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

Prompt-to-State Stable Vision-Language MPC for Approximated Neural Network Dynamics: A Case Study on Soft Robot Control

Emanuele Nicotra[1], James Davies[1], Kefan Zhu[1], Bibhu Sharma[1], Adrienne Ji[1], Phuoc Thien Phan [1], Hung Manh La [2], Nigel H. Lovell[1], *IEEE Fellow*, Thanh Nho Do[1], *IEEE Member*,

***Abstract*— The integration of large-scale foundation models in control loops proven to be effective for executing complex tasks from natural language inputs. However, ensuring stability and real-time performance remains a significant challenge when such models are used, especially for systems with hard-to-model dynamics. In this paper we introduce the concept of Prompt-to-State Stability (PSS) and we present the Prompt-to-State Stable Vision-Language Model Predictive Control (PSS-VLMPC), a novel framework that integrates a VLM with a robust MPC. We use the VLM to interpret user commands and visual feedback, translating them into parameters for the MPC that controls the system. The system's dynamics are entirely learned by a neural network, and approximated for real-time performance of the MPC. Starting from the prediction error bound we provide rigorous stability guarantees for the closed-loop system, provided the environment dynamics do not exceed the VLM update rate. The effectiveness of the PSS-VLMPC is validated through simulations and real-world experiments on a soft continuum robot, demonstrating its capability to execute tasks from natural language inputs.**

Keywords: Robotics, Vision Language Model, Model Predictive Control, Neural Network Dynamics, Prompt-to-State Stability

I. Introduction

The seamless interaction between humans and robots is a cornerstone of next-generation robotics, with natural language serving as the most intuitive communication medium. Vision Language Models (VLMs) and their embodied counterparts, Vision Language Action (VLA) models, have recently emerged as a novel technology for enabling robots to understand and execute high-level tasks from multimodal instructions. Recent works connect vision and language to control [1], [2], focusing either on open-ended tasks [3], [4] or foundational capabilities [5]–[7]. These models have demonstrated remarkable capabilities, from complex object manipulation [3] to autonomous navigation [8].

However, deploying these powerful foundation models directly into closed-loop control presents significant challenges: their "black-box" nature and probabilistic outputs make formal guarantees on safety and stability difficult, despite being crucial for physical human-robot interaction [9]–[12]. An instruction misinterpreted or an action generated without regard to the system's limits could lead to catastrophic failures, as shown by recent red-teaming and adversarial studies where VLA/LLM-robot systems were jailbroken or perturbed into unsafe, unintended behaviors [13]–[17]. To bridge the gap between high-level semantic instruction and low-level safety-critical control, structured approaches that layer formal safety tools beneath VLM/VLA perception and tasking are increasingly advocated [18]–[20].

Model Predictive Control (MPC) offers a compelling framework for this integration [8], [21]. By optimizing control actions over a predictive horizon while explicitly respecting state and input constraints, MPC provides a principled way to translate high-level goals into safe, dynamically feasible control inputs, provided a state function of the system is known.

Several recent works propose tighter integration between VLMs (or VLAs) and MPC or other controllers. For example, VLMPC frameworks fuse visual and language goals with MPC for manipulation tasks [21], and VLM-guided MPC has been explored in autonomous driving scenarios [8].

A further challenge arises when applying MPC to systems with complex, nonlinear dynamics, where deriving first-principles models is difficult. Neural networks offer a powerful learning-based alternative; however, while their forward inference is fast [22], embedding them as the prediction model inside the MPC optimization loop at every step creates a significant computational burden. Simplified or approximated models (e.g., via Taylor expansions) ease computation [23], but introduce truncation errors that compound over the prediction horizon, which must be formally bounded to guarantee closed-loop stability.

This paper addresses these challenges by proposing a novel framework that synergizes the strengths of VLMs and robust control theory. Our main contributions are:

1) The definition of Prompt-to-State Stability (PSS), which provides a theoretical guarantee for the stability of the closed-loop system under arbitrary user prompts, by leveraging the underlying Input-to-State Stability (ISS) of the low-level system.
2) A novel Prompt-to-State Stable Vision-Language MPC (PSS-VLMPC) framework that translates natural language commands into MPC parameters, enabling high-level task execution for any system with bounded state prediction error, while ensuring PSS by appropriately scaling the terminal cost weight based on the model approximation error.
3) Validation of the proposed framework through both simulation and real-world experiments on a soft continuum robot, demonstrating its effectiveness in executing tasks specified via natural language.

The rest of the paper is organized as follows. Section II.A introduces the MPC problem without terminal constraints for

[1] The authors are with UNSW Sydney, Kensington Campus, NSW 2052, Australia (e-mail: e.nicotra@unsw.edu.au, j.j.davies@student.unsw.edu.au, kefan.zhu@student.unsw.edu.au, bibhu.sharma@unsw.edu.au, adrienne.ji@unsw.edu.au, phuocthien.phan@unsw.edu.au, n.lovell@unsw.edu.au, tn.do@unsw.edu.au)

[2] The author is with Computer Science and Engineering, University of Nevada, USA (e-mail: hla@unr.edu)

a system with learned neural network dynamics. Section II.B presents the sufficient conditions for the Input-to-State Stability of systems with approximated neural network dynamics, giving the explicit computation of the terminal weight that guarantees ISS. Section II.C introduces the proposed PSS-VLMPC framework after the defintion of Prompt-to-State Stability. Section III presents simulations and experimental results validating the PSS-VLMPC framework on a soft continuum robot. Finally, we conclude the paper in Section IV with a discussion of future research directions.

II. Methods

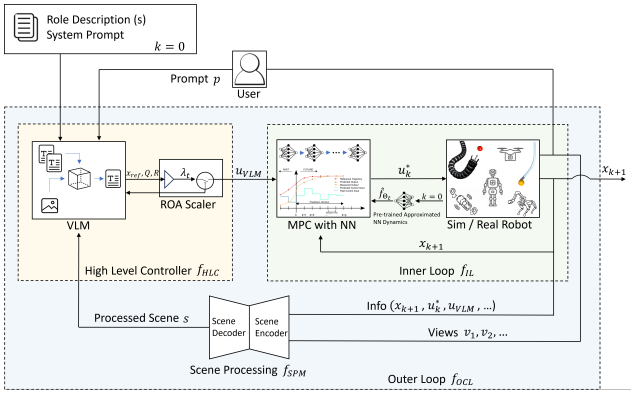

Fig. 1: High-level representation of the PSS-VLMPC framework. The Outer Loop (OL) utilizes a Vision-Language Model to translate user prompts and visual scenes into MPC parameters. The Inner Loop (IL) subsequently acts as the plant for the OL, solving the MPC problem using a Taylor expansion of the learned neural network dynamics to achieve computational efficiency while guaranteeing Input-to-State Stability.

