k = 0, ..., N-1. \tag{2}$$

with $\hat{x}(0) = x(n) = x_0$.

A cost function is formulated to quantify the discrepancy between the predicted and reference trajectories and penalizes the control effort. This function takes the form:

$$J_N(n, x_0, u(\cdot)) := \sum_{k=0}^{N-1} \ell(n+k, \hat{x}(k), u(k)), \tag{3}$$

where the stage cost ℓ is defined as:

$$\begin{aligned} \ell(n+k, \hat{x}(k), u(k)) = d_X(\hat{x}(k), x^{ref}(n+k)) \\ +\lambda \cdot d_U(u(k), u^{ref}(n+k)), \end{aligned} \tag{4}$$

with $\lambda > 0$ and $d_A(\cdot, \cdot)$ denoting a certain metric between two elements in the metric space A.

The finite-horizon Optimal Control Problem (OCP) is defined as the minimization of J_N. By solving the OCP through online optimization, a solution $u^*(0), u^*(1), ..., u^*(N-1)$ is obtained, which is one that enables the predicted state trajectory to optimally track the reference trajectory. Finally, the first element of the optimal control sequence is applied as the control input.

The above process is summarized as the NMPC algorithm 1, which can be concluded as a implicit policy, predicting optimal sequence via online optimization and executing only the first action to realize receding-horizon feedback control. In contrast, robotic IL learns a policy from observation-action pairs, which can be viewed as an explicit mapping. In what follows, this paper gradually integrates these two paradigms, adapting core principles from NMPC to the robotic IL task to construct our Viper methodology.

B. Injecting NMPC to Robotic IL

For clarity of method exposition, this work takes a canonical Pick-and-Place task as an example, performed by a single/dual-arm manipulator equipped with an end-effector. The following fundamental notations are adopted:

- Discrete time step: n.
- Action (the input for controlling): $u(n) \in \mathbb{R}^{1\times(J+E)}$. It comprises the control values for the robot arm's joints (dimension J) and its end-effector (dimension E).
- State: $x(n) = x_{img}(n) + x_{pro}(n)$. Here, $x_{pro}(n) \in \mathbb{R}^{1\times(J+E)}$ represents the proprioceptive state (similar to the action). The term $x_{img}(n) \in \mathbb{R}^{1\times(H\times W\times C)}$ represents the exteroceptive state, which is the external environmental perception captured as an RGB image from the system's camera.

In time-varying NMPC, the value of reference trajectory $x^{ref}(n)$ changes with each time step. A corresponding NMPC controller can be designed to enable the closed-loop system to "track" this evolving trajectory. To achieve "intelligent tracking" of expert trajectories, a simple, straightforward application of the NMPC framework is investigated as an initial approach, requiring considerations of the following three aspects:

(1) Construct a nonlinear process model with environmental perception. Normally, the process model can be described by an analytical formula derived from principles of dynamics. However, once perceptual images are incorporated into the state, the resulting process model becomes a complex nonlinear system, which is intractable to express analytically. A possible approach is to learn the process model f by training a neural network on a dataset of (x, u, x^+) tuples collected from demonstrations.

(2) Define the specific formulation of the J_N, especially d_X and d_U accroding to the (3) and (4):

$$\begin{aligned} d_X(\hat{x}(k), x^{ref}(n+k)) = \\ \omega_1 \times mean(||\hat{x}_{pro}(k) - x^{ref}_{pro}(n+k)||_1) + \\ \omega_2 \times mean(||\hat{x}_{\widetilde{img}}(\breve{k}) - x^{ref}_{\widetilde{img}}(n+\breve{k})||_2^2), \end{aligned} \tag{5}$$

$$\begin{aligned} d_U(u(k), u^{ref}(n+k)) = \\ mean(||u(k) - u^{ref}(n+k)||_1), \end{aligned} \tag{6}$$

where $k = 0, 1, ..., N-1$. The d_X is a weighted sum penalizing deviations in both proprioceptive states x_{pro} via the L1 norm and visual states $x_{\widetilde{img}}$ via the squared L2 norm. The visual state is a feature map extracted by a backbone network (e.g., ResNet-18). To reduce computational overhead, the visual penalty is applied sparsely. This is denoted by the index $\breve{k}$, which indicates that the term is computed only at specific intervals within the prediction horizon. For example, given a horizon $N = 50$, this term may be evaluated only for $k \in \{10, 20, 30, 40, 50\}$. The d_U applies an L1 penalty to the deviation from the reference action.

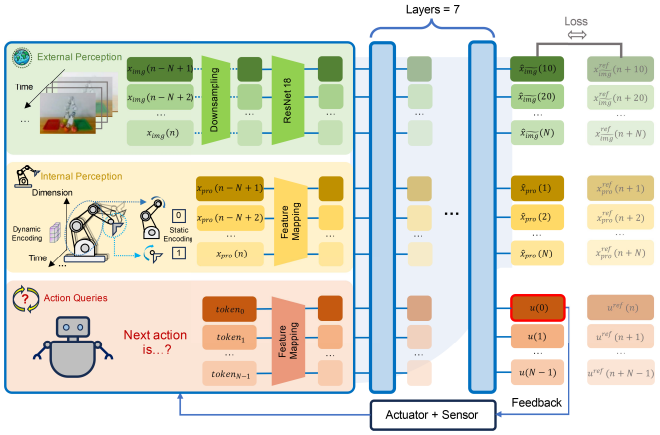

Fig. 2. **Overall architecture of Viper.** "n" denotes the current timestep. The mapping from inputs, covered by a light blue background, to the predicted state sequence reflects the model's understanding of the process dynamics. Within the multi-layer architecture, the predicted state and action tokens at each layer are progressively refined, "improving" or "converging" toward an optimal region. By the final layer, the output represents the optimal state trajectory and action sequence. And the first action is then applied to the Viper system.

(3) Determine the expected reference sequence $x^{ref}(n)$ for online optimization. While a weighted average of expert data might serve as a reference, such a fixed reference trajectory directly leads to rigid motion patterns, failing to adapt flexibly to dynamic real-world environments. Particularly in visual perception, the precomputed average visual reference rarely matches the current scene, resulting in an excessively high failure rate. Furthermore, the complexity of the learned process model makes online optimization computationally prohibitive for real-time applications, despite potential engineering optimizations.

The primary limitation of the aforementioned straightforward NMPC approach lies in the online solution of the OCP; specifically, it requires flexibly and appropriately defining the reference trajectory and rapidly completing the online optimization computation. Thus, the key challenge is to overcome these difficulties while preserving the core principles of NMPC.

Notably, the core of OCP is essentially seeking an optimal control sequence over a finite prediction horizon. In the context of robotic IL tasks, the notion of "optimality" is defined by the expert demonstration, which implicitly encodes human preferences. This insight motivates a paradigm shift: Rather than solving the OCP through online optimization, **we instead predict the optimal control sequence via an offline model learned from expert data.**

Through offline training, the proposed model learns to produce a strong approximation of the finite-horizon optimal sequence. This approach circumvents the need for an explicit reference trajectory and eliminates the significant computational overhead associated with online optimization. However, the choice of model architecture is not arbitrary: It must be carefully designed to reflect the intrinsic characteristics of the NMPC solution process. First, the iterative nature of online OCP solvers motivates the design of a multi-layer decoder to progressively refine the predicted sequence. Second, the reliance of NMPC on a process model to predict optimal state-action sequences necessitates an architecture capable of modeling the system dynamics and co-predicting serialized outputs for both states and actions. To satisfy these requirements, a multi-layer Transformer-Decoder is therefore proposed for the real-time prediction of optimal state and action sequences. This model, combined with the feedback, constitutes the Viper architecture.

C. Design of Viper Architecture

As illustrated in Fig. 2, the unified predictive model of Viper takes as input a sequence of N historical proprioceptive states, corresponding perceptual feature maps, and N action query tokens. It then outputs the predicted near-optimal future sequences of proprioceptive states, perceptual features, and control actions over the next N time steps. Based on the feedback, the first predicted action is executed as the control input, transitioning the Viper system into a new state. Repeating iteratively, this process forms a receding-horizon feedback control paradigm that is conceptually similar to NMPC.