A. Problem Formulation

Consider a system with state $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ and control input $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^m$. The system dynamics are described by a state function $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$. Where $\mathcal{X}$ is the state constraint set, and $\mathcal{U}$ is the control input constraint set. The discretized system dynamics can be expressed as

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{1}$$

where $\mathbf{x}_k$ is the state at time step k and $\mathbf{u}_k$ is the control input at time step k. The goal is to find a sequence of control inputs that minimizes a cost function $J(\mathbf{x}, \mathbf{u})$ while satisfying the system dynamics and constraints. The general MPC problem P_N without terminal constraints is formulated as follows for the system in (1):

$$\min_{\mathbf{u}} \quad J(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^{N-1} \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) \tag{2}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{3}$$

$$\mathbf{x}_0 = \mathbf{x}_{\text{init}} \tag{4}$$

$$\mathbf{x}_k \in \mathcal{X}, \quad \mathbf{u}_k \in \mathcal{U} \quad \forall k = 0, \dots, N-1 \tag{5}$$

where $\mathbf{x}_k$ is the state at time step k, $\mathbf{u}_k$ is the control input, $\mathcal{L}(\cdot)$ is the cost. The optimization is performed over the prediction horizon N. The closed-loop system, obtained by applying the sequence of the first optimal control inputs, is Input-to-State Stable (ISS) when the cost is chosen as the sum of a positive definite stage cost and a terminal cost [24], i.e.

$$J(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + \lambda V(\mathbf{x}_N) \tag{6}$$

where $V(\mathbf{x}_N)$ is a control Lyapunov function at the terminal state $\mathbf{x}_N$ and $\lambda > 0$ is a weighting factor for the terminal cost. In particular λ defines a Region Of Attraction (ROA) $\mathcal{X}_N(\lambda)$ for the closed-loop system, i.e. the set of initial states $\mathbf{x}_0$ such that the closed-loop system converges to the origin for all $\mathbf{x}_0 \in \mathcal{X}_N(\lambda)$. Hence given an initial feasible state it's possible to find the value of λ such that the closed-loop system is ISS.

When the system dynamics are hard to model analytically, the state function is often approximated either partially or entirely by a neural network (NN) $f_\Theta(\mathbf{x}, \mathbf{u})$:

$$f(\mathbf{x}, \mathbf{u}) \approx f_\Theta(\mathbf{x}, \mathbf{u}) \tag{7}$$

where Θ are the parameters of the NN.

In order to solve the MPC problem, it's necessary to call the state function several times inside the optimization loop, which can be computationally expensive for the case of large neural networks. To address this problem, we approximate the neural network dynamics with its t-th Taylor expansion $\hat{f}_{\Theta_t}(\cdot)$ around the current state $\mathbf{x}_i$ and control input $\mathbf{u}_i$. Differently from [23], in our case $f(\cdot)$ predicts directly the next state $\mathbf{x}_{k+1}$, then no analytical model and no subsequent discrete integration are needed:

$$f_\Theta(\mathbf{x}, \mathbf{u}) = \hat{f}_{\Theta_t}(\mathbf{x}, \mathbf{u})|_{\mathbf{x}_i, \mathbf{u}_i} + \mathcal{O}(\|\mathbf{x} - \mathbf{x}_i\|^{t+1} + \|\mathbf{u} - \mathbf{u}_i\|^{t+1}) \tag{8}$$

For example the second order approximation is given by

$$\begin{aligned} f_\Theta(\mathbf{x}, \mathbf{u}) = f_\Theta(\mathbf{x}_i, \mathbf{u}_i) + \mathbf{J}_\Theta^i \begin{bmatrix} \mathbf{x} - \mathbf{x}_i \\ \mathbf{u} - \mathbf{u}_i \end{bmatrix} + \\ \frac{1}{2} \begin{bmatrix} \mathbf{x} - \mathbf{x}_i \\ \mathbf{u} - \mathbf{u}_i \end{bmatrix}^\top \mathbf{H}_\Theta^i \begin{bmatrix} \mathbf{x} - \mathbf{x}_i \\ \mathbf{u} - \mathbf{u}_i \end{bmatrix} + \\ \mathcal{O}(\|\mathbf{x} - \mathbf{x}_i\|^3 + \|\mathbf{u} - \mathbf{u}_i\|^3) \end{aligned} \tag{9}$$

where $\mathbf{J}_\Theta^i = \frac{\partial f_\Theta}{\partial \mathbf{x}, \mathbf{u}}|_{\mathbf{x}_i, \mathbf{u}_i}$ and $\mathbf{H}_\Theta^i = \frac{\partial^2 f_\Theta}{\partial \mathbf{x}, \mathbf{u}^2}|_{\mathbf{x}_i, \mathbf{u}_i}$ are the Jacobian and the Hessian matrices of the neural network dynamics with respect to the state and control input, evaluated at $\mathbf{x}_i, \mathbf{u}_i$.

The *prediction error* is defined as the difference between the true system dynamics and the NN dynamics:

$$\mathbf{e} = f(\mathbf{x}, \mathbf{u}) - f_\Theta(\mathbf{x}, \mathbf{u}) \tag{10}$$

For any pair of state and control input $\mathbf{x}_i$ and $\mathbf{u}_i$, the *approximation error* is defined as the difference between the NN dynamics and the t-th order approximation of the NN dynamics, while the *approximated prediction error* is defined as the difference between the true system dynamics and the t-th order approximation of the NN dynamics, respectively:

$$\mathbf{e}_{\Theta_t} = f_\Theta(\mathbf{x}, \mathbf{u}) - \hat{f}_{\Theta_t}(\mathbf{x}, \mathbf{u})|_{\mathbf{x}_i, \mathbf{u}_i} \tag{11}$$

$$\mathbf{e_t} = f(\mathbf{x}, \mathbf{u}) - \hat{f}_{\Theta_t}(\mathbf{x}, \mathbf{u})|_{\mathbf{x}_i, \mathbf{u}_i} \tag{12}$$

The previous definitions imply that the approximated prediction error is the sum of the prediction error and the approximation error of the NN dynamics $\mathbf{e_t} = \mathbf{e} + \mathbf{e}_{\Theta_t}$

In conclusion, including the NN approximation of the system dynamics and the terminal cost, the MPC problem in (2) can be reformulated as follows

$$\min_{\mathbf{u}} \quad J(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + \lambda V(\mathbf{x}_N) \tag{13}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \hat{f}_{\Theta_t}(\mathbf{x}_k, \mathbf{u}_k)|_{\mathbf{x}_i, \mathbf{u}_i} \tag{14}$$

$$\mathbf{x}_0 = \mathbf{x}_{\text{init}} \tag{15}$$

$$\mathbf{x}_k \in \mathcal{X}, \quad \mathbf{u}_k \in \mathcal{U} \quad \forall k = 0, \dots, N-1 \tag{16}$$

Which is the formulation adopted in this work. In particular, we assume the following

Assumption 1 (stage cost). The stage cost is chosen such that

$$\ell(\mathbf{x}_k, \mathbf{u}_k) = (\mathbf{x}_k - \mathbf{x}_{ref})^T Q (\mathbf{x}_k - \mathbf{x}_{ref}) + (\mathbf{u}_k - \mathbf{u}_{ref})^T R (\mathbf{u}_k - \mathbf{u}_{ref}) \tag{17}$$

where $\mathbf{x}_{ref}$ and $\mathbf{u}_{ref}$ are the reference state and control input, respectively and Q and R are positive definite diagonal matrices.