Joint Prediction of Optimal States and Actions. In NMPC theory, the process model tightly couples states and control actions, and online optimization solves for both simultaneously. This insight motivates the use of the unified predictive model, whose multi-layer decoder structure iteratively optimizes both state and action representations via tightly coupled, cross-layer updates. As a result, the model produces consistent estimates of the optimal state-action sequence, ensuring the accuracy and stability of the control input selected by the feedback.

Soft Constraints via Static and Dynamic Encoding. Deriving admissible actions and reachable states requires rigorous analysis of robotic systems' physical properties and environmental interactions. However, the complexity of modern intelligent agents and real-world physics often poses a significant challenge in specifying explicit physical constraints for robotic tasks. To overcome this, the Viper architecture integrates a static-dynamic encoding mechanism within the predictive model. As shown in Fig. 2: Static encoding embeds human prior knowledge by structurally segmenting the proprioceptive vector. For example, joint dimensions belonging to the same robotic arm are assigned a shared encoding vector, while those associated with the same gripper share a distinct one. Dynamic encoding applies 2D convolutions across the time and proprioceptive dimensions to capture the temporal and inter-joint dependencies of system dynamics.

Mitigation of Compounding Errors. The proposed predictive model implicitly incorporates the underlying process model, which is informed by both external perception and internal proprioception. This hybrid sensing enables the model to account for dynamic changes in both environmental context and the robot's own embodiment, significantly enhancing the robustness of action prediction. Furthermore, by

combining receding-horizon prediction with feedback-based execution, the Viper system maintains long-term proactive foresight and cautiously selects the next action, reducing the accumulation of errors over time.

Loss Function. The unified predictive model of Viper is trained via a loss function structurally aligned with the cost function (3), (4). This loss penalizes the deviation of the predicted sequences of states and actions from their corresponding ground-truth references. The specific form of the loss is as follows:

$$Loss = J_N = \sum_{k=0}^{N-1} [d_X(\hat{x}(k), x^{ref}(n+k)) + \lambda d_U(u(k), u^{ref}(n+k))], \tag{7}$$

where d_X and d_U follow the definition accroding to (5), (6).

D. Stability Analysis

Since the Viper framework is theoretically grounded in NMPC, where stability analysis is paramount, we first briefly review the core principles of NMPC stability. Then the analysis will be extended to the proposed Viper architecture within the context of robotic IL tasks.

Intuitively, the "stability" of a closed-loop system means that for a time-varying reference trajectory $x^{ref}(n)$, there exists a neighborhood around it such that any perturbed system state $x(n)$ will progressively approach and eventually remain within this neighborhood during a finite time, tracking the reference trajectory closely thereafter.

Lyapunov functions provide a well-known tool for analyzing such stability. The core idea is to construct a designed time-varying "energy" function, $V(n, x)$, which is zero at the reference $x^{ref}(n)$. This function must satisfy: (1) its value increases as the state deviates further from the reference, and (2) its value decreases with each time step of the system's evolution. In summary, as time progresses, $V(n, x)$ gradually decreases, indicating that the $x(n)$ is converging towards $x^{ref}(n)$. According to established theorems [10], if a Lyapunov function which is positive definite, radially unbounded, and monotonically decreasing over time can be found within the state space containing the reference trajectory, then the closed-loop system is stable with respect to that trajectory.

In the finite-time NMPC, for a given initial state $x_0 \in \mathbb{X}$ and time $n \in \mathbb{N}_0$, the Optimal Value Function is defined as:

$$V_N(n, x_0) := \inf_{u(\cdot) \in \mathbb{U}^N(x_0)} J_N(n, x_0, u(\cdot)) = \sum_{k=0}^{N-1} \ell(n+k, \hat{x}(k), u(k)). \tag{8}$$

When this infimum is achieved by an optimal control sequence $u^*(\cdot) \in \mathbb{U}^N$, such that

$$V_N(n, x_0) = J_N(n, x_0, u^*(\cdot)), \tag{9}$$

the corresponding $\hat{x}^*(\cdot)$ is termed the optimal trajectory.

In fact, this optimal value function V_N can be regarded as a candidate for the desired Lyapunov function. To establish stability, the feedback must ensure that V_N decays by at least a fraction of the stage cost at each step [10]:

$$V_N(n, x_0) \geq \alpha \ell(n, \hat{x}(0), u^*(0)) + V_N(n+1, x^+), \tag{10}$$

where $\alpha \in (0, 1]$. This serves as our criterion for verifying closed-loop stability. In practice, the system is considered to have satisfactory practical stability if V_N demonstrates a monotonic decrease following a disturbance, even if it later settles into bounded oscillations within a small region.

In the Viper architecture, the explicit optimization of the NMPC's OCP is replaced by a learned model that directly predicts the optimal control sequence. By employing a loss function that is structurally identical to NMPC cost function J_N, the supervised learning process minimizes this loss until convergence. This ensures that for inputs within the training distribution, the well-trained model predicts a control sequence nearly identical to the one obtained via online optimization. For out-of-distribution inputs, the model's prediction is expected to be an optimal approximation. This learning-based prediction is not only faster but also exhibits superior intelligence and generalization in unstructured scenarios.

The well-established stability of finite-horizon NMPC, given its structural parallels to Viper, provides a qualitative argument for the stability of our approach. This claim is then addressed empirically in the experimental section, where a monotonic decrease of the Lyapunov function, V_N, is demonstrated.

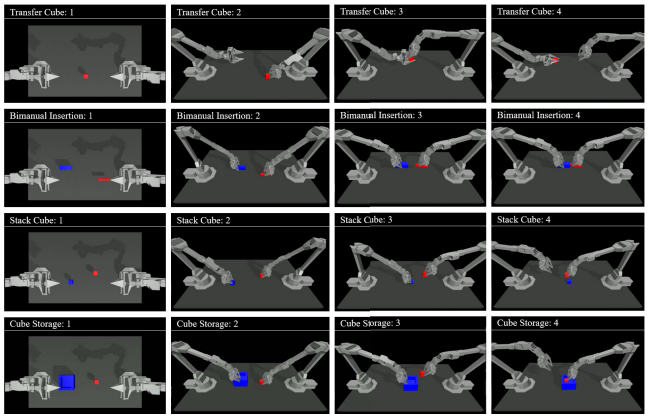

Fig. 3. **Overview of the simulation tasks. Transfer Cube:** The right arm grasps and lifts the red cube, then transfers it to the left arm through coordinated bimanual manipulation. **Bimanual Insertion:** The left arm acquires the blue socket, and the right arm grasps the red peg. The arms then align the peg and insert it into the socket. **Stack Cube:** The left arm places the blue cube at the center of the workspace, and the right arm stacks the red cube atop it. **Cube Storage:** The left arm moves the blue container to the center, and the right arm deposits the red cube inside it.

IV. Experiments

In this section, we perform a series of experiments aimed at addressing the following questions (Q1-Q5):

Q1: How does the performance of Viper compare with existing popular methods, particularly in terms of task success rate, robustness, and inference efficiency?

TABLE I

SIMULATION RESULTS

Policy	1. Transfer Cube			2. Bimanual Insertion			3. Stack Cube			4. Cube Storage		
	Success	Speed	Param	Success	Speed	Param	Success	Speed	Param	Success	Speed	Param
ACT (μ = temporal ensemble)	98%	**5.92ms**	83.90M	34%	**6.30ms**	83.90M	**94%**	**5.85ms**	83.90M	98%	**6.02ms**	83.90M
ACT ($\mu = 1$)	0%	5.92ms	83.90M	0%	6.30ms	83.90M	0%	5.85ms	83.90M	0%	6.02ms	83.90M
DP ($\mu = 1$)	98%	183.12ms	263.44M	50%	181.71ms	263.44M	86%	181.32ms	263.44M	68%	183.12ms	263.44M
Viper ($\mu = 1$)	**100%**	13.45ms	**34.30M**	**92%**	13.34ms	**34.30M**	92%	13.33ms	**34.30M**	**98%**	13.45ms	**34.30M**