Assumption 2 (terminal cost). The terminal cost is chosen as a quadratic function of the state by solving the LQR problem for the linearized system, i.e. using $\hat{f}_{\Theta_t}(\cdot)$ with $t = 1$

$$V(\mathbf{x}_N) = (\mathbf{x}_k - \mathbf{x}_{ref})^T P (\mathbf{x}_k - \mathbf{x}_{ref}) \tag{18}$$

where P is the solution of the discrete Riccati equation.

Assumption 3 (NN error bound). There exists a constant that bounds the prediction error 10, i.e. $\exists \mu < \infty$ such that $\forall k \ \|\mathbf{e}_k\| \leq \mu$

The closed-loop control law is given by the series of first control inputs of the optimal sequence at every time instant, i.e., $\mathbf{u}^* = \{\mathbf{u}_0^*, \mathbf{u}_1^*, ...\}$ and the resulting closed-loop system is given by

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k^*) \tag{19}$$

B. Input-to-State Stability

As in [25] we distinguish between the nominal system that uses the true dynamics and the uncertain system that uses the neural network dynamics.

Theorem 1 (Inner Loop Stability). Consider the MPC problem 13. If Assumptions 1-2-3 hold and the approximated state function (7) is uniformly continuous $\forall \mathbf{x}_k \in \mathcal{X}_N(\lambda)$, then there exists a terminal weight $\lambda_t \geq 1$ that defines a ROA $\Omega_t \subseteq \mathcal{X}_N$ such that $\forall \mathbf{x}_0 \in \Omega_t(\lambda_t)$, the closed-loop system (19) is ISS with respect to the optimal control input $\mathbf{u}^*$ and the approximated prediction error $\mathbf{e}_t$.

Proof: Consider the Taylor expansion (8) of the neural network dynamics $f_\Theta(\cdot)$. The approximation error is bounded by a constant $\epsilon_t > 0$

$$\|\mathbf{e}_{\Theta_t}\| = \mathcal{O}(\|\mathbf{x} - \mathbf{x}_i\|^{t+1} + \|\mathbf{u} - \mathbf{u}_i\|^{t+1}) \leq \epsilon_t \tag{20}$$

By assumption the prediction error is bounded by a constant $\mu < \infty$, i.e. $\|\mathbf{e}\| \leq \mu$. Hence, there exists a constant $\mu_t > 0$ such that the approximated prediction error is bounded as

$$\|\mathbf{e}_t\| = \|\mathbf{e}\| + \|\mathbf{e}_{\Theta_t}\| \leq \epsilon_t + \mu \leq \mu_t \tag{21}$$

Given the choice of the stage and terminal costs, the proof of the input-to-state stability follows from *Theorem 2* in [25] by substituting the approximated prediction error $\mathbf{e}_t$ in place of the prediction error $\mathbf{e}$.

For the case of the nominal closed-loop system, i.e. when the approximated prediction error $\mathbf{e}_t$ is zero, the value (*cf. Theorem 3* [24]) of the stabilizing terminal weight λ is

$$\lambda = \frac{L_N - Nd}{(1-\rho)\alpha} \tag{22}$$

where L_N is the sum of the stage cost along the trajectory; $d > 0$ is a constant that characterizes the set of states outside the terminal region, i.e. such that $\ell(\mathbf{x}, \mathbf{u}) > d \ \forall \mathbf{u} \in \mathcal{U}$ and $\forall \mathbf{x} \notin \mathcal{X}_N$; $\alpha > 0$ is a constant that bounds the terminal cost $V(\mathbf{x}) \leq \alpha$; $\rho \in [0, 1)$ is a constant that characterizes the ROA defined as $\{\mathbf{x} : V(\mathbf{x}) \leq \rho\alpha\}$. In order to apply the same results to the case of the uncertain system, we need to define the new constants that depend on the approximated state function $\hat{f}_{\Theta_t}(\cdot)$ and the approximated prediction error $\mathbf{e}_t$.

Given the uniform continuity of the state function, stage cost and terminal cost, there exist three class $\mathcal{K}_\infty$ functions σ_f, σ_ℓ and σ_V such that $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ and $\forall \mathbf{u} \in \mathcal{U}$ the following holds (*cf. A.1, Lemma 1.* [26]):

$$\|f(\mathbf{x}_1, \mathbf{u}) - f(\mathbf{x}_2, \mathbf{u})\| \leq \sigma_f(\|\mathbf{x}_1 - \mathbf{x}_2\|) \tag{23}$$

$$\|\ell(\mathbf{x}_1, \mathbf{u}) - \ell(\mathbf{x}_2, \mathbf{u})\| \leq \sigma_\ell(\|\mathbf{x}_1 - \mathbf{x}_2\|) \tag{24}$$

$$\|V(\mathbf{x}_1) - V(\mathbf{x}_2)\| \leq \sigma_V(\|\mathbf{x}_1 - \mathbf{x}_2)\| \tag{25}$$

It follows that it's possible to construct the following uniform continuity function [26] as the composition

$$\sigma_f^j(\cdot) = \sigma_f^0 \circ \sigma_f^1 \circ ... \circ \sigma_f^j(\cdot) \tag{26}$$

that bounds the difference between the evolution of the system starting from the states $\mathbf{x}_1$ and $\mathbf{x}_2$ up to the step j for the same sequence of control inputs (*cf. A.1, Lemma 2.* [26]). Since $\mathbf{x}_1$ $\mathbf{x}_2$ are two arbitrary states, we can choose $\mathbf{x}_1 = f(\mathbf{x}, \mathbf{u})$ and $\mathbf{x}_2 = \hat{f}_{\Theta_t}(\mathbf{x}, \mathbf{u})|_{\mathbf{x}_i, \mathbf{u}_i}$, so that $\mathbf{x}_1 - \mathbf{x}_2 = \mathbf{e}_t$. Since the approximated prediction error is bounded by μ_t (20), then

$$\sigma_f(\|\mathbf{e}_t\|) \leq \sigma_f(\mu_t) \tag{27}$$

Then in the worst case the sum of stage costs along the trajectory is bounded by

$$\sum_{i=0}^{N} \ell(\hat{\mathbf{x}}_i, \mathbf{u}_i) \leq L_N + \Delta L_N(\mu_t) \tag{28}$$

where $\Delta L_N(\mu_t)$ is the additional cost due to the approximated prediction error defined by the uniform continuity function (26) as

$$\Delta L_N(\mu_t) := \sum_{i=0}^{N-1} \sigma_\ell(\sigma_f^i(\mu_t)) \tag{29}$$

Similarly, for the terminal cost we can define

$$\Delta V_N(\mu_t) := \sigma_V(\sigma_f^N(\mu_t)) \tag{30}$$

which implies that the terminal cost is bounded by

$$V(\hat{\mathbf{x}}_N) \leq \alpha + \Delta V_N(\mu_t) \tag{31}$$

Define the new constants that depend on the approximated prediction error bound $\hat{L}_N := L_N + \Delta L_N(\mu_t)$, $\Delta\rho := \Delta V_N(\mu_t)/\alpha$ and $\hat{\rho} := \rho + \Delta\rho$, then following the same steps in the proof of *Theorem 3* in [24] the terminal weight can be rewritten as

$$\lambda_t = \frac{\hat{L}_N - Nd}{(1-\hat{\rho})\alpha} \tag{32}$$

which is the value of the terminal weight that defines a ROA $\Omega_t(\lambda_t)$ such that $\forall \mathbf{x}_0 \in \Omega_t(\lambda_t)$, the closed-loop system (19) is ISS with respect to the optimal control input $\mathbf{u}^*$ and the approximated prediction error $\mathbf{e}_t$, thus concluding the proof. ■

Remark 1: Feasibility Condition. The denominator (32) must be positive, i.e. $(1-\hat{\rho})\alpha > 0$, which is equivalent to $(1-\rho)\alpha < \Delta V_N(\mu_t)$. If this check fails, no finite value of λ_t can guarantee the ISS stability of the closed-loop system. Intuitively the model error can move the terminal state outside the tolerated margin so much that weighting cannot recover it.

Remark 2. Conservatism. The value of the terminal weight can be reduced both by computing online the value of the sum of stage costs along the trajectory $\hat{L}_N$ and with an iterative computation as illustrated in [24].

Remark 3. Concrete Computation. A common (and conservative) chooice for the functions σ_f, σ_ℓ and σ_V are the Lipschitz constants L_f, L_l and L_v, i.e. $\sigma_f(s) := L_f s$, $\sigma_\ell(s) := L_\ell s$ and $\sigma_V(s) := L_V s$. Then, for $L_f \neq 1$, the additional costs due to the approximated prediction error (29) and 30 can be computed respectively as

$$\Delta L_N(\mu_t) = L_\ell \mu_t \sum_{i=0}^{N-1} L_f^i = L_\ell \mu_t \frac{L_f^N - 1}{L_f - 1} \tag{33}$$

$$\Delta V_N(\mu_t) = L_V L_f^N \mu_t \tag{34}$$

Remark 4. Optimality. The input-to-state stability of the closed-loop system is guaranteed for any value of the terminal weight λ_t greater than the one computed (32). However, a large value of λ_t (for example for conservative bounds) can lead to suboptimal solutions of the MPC problem, hence it's desirable to keep it as low as possible.

C. Prompt-to-State Stable Vision-Language Model Predictive Control (PSS-VLMPC)

A Vision-Language Model is a neural network that can process both visual and textual information, allowing it to understand and generate responses based on these multimodal inputs and its system prompt. In the context of Model Predictive Control, a VLM can be used to predict the MPC parameters such as the reference state and stage cost, based on the visual observation of the system, a textual description of the task from the user and the instructions in the system prompt. Similarly to [27], we propose a Vision-Language Model Predictive Control (VLMPC) framework that consists of two nested control loops: an Outer Loop (OL) that uses a VLM to predict the MPC parameters and an Inner Loop (IL) that solves the MPC problem 13 using the predicted parameters. The OL running at a lower frequency than the IL.

As illustrated in fig. 1 The OL includes, the High Level Controller (HLC), the IL and the Scene Processing Module (SPM). The SPM takes as input the visual observations of the system (e.g. camera views) and the system info (e.g. current state, the last MPC parameters and control inputs) and outputs a schematic view of the system enhanced with the system info. The schematic view is a simplified representation of the system that highlights the relevant features for the control task, gathering all the information in a single image possibly composed of multiple views. The SPM can be implemented using computer vision techniques such as object detection, segmentation and tracking.

The initial role description is set by the user at time zero and it describes the role of the VLM in the control loop. It includes variables that are updated online during the simulation by using the processed scene information from the SPM, e.g. *"You are controlling the tip of a soft robot whose current state is $\{x_{current}\}$ by predicting the MPC parameters..."*.

The HLC includes the Region of Attraction Scaler (ROA scaler) that first checks the correctness of the MPC parameters predicted by the VLM (e.g. by checking if the values of Q and R are positive definite and bounded), secondly it checks the feasibility condition (32) and finally it computes the value of the terminal weight λ_t sufficiently large to guarantee the ISS of the closed-loop system. In practice, a user prompt is held active while the VLM continuously processes incoming scene images, updating the high-level MPC parameters $\mathbf{u}_{VLM}$ at 1Hz. In contrast, the Inner Loop computes the low-level torque command $\mathbf{u}_k^*$ at 50Hz. If any of these scalar checks fail, the ROA scaler rejects the predicted parameters and uses the last valid ones. Consequently, if an invalid prompt is provided, the previous task settings are maintained and the new task is temporarily ignored.

A prompt $\mathbf{p}$ is a sequence of charcters $c \in \mathcal{C}$ where $\mathcal{C}$ is the set of all possible characters, i.e. that can be processed by the VLM. The processed scene image $\mathbf{s}$ is the $w \times h$-grid of pixels with real-valued RBG values output of the SPM, then $f_{SPM}(\cdot)$ is the function that maps the visual observations and system info to the processed scene image.

The HLC can be represented as a function of the prompt $\mathbf{p} = \{c_1, c_2, ..., c_{l_p}\}$ of length l_p, and the processed scene image $\mathbf{s}$, that output the MPC parameters gathered in a vector $\mathbf{u}_{VLM} = [\mathbf{x}_{ref}, Q, R, \lambda_t] \in \mathbb{R}^{2n+m+1}$, i.e. $\mathbf{u}_{VLM} = f_{VLM}(\mathbf{p}, \mathbf{s})$, provided an initial system prompt and a role description are given.

The IL can be represented as a function that takes as input the predicted MPC parameters $\mathbf{u}_{VLM}$, and outputs the info of the system controlled by the MPC and the views $\mathbf{v} = [v_1, v_2, ..., v_{N_v}]$ of the cameras fed to the SPM each at its own resolution of $w_s \times h_s$ pixels, i.e. $\mathbf{o}_{k+1} = f_{IL}(\mathbf{u}_{VLM})$, with $\mathbf{o} \in \mathbb{R}^{n+3N_v(w_s \times h_s)}$ for RGB values, where $s = 1, ..., N_v$ is the camera index and N_v is the number of cameras and for simplicity is assumed that the system info is just the next state $\mathbf{x}_{k+1}$ (altough this can be easily extended to include the last MPC parameters and control inputs).