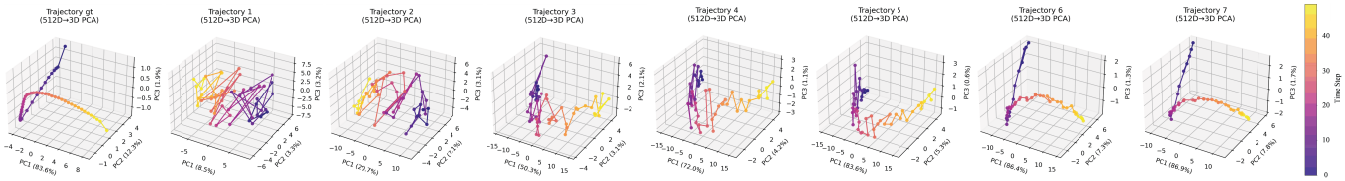

Fig. 4. **Visualization of each decoder's output.** The high-dimensional proprioceptive outputs from each decoder layer (Trajectory 1-7), along with their corresponding high-dimensional ground-truth mappings (Trajectory gt), are projected into a lower-dimensional space using Principal Component Analysis (PCA) and visualized accordingly. The axis labels indicate the proportion of variance retained by each principal component, reflecting the relative importance of each dimension. A color gradient from blue to yellow, applied to the data points, represents the temporal progression of the time steps.

Q2: To what extent does the unified predictive model design contribute to performance?

Q3: Can we provide intuitive visualizations of the unified predictive model's internal mechanisms and the resulting behavior of the Viper closed-loop architecture?

Q4: Can the stability of the Viper closed-loop architecture be empirically verified?

A. Simulation Experiments

Experimental Setup. Our experiments were conducted in the MuJoCo [42] simulation environment across four dual-arm manipulation tasks: Transfer Cube, Bimanual Insertion, Stack Cube, and Cube Storage, as illustrated in Fig. 3. Each task incorporated 200 expert demonstrations and featured varying levels of randomized initial conditions coupled with sparse reward structures to mimic real-world operational challenges. Aligned with ACT settings, the simulated workspace comprised two ViperX 300 robotic arms equipped with parallel-jaw grippers, positioned symmetrically on a tabletop. Visual inputs were captured through a top-mounted RGB camera system, providing full workspace observation for policy training and evaluation.

Metrics and Training Setup. We benchmarked Viper against two popular baselines: ACT and DP. The ACT model was trained for 2,500 epochs with a fixed action chunking size of 50. The DP model underwent 200,000 steps with a horizon length of 64, utilizing 100 iterations during the diffusion process. Our Viper model was trained for 2,500 epochs with a prediction horizon of 50. All three models used the same feedback, executing only the first action after each prediction, denoted as $\mu = 1$. The methods were systematically evaluated through 50 randomized trials per task, with success rate (denoted as "Success") computed as the average over these trials. Inference speed ("Speed") was measured on an RTX 4090 D GPU by averaging the time required to generate 50 consecutive action predictions over 50 evaluation runs. Model scale ("Param") was quantified using parameter counts derived from identical PyTorch implementation frameworks.

TABLE II

ABLATION EXPERIMENT RESULTS

Basic Framework	Unified Predictions	Sequential Predictions	Soft Constraints	Success Rate
H.Len=1 States&Actions	✓	✗	✗	2% | 0%
H.Len=50 Actions only	✗	✓	✗	74% | 88%
H.Len=50 States&Actions	✓	✓	✗	80% | 94%
H.Len=50 States&Actions	✓	✓	✓	**92%** | **98%**

"H.Len" refers to the horizon length of predictions. "States&Actions" indicates that the model predicts states and actions simultaneously.

Results and Analysis. As demonstrated in Table I, Viper achieved highly competitive task success rates across all four simulation tasks while maintaining the smallest model size and comparable inference speed to ACT, indicating superior overall performance and supporting **Q1**. To further validate the effectiveness of each component in the unified predictive model of Viper, we conducted ablation studies on the Bimanual Insertion and Cube Storage tasks, following the same training and testing protocols as previously described. The results are presented in Table II. "Unified Predictions" and "Sequential Predictions" represent whether the model simultaneously predicts states and actions, and whether it predicts future sequence rather than just a single timestep. "Soft Constraints" refers to the static-dynamic encoding method described in Section III-B. The results reveal that each component progressively enhances model performance, answering **Q2**.