The outer closed loop system is then given by

$$\mathbf{x}_{k+1} = f_{OCL}(\mathbf{p}_k, \mathbf{x}_k) \tag{35}$$

where the function $f_{OCL}(\cdot)$ composition of the SPM, the

VLM and the IL, i.e. $f_{OCL}(\cdot) := f_{IL} \circ f_{VLM} \circ f_{SPM}$. The prompt $\mathbf{p}$ is the only exogenous signal to the system, arbitrarly provided by the user observing the its behaviour. Now we can introduce the defintion of Prompt-to-State Stability, which requires the IL system 19 to be ISS for every possible prompt.

Definition 1 (Prompt-to-State Stability). Given the set possible charcters $\mathcal{C}$, a maximum context window l_{cw} of the VLM, consider the set of all possible prompt of maximum length l_{cw}

$$P_w = \{\mathbf{p} : \mathbf{p} = \{c_j\}, j = 1, ..., l_p \quad \forall l_p \leq l_{cw}, \forall c_j \in \mathcal{C}\} \tag{36}$$

The outer closed-loop system (35) is said to be Prompt-to-State Stable (PSS) if for every prompt $\mathbf{p}_i \in P_w$, there exists a class $\mathcal{K}_\infty$ function β such that for all initial states $\mathbf{x}_0 \in \mathcal{X}$, the following holds:

$$\|\mathbf{x}_k\| \leq \beta(\|\mathbf{x}_0\|, k) + \sup_{i=0,...,k} \|\mathbf{e}_k\| \tag{37}$$

Assumption 4 (Alignment). The training and the system prompt of the VLM are such that for every processed scene image $\mathbf{s}$, there exists at least one prompt such that it's possible for the VLM to output a vector of MPC parameters $\mathbf{u}_{VLM}$ in a desired format

$$\exists \mathbf{p} \in P_w : f_{VLM}(\mathbf{p}, \mathbf{s}) \in U_{VLM} \subseteq \mathbb{R}^{2n+m+1} \tag{38}$$

where U_{VLM} is the set of valid MPC parameters, i.e. such that Q and R are positive definite and bounded matrices and $\mathbf{x}_{ref}$ is a possible state of the system.

Theorem 2 (Outer Loop Stability). Consider the outer closed-loop system (35), a possible user prompt $\mathbf{p} \in P_w$, and the MPC problem (13). If Assumptions 1-4 hold, the approximated state function (7) is uniformly continuous $\forall \mathbf{x}_k \in \mathcal{X}_N(\lambda)$, a processed scene image $\mathbf{s}$ is available $\forall k \in \mathbb{Z}_{\geq 0}$, and the initial state $\mathbf{x}_0 = x(0)$ is feasible, then the outer closed-loop system (35) is Prompt-to-State Stable with respect to the user prompt $\mathbf{p}$.

Proof: At $k = 0$ the system is initialized such that the reference state is equal to the initial state i.e. $\mathbf{x}_{ref} = \mathbf{x}_0$, and the initial value of R and Q are chosen such that it's possible to find an initial value of λ_t, whose existence comes from *Theorem 1*. Thus there exist an initial parameter vector $\mathbf{u}_{VLM_0}$ that guarantees the IL is ISS.

Consider three sets of prompts P_a, P_b and P_c such that $P_a \cup P_b \cup P_c = P_w$, where P_w is the set (36) and the three sets of MPC parameters U_{VLM_a}, U_{VLM_b} and U_{VLM_c} such that $U_{VLM_a} \cup U_{VLM_b} \cup U_{VLM_c} = U_{VLM} \subseteq \mathbb{R}^{2n+m+1}$, where for any processed scene image $\mathbf{s}$

$$U_{VLM_a} = \{f_{VLM}(\mathbf{p}, \mathbf{s}) : \mathbf{p} \in P_a\} \tag{39}$$

$$U_{VLM_b} = \{f_{VLM}(\mathbf{p}, \mathbf{s}) : \mathbf{p} \in P_b\} \tag{40}$$

$$U_{VLM_c} = \{f_{VLM}(\mathbf{p}, \mathbf{s}) : \mathbf{p} \in P_c\} \tag{41}$$

Without loss of generality we can assume that U_{VLM_a} is the set of valid and feasible MPC parameters, U_{VLM_b} is the set of valid and not feasible MPC parameters and U_{VLM_c} is the set of invalid MPC parameters. Consider a possible user prompt $\mathbf{p}_i \in P_w$ and its corresponding MPC parameters vector $\mathbf{u}_{VLM_i} = f_{VLM}(\mathbf{p}_i, \mathbf{s})$.

From Assumption 4, the sets P_a and P_b are non-empty, hence there exists a prompt $\mathbf{p}_i$ at time k that belongs to either of these two sets. If $\mathbf{p}_i \in P_a$, then $\mathbf{u}_{VLM_i} \in U_{VLM_a}$ and the ROA scaler computes the value of λ_t that guarantees the ISS of the closed-loop system (19) from (32). If $\mathbf{p}_i \in P_b$, then $\mathbf{u}_{VLM_i} \in U_{VLM_b}$ and the MPC problem is not feasible, then the ROA scaler cannot compute a finite value of λ_t that guarantees the ISS of the closed-loop system (19), hence it rejects the predicted parameters and uses the last valid ones $\mathbf{u}_{VLM_{i-1}}$, which are either from U_{VLM_a} or equal to $\mathbf{u}_{VLM_0}$. Similarly if $\mathbf{p}_i \in P_c$, then $\mathbf{u}_{VLM_i} \in U_{VLM_c}$. Being the latter not valid, the ROA scaler rejects the predicted MPC parameters and uses the last valid ones $\mathbf{u}_{VLM_{i-1}}$, which again are either from U_{VLM_a} or equal to $\mathbf{u}_{VLM_0}$, whose stability is guaranteed for either case.

Since $\mathbf{p}_i$ is arbitrary, the IL is ISS for every $\mathbf{p}_i$ in either P_a, P_b or P_c. Since the union of the three sets of prompts cover the whole set of possible prompts of length $l_p \leq l_{cw}$, then for every possible prompt $\mathbf{p}_k$ at time k the closed-loop system (19) is ISS. Hence the outer closed-loop system (35) is Prompt-to-State Stable with respect to the user prompt $\mathbf{p}$, thus concluding the proof. ■

Remark 5. In practice, the slower OL $\mathbf{u}_{VLM}$ acts as a zero-order hold for the faster IL; checking feasibility at every 1Hz update preserves stability across switches. Furthermore, while PSS guarantees safety via the fallback strategy, it does not guarantee task completion for invalid or adversarial prompts.