To address the first part of **Q3**, we visualized the high-

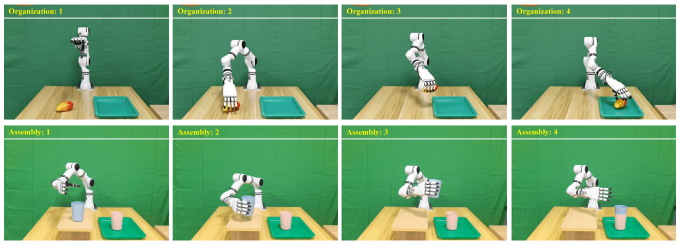

Fig. 5. **Overview of the real-world tasks. Organization:** The manipulator grasps scattered fruits from a tabletop and deposits them into a tray. **Assembly:** The manipulator grasps a blue cup and performs a precision stack onto a red cup.

TABLE III

STANDARD REAL-WORLD EXPERIMENT TESTING RESULTS

Policy	1. Organization			2. Assembly		
	Success	Time	Param	Success	Time	Param
ACT ($\mu = 5$)	0%	-	83.92M	0%	-	83.92M
ACT ($\mu = 10$)	45%	34.67s	83.92M	0%	-	83.92M
ACT ($\mu = 20$)	75%	42.94s	83.92M	50%	24.40s	83.92M
DP ($\mu = 5$)	10%	43.60s	263.32M	0%	-	263.32M
Viper ($\mu = 5$)	**75%**	**29.23s**	**34.23M**	**85%**	**23.88s**	**34.30M**

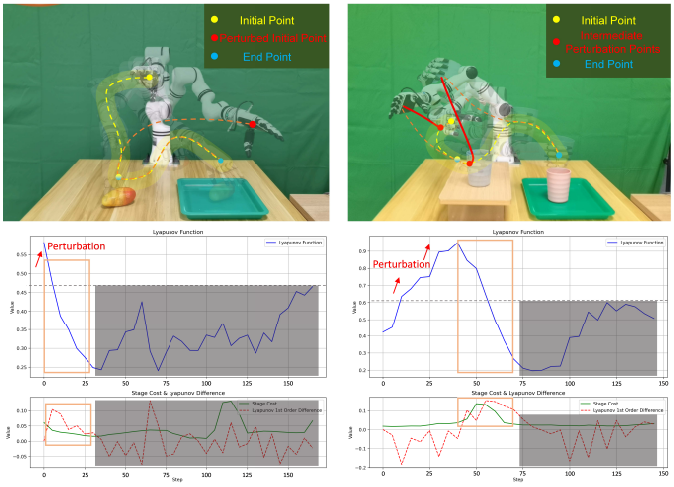

Fig. 6. **Stability Analysis in the Presence of Perturbations.** For each task, two plots are provided: The top plot illustrates the evolution of the Lyapunov function over time under perturbations, while the bottom plot compares its first-order difference with the stage cost. Given the $\mu = 5$, the Viper system executes initial 5 actions of each predicted optimal sequence, after which the corresponding data is computed and plotted.

dimensional output from each layer of the unified predictive model, using a randomly selected segment of proprioceptive state data from the Stack Cube task as input (Fig. 4). The visualization shows that the high-dimensional trajectory patterns, which are initially noisy, become progressively more structured and ordered with each successive layer. The final output trajectory pattern closely matches the pattern of the ground-truth proprioceptive states. This visualization indecates that the unified predictive model indeed performs an iterative refinement, akin to online optimization, to generate its prediction of the optimal sequence.

B. Evaluation on Real-World

Experimental Setup. The experimental platform comprises a Realman RM65-B robotic arm, an INSPIRE five-fingered dexterous hand, a RealSense D435i camera, and a visual teleoperation data acquisition system. To collect demonstration data for the real-world robotic IL tasks (Organization and Assembly, see Fig. 5), operators manipulated the robotic system through headset-enabled teleoperation with a Vision Pro. For each task, 150 demonstrations were collected.

Implement Details. We conducted real-world comparative evaluations of Viper, DP and ACT. All three methods processed RGB visual observations and proprioceptive inputs. Viper and ACT were trained for 4000 epochs, while DP was trained for 400,000 steps; other training parameters remained consistent with the simulation settings described previously. To ensure practical applicability, all three methods employed an identical feedback for action execution, implementing the first five steps from the predicted action sequence. The performance for each task was measured over 20 trials. The "Success" and "Param" metrics follow the same definitions as in the simulation study. The "Time" metric measures the average task completion time on the physical robot. Beyond this standard real-world testing, specific perturbations are introduced to the Viper system to examine the behavior of the Lyapunov function during task execution and empirically validate the system's stability. The performance of Viper under a broader range of disturbances is presented in the supplementary video.

Resultes and Analysis. As shown in table III, Viper demonstrates comparable or superior performance to ACT and DP in real-world tasks concerning success rate, task completion time and model scale, thereby affirming **Q1**. It is noteworthy that despite a slightly slower inference time, Viper completes tasks faster than ACT. This suggests that the Viper system is more effective, producing fewer redundant action steps compared to the trajectories from ACT. For the practical stability analysis of Viper system, the demonstration data are weighted and length-adjusted to obtain reference trajectories, with the expectation that the predicted sequences from Viper will be optimal. The data are then substituted into Equations (9), (3), (4), (5) and (6) for real-time computation, observing the evolution of the Lyapunov function V_N, its first-order difference, and the stage cost ℓ. As illustrated in Fig. 6, for both tasks, when subjected to different perturbation, V_N initially exhibits a rapid, monotonic decrease, followed by bounded oscillations within a region. In the interval of monotonic decrease, the condition (10) is satisfied. This empirically demonstrates the practical stability of the Viper system, addressing **Q4** and the latter part of **Q3**.

V. CONCLUSION

This work introduces Viper, a novel visuomotor policy framework that integrates the principles of NMPC with robotic IL. It replaces the explicit online optimization module of NMPC with a unified predictive model, which, in conjunction with a corresponding feedback, enables the efficient

execution of imitation learning tasks. Furthermore, Viper reframes IL as an intelligent tracking problem centered on expert trajectories. This provides a new lens through which to assess the reliability of learned policies, leveraging the Lyapunov stability analysis. Experimental results demonstrate that Viper framework offers significant gains in interpretability and task performance. Future work will focus on validating the effectiveness of the Viper method with 3D visual inputs.

REFERENCES

[1] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, "A survey of imitation learning: Algorithms, recent developments, and challenges," *IEEE Transactions on Cybernetics*, 2024.

[2] S. Mahmoudi, A. Davar, P. Sohrabipour, R. B. Bist, *et al.*, "Leveraging imitation learning in agricultural robotics: a comprehensive survey and comparative analysis," *Frontiers in Robotics and AI*, vol. 11, p. 1441312, 2024.

[3] Y. Hu, F. J. Abu-Dakka, F. Chen, X. Luo, *et al.*, "Fusion dynamical systems with machine learning in imitation learning: A comprehensive overview," *Information Fusion*, p. 102379, 2024.

[4] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 955–22 968, 2022.

[5] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[6] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, *et al.*, "Behavior generation with latent actions," *arXiv preprint arXiv:2403.03181*, 2024.

[7] C. Chi, Z. Xu, S. Feng, E. Cousineau, *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[8] M. Reuss, Ö. E. Yağmurlu, F. Wenzel, and R. Lioutikov, "Multimodal diffusion transformer: Learning versatile behavior from multimodal goals," in *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024.

[9] Y. Wang, Y. Zhang, M. Huo, R. Tian, *et al.*, "Sparse diffusion policy: A sparse, reusable, and flexible policy for robot learning," *arXiv preprint arXiv:2407.01531*, 2024.

[10] L. Grüne, J. Pannek, L. Grüne, and J. Pannek, *Nonlinear model predictive control*. Springer, 2017.

[11] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[12] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.

[13] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Machine Intelligence 15*, 1995, pp. 103–129.

[14] T. Zhang, Z. McCarthy, O. Jow, D. Lee, *et al.*, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5628–5635.