III. Case Study: Soft Continuum Robot Control

We validate the proposed PSS-VLMPC framework both in simulation and on a real robot. In simulation we consider the case study of a soft continuum robot composed of two rods attached in series along the z-axis, each controlled by two torques τ_{ix} and τ_{iy}, where i is the rod number, and the subscripts x and y indicate the bending axis. The state of the robot is defined by the tip 3D position and velocity, i.e. $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T \in \mathbb{R}^6$. The control input is defined by the four torques applied to the rods, i.e. $\mathbf{u} = [\tau_{1x}, \tau_{1y}, \tau_{2x}, \tau_{2y}]^T \in \mathbb{R}^4$. The dynamics of the robot are learned by a feedforward neural network $f_\Theta(\cdot)$ with 3 hidden layers of 128 neurons each, using Tanh activation functions to guarantee uniform continuity. The input layer has 10 neurons (6 for the state and 4 for the control input) and the output layer has 6 neurons for the next state prediction. The NN is trained on a dataset of 5000 trajectories of highly exciting, dynamic motions. Additional failure cases demonstrating the boundaries of the ROA scaler under aggressive prompts are provided in the supplementary video. Each torque sequence is applied for 7 seconds, followed by a hold time of the last value of 2 seconds and a pause time of 1 second with zero torques. Note that this formulation is equivalent to defining an output function $y = h(x) \in \mathbb{R}^p$, where $p = 6$ and $h(\cdot)$ select the tip position and velocity from the state vector, which can be extended to better describe the system behaviour $x \in \mathbb{R}^n$ and another NN can be used for learning a more complex next state prediction. The neural network is trained to predict the next state of the robot given the current state and control input, using a mean squared error loss function. The training

is performed using the Adam optimizer with a learning rate of 10^{-3}, a batch size of 64 and a weight decay of 10^{-5}. The training is stopped when the validation loss does not improve for 10 consecutive epochs.

All the simulations are perfomed using PyElastica [28], a Python library for simulating the dynamics of Cosserat rods. The IL frequency is 50Hz while the OL runs at 1Hz. Others parameters used in the simulations are summarized in Table I.

TABLE I: Simulation parameters and constraints

Prediction horizon	20
Sim time step	10^{-4} s
MPC time step	0.02 s
VLM time step	1 s
Elements per rod	30
Rod length	0.40 m
Rod radius	0.02 m
Density	500 kg m^{-3}
Young's modulus	1.0×10^5 Pa
Poisson's ratio	0.5
Input constraints $\mathcal{U}$	$\tau_{ix}, \tau_{iy} \in [-0.1, 0.1]$ Nm
State constraints $\mathcal{X}$	$x, y \in [-0.6, 0.6]$ m
	$z \in [0.0, 0.8]$ m
	$\dot{x}, \dot{y}, \dot{z} \in [-0.2, 0.2]$ m/s

The PSS-VLMPC is independent of the specific VLM architecture, as long as it can process visual and textual inputs and produce the required MPC parameters. In this work we tested SmolVLM [29] and Gemini 2.5 [30] as VLM, reporting the results for the latter given the better performance. The final output of the VLM is the next state waypoint, the stage cost matrices Q and R. The full text of the system prompt and the role description can be found in the repository of the project [1], as well as the code for all the simulations reported, including the training of the neural network dynamics and the mpc framework used [23]. For the sake of simplicity, we used a scripted version of the scene encoder that generates the schematic views of the robot and its environment from the available simulation data. The views include the top, side and 3D perspective of the robot within the simulation environment, enriched with the targets and obstacles when present.

A. Rollout Error

Before presenting the results of the case study, we introduce the concept of rollout error, which is used to evaluate the performance of the approximated dynamics in predicting the future states of the system over a finite horizon. We define the j-step rollout error as the approximated prediction error accumulated over the horizon j

$$\mathbf{e}_j := \sum_{k=0}^{j-1} \mathbf{x}_j - \hat{\mathbf{x}}_j \tag{42}$$

where $\hat{\mathbf{x}}_j$ is the predicted state at step j using the approximated dynamics $\hat{f}_{\Theta_t}(\hat{\mathbf{x}}_{j-1}, \mathbf{u}_{j-1})$ and the predicted state at step $j-1$ $\hat{\mathbf{x}}_{j-1}$. Note that $\hat{\mathbf{x}}_0 = \mathbf{x}_0$, then $\mathbf{e}_j = 0$ for $j = 0$. Figure 2 illustrates the concept of rollout error for the case study of a soft continuum robot. As the prediction horizon j increases, the rollout error $\mathbf{e}_j$ tends to accumulate, leading to a larger deviation from the true state. This is due to the fact that the approximated dynamics may not capture all the nuances of the true system dynamics, especially over longer horizons. In the nominal case the MPC performance always increases with the prediction horizon N, however in the uncertain case, a larger prediction horizon can lead to a larger rollout error, which can degrade the performance of the MPC. As a matter of facts, both λ and λ_t are proportional to N, then for larger horizon, a larger terminal weight is needed, whch result in a less optimal controller. Therefore, it's important to find a balance between the prediction horizon and the accuracy of the approximated dynamics to achieve better performance in the uncertain case.

[1] https://github.com/em-ni/PSS-VLMPC

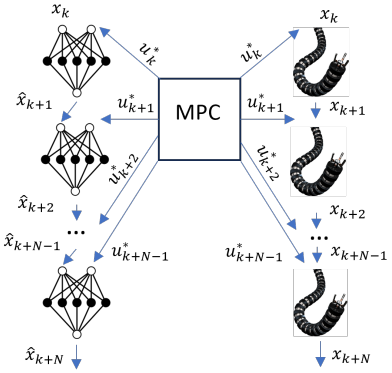

Fig. 2: Visualization of the rollout error for the case study of a soft continuum robot. At each step the MPC computes the optimal control input $\mathbf{u}_k^*$ using the approximated dynamics $\hat{f}_{\Theta_t}(\cdot)$. The latter when applied to the neural network model produces the next predicted state $\hat{\mathbf{x}}_{k+1}$, when applied to the real robot produces the true state $\mathbf{x}_{k+1}$. The rollout error $\mathbf{e}_j$ is the accumulated difference between the true state and the predicted state over a horizon of j steps, and it measure the ability of the approximated dynamics to predict the future states of the system.