[15] S. Kareer, D. Patel, R. Punamiya, P. Mathur, *et al.*, "Egomimic: Scaling imitation learning via egocentric video," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 13 226–13 233.

[16] E. Johns, "Coarse-to-fine imitation learning: Robot manipulation from a single demonstration," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4613–4619.

[17] J. Carvalho, A. T. Le, P. Kicki, D. Koert, *et al.*, "Motion planning diffusion: Learning and adapting robot motion planning with diffusion models," *IEEE Transactions on Robotics*, 2025.

[18] P. Wu, Y. Shentu, Q. Liao, D. Jin, *et al.*, "Robocopilot: Human-in-the-loop interactive imitation learning for robot manipulation," *arXiv preprint arXiv:2503.07771*, 2025.

[19] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in Neural Information Processing Systems*, vol. 1, 1988.

[20] Z. Chen and X. Huang, "End-to-end learning for lane keeping of self-driving cars," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1856–1860.

[21] J. Aoki, F. Sasaki, K. Matsumoto, R. Yamashina, *et al.*, "Environmental and behavioral imitation for autonomous navigation," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 7779–7786.

[22] S. Belkhale, Y. Cui, and D. Sadigh, "Data quality in imitation learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 80 375–80 395, 2023.

[23] K. Doshi, M. Bagatella, and S. Coros, "Problem space transformations for generalisation in behavioural cloning," *arXiv preprint arXiv:2411.04056*, 2024.

[24] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[25] P. Florence, L. Manuelli, and R. Tedrake, "Self-supervised correspondence in visuomotor policy learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 492–499, 2019.

[26] L. Ke, J. Wang, T. Bhattacharjee, B. Boots, *et al.*, "Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6185–6191.

[27] A. Zeng, P. Florence, J. Tompson, S. Welker, *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2021, pp. 726–747.

[28] Y. Avigal, L. Berscheid, T. Asfour, T. Kröger, *et al.*, "Speedfolding: Learning efficient bimanual folding of garments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1–8.

[29] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, *et al.*, "Implicit behavioral cloning," in *Conference on Robot Learning*. PMLR, 2022, pp. 158–168.

[30] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-conditioned imitation learning using score-based diffusion policies," *arXiv preprint arXiv:2304.02532*, 2023.

[31] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.

[32] M. Grotz, M. Shridhar, Y.-W. Chao, T. Asfour, *et al.*, "Peract2: Benchmarking and learning for robotic bimanual manipulation tasks," in *CoRL 2024 Workshop on Whole-body Control and Bimanual Manipulation: Applications in Humanoids and Beyond*, 2024.

[33] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[34] L. S. Pontryagin, *Mathematical theory of optimal processes*. Routledge, 2018.

[35] H. Deng and T. Ohtsuka, "A highly parallelizable newton-type method for nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 349–355, 2018.

[36] W. Jallet, E. Dantec, E. Arlaud, N. Mansard, *et al.*, "Parallel and proximal constrained linear-quadratic methods for real-time nonlinear mpc," in *Robotics: Science and Systems*, 2024.

[37] M. Karimshoushtari, C. Novara, and F. Tango, "How imitation learning and human factors can be combined in a model predictive control algorithm for adaptive motion planning and control," *Sensors*, vol. 21, no. 12, p. 4012, 2021.

[38] D. H. Shim, H. J. Kim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots," in *42nd IEEE International Conference on Decision and Control*, vol. 4. IEEE, 2003, pp. 3621–3626.

[39] S. Mamedov, R. Reiter, S. M. B. Azad, R. Viljoen, *et al.*, "Safe imitation learning of nonlinear model predictive control for flexible robots," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 3613–3619.

[40] M. Novak and T. Dragicevic, "Supervised imitation learning of finite-set model predictive control systems for power electronics," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 2, pp. 1717–1723, 2020.

[41] P. I. Gómez, M. E. L. Gajardo, N. Mijatovic, and T. Dragičević, "Enhanced imitation learning of model predictive control through importance weighting," *IEEE Transactions on Industrial Electronics*, 2024.

[42] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 5026–5033.