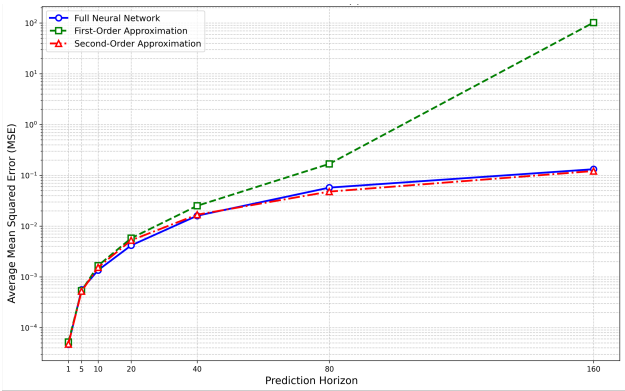

Fig. 3: MSE of the Rollout error (log scale) for different values of the prediction horizon computed over the test set of the trajecotry dataset. The full neural network dynamics (blue) follows the same trend of the second order Taylor expansion (red), yelding an "acceptable" error even for large prediction horizons. The first order Taylor expansion (green) shows a larger error for increasing prediction horizons, which can lead to instability in the closed-loop system for large prediction horizons.

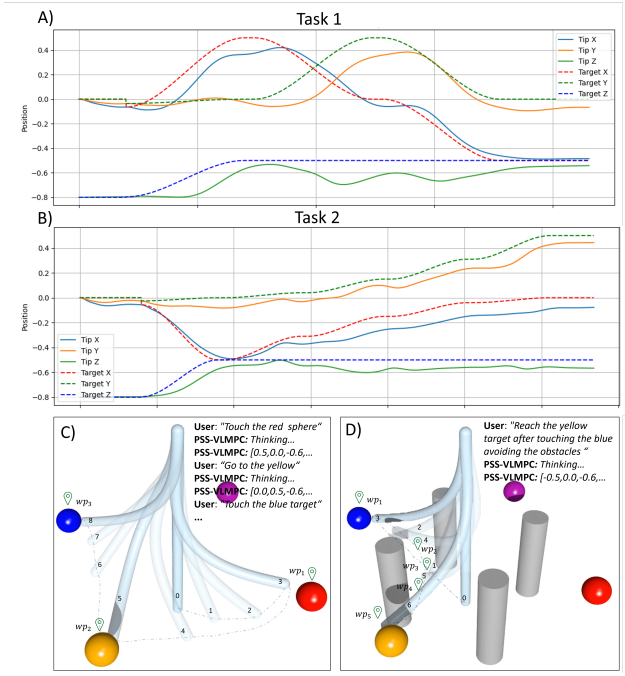

Fig. 4: Results of the simulations for the case study of a soft continuum robot controlled by the proposed PSS-VLMPC framework. A)-B) Tip and target position for the two reaching task. C)-D) Motion trails for the two tasks, showing the waypoints selected by the VLM (green pins) and the trajectory followed by the robot tip (blue dashed line).

B. Simulation Results

We report the results of two simulations of the soft continuum robot controlled by the proposed PSS-VLMPC framework. The first simulation is a reaching task where the robot has to reach different targets in 3D space in a given order. The second simulation is augmented with a set of obstacles that the robot has to avoid while reaching the targets. The results of the simulations are shown in Figure 4. The robot is able to complete the reaching task after being prompted for touching the red, yellow and blue targets in this order, and the obstacle avoidance with the prompt: *"Reach the yellow target after touching the blue avoiding the obstacles"*.

C. Real Robot Experiment

To conclude we validate the proposed PSS-VLMPC framework on a real soft continuum robot made out of three bio-inspired soft hydraulic filament artificial muscles [31], each controlled by an independent motor placed at the base of the syringe to supply the muscle pressure. The robot is able to bend in 3D space by differentially actuating the three muscles. The same considerations made for the simulation apply to this case study, where the only difference is the dimension of the control vector, defined in this case by the three linear displacements applied to the motors, i.e. $\mathbf{u} = [v_1, v_2, v_3]^T \in \mathbb{R}^3$. The neural network dynamics is trained on a dataset of 2000 trajectories of 3 seconds each. Figure 5 shows the experimental setup, and the tasks executed by the robot. The robot is able to complete four reaching tasks similarly to the simulation, and with the addidion of external disturbances. The first task is for reaching two targets in an exact order, the second is for colliding with a target and immediately reaching back the downward position. The third and fourth tasks are reaching tasks with single target with external disturbances applied to the robot during the execution of the task.

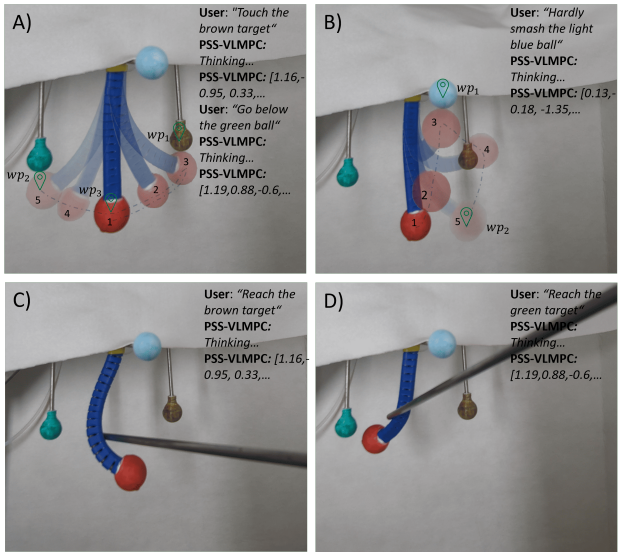

Fig. 5: Results of the experiments for the case study of a soft continuum robot controlled by the proposed PSS-VLMPC framework. A) Motion trail of the Task 1, the robot is first prompted for reaching the brown target, and then instructed to reach the green one on the opposite side of the workspace. B) The robot is prompted for smashing the light-blue target in the upper part of the workspace, resulting in a fast collision and the immediate return to the downward position. C)-D) Snapshots of the third and fourth reaching tasks, the robot is disturbed with an external force while reaching the target, but it is able to recover and complete the task.

IV. Conclusion and Future Works

In this work we presented the Prompt-to-State Stable Vision-Language Model Predictive Control (PSS-VLMPC) framework for controlling complex systems with unknown dynamics. The proposed framework combines the strengths of VLM and MPC to achieve real-time control from natural language input of any system modeled with a neural network, while ensuring stability and performance. The effectiveness of the proposed framework is demonstrated through simulations and experiments on a soft continuum robot, showcasing its ability to execute the desired tasks from the user prompt. A key limitation is the $\sim$1Hz update frequency of the Outer Loop, restricting the framework to similarly slow-evolving environments; thus, fast adversarial events or Scene Processing Module mis-specifications could compromise the safety of reused parameters. Future works will focus on improving the various components of the PSS-VLMPC. First, enhancing the capabilities of the neural network to better capture the system dynamics for more challenging cases (for example soft robots whose body is in contact with the environment, hence learning the zero dynamics of the system). Second, improving the SPM, for building more informative images to give the VLM a clearer vision on the robot and environment states. Third, finetuning or training the VLM on a dataset of robotic tasks to understand better the control context and assign smarter MPC parameters to solve more complex tasks.

REFERENCES

[1] D. Driess *et al.*, "PaLM-E: An embodied multimodal language model," in *Proceedings of the 40th International Conference on Machine Learning*, vol. 202. PMLR, 2023, pp. 8469–8488, generalist vision, language, and robotics model.

[2] A. Brohan *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control." [Online]. Available: https://arxiv.org/abs/2307.15818

[3] L. X. Shi *et al.*, "Hi robot: Open-ended instruction following with hierarchical vision-language-action models," 2025. [Online]. Available: https://arxiv.org/abs/2502.19417

[4] P. Intelligence *et al.*, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," 2025. [Online]. Available: https://arxiv.org/abs/2504.16054

[5] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, "Openvla: An open-source vision-language-action model," 2024. [Online]. Available: https://arxiv.org/abs/2406.09246

[6] H. Huang, F. Liu, L. Fu, T. Wu, M. Mukadam, J. Malik, K. Goldberg, and P. Abbeel, "Otter: A vision-language-action model with text-aware visual feature extraction," 2025. [Online]. Available: https://arxiv.org/abs/2503.03734

[7] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, X. Wang, B. Liu, J. Fu, J. Bao, D. Chen, Y. Shi, J. Yang, and B. Guo, "Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation," 2024. [Online]. Available: https://arxiv.org/abs/2411.19650

[8] K. Long, H. Shi, J. Liu, and X. Li, "Vision language foundation model (vlm)-guided model predictive controller for autonomous driving," *arXiv preprint arXiv:2408.04821*, 2024, vLM-MPC framework for autonomous driving.

[9] X. Zheng, A. K. Mok, R. Piskac, Y. J. Lee, B. Krishnamachari, D. Zhu, O. Sokolsky, and I. Lee, "Testing learning-enabled cyber-physical systems with large-language models: A formal approach," 2024. [Online]. Available: https://arxiv.org/abs/2311.07377

[10] X. Wu, R. Xian, T. Guan, J. Liang, S. Chakraborty, F. Liu, B. M. Sadler, D. Manocha, and A. Bedi, "On the safety concerns of deploying LLMs/VLMs in robotics: Highlighting the risks and vulnerabilities," in *First Vision and Language for Autonomous Driving and Robotics Workshop*, 2024. [Online]. Available: https://openreview.net/forum?id=4FpuOMoxsX

[11] "Robots and robotic devicesâ€"safety requirements for industrial robots (iso 10218)," International Organization for Standardization, 2025.

[12] "Robots and robotic devicesâ€"collaborative robots (iso/ts 15066:2016)," International Organization for Standardization, 2016.

[13] E. K. Jones, A. Robey, A. Zou, Z. Ravichandran, G. J. Pappas, H. Hassani, M. Fredrikson, and J. Z. Kolter, "Adversarial attacks on robotic vision language action models," 2025. [Online]. Available: https://arxiv.org/abs/2506.03350

[14] H. Zhang, C. Zhu, X. Wang, Z. Zhou, C. Yin, M. Li, L. Xue, Y. Wang, S. Hu, A. Liu, P. Guo, and L. Y. Zhang, "Badrobot: Jailbreaking embodied llms in the physical world," 2025. [Online]. Available: https://arxiv.org/abs/2407.20242

[15] X. Wang, H. Pan, H. Zhang, M. Li, S. Hu, Z. Zhou, L. Xue, P. Guo, Y. Wang, W. Wan, A. Liu, and L. Y. Zhang, "Trojanrobot: Physical-world backdoor attacks against vlm-based robotic manipulation," 2025. [Online]. Available: https://arxiv.org/abs/2411.11683

[16] E. Sheetz, E. Zemler, M. Savchenko, C. Rainen, E. Holum, J. Graf, A. Albright, S. Azimi, and B. Kuipers, "Human-robot red teaming for safety-aware reasoning," 2025. [Online]. Available: https://arxiv.org/abs/2508.01129

[17] T. Wang, C. Han, J. C. Liang, W. Yang, D. Liu, L. X. Zhang, Q. Wang, J. Luo, and R. Tang, "Exploring the adversarial vulnerabilities of vision-language-action models in robotics," 2025. [Online]. Available: https://arxiv.org/abs/2411.13587

[18] S. Sanyal and K. Roy, "Asma: An adaptive safety margin algorithm for vision-language drone navigation via scene-aware control barrier functions," 2025. [Online]. Available: https://arxiv.org/abs/2409.10283

[19] M. Black, G. Fainekos, B. Hoxha, H. Okamoto, and D. Prokhorov, "Cbfkit: A control barrier function toolbox for robotics applications," 2024. [Online]. Available: https://arxiv.org/abs/2404.07158

[20] K. Garg, J. Usevitch, J. Breeden, M. Black, D. Agrawal, H. Parwana, and D. Panagou, "Advances in the theory of control barrier functions: Addressing practical challenges in safe control synthesis for autonomous and robotic systems," 2023. [Online]. Available: https://arxiv.org/abs/2312.16719

[21] J. Chen, W. Zhao, Z. Meng, D. Mao, R. Song, W. Pan, and W. Zhang, "Vision-language model predictive control for manipulation planning and trajectory generation," *arXiv preprint arXiv:2504.05225*, 2025, introduces VLMPC and Traj-VLMPC combining VLM with MPC.

[22] J. Nubert, J. KÀ†ohler, V. Berenz, F. Allg€ower, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust mpc and neural network control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3050–3057, 2020.

[23] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, p. 2397â€"2404, Apr. 2023. [Online]. Available: http://dx.doi.org/10.1109/LRA.2023.3246839

[24] D. Limon, T. Alamo, F. Salas, and E. Camacho, "On the stability of constrained mpc without terminal constraint," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 832–836, 2006.

[25] K. Seel, E. I. GrÃ¸tli, S. Moe, J. T. Gravdahl, and K. Y. Pettersen, "Neural network-based model predictive control with input-to-state stability," in *2021 American Control Conference (ACC)*, 2021, pp. 3556–3563.

[26] D. Limon, T. Alamo, D. M. Raimondo, D. M. de la Peña, J. M. Bravo, A. Ferramosca, and E. F. Camacho, *Input-to-State Stability: A Unifying Framework for Robust Model Predictive Control*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–26. [Online]. Available: https://doi.org/10.1007/978-3-642-01094-1_1

[27] K. Long, H. Shi, J. Liu, and X. Li, "Vlm-mpc: Vision language foundation model (vlm)-guided model predictive controller (mpc) for autonomous driving," 2024. [Online]. Available: https://arxiv.org/abs/2408.04821

[28] A. Tekinalp, S. H. Kim, Y. Bhosale, T. Parthasarathy, N. Naughton, A. Albazroun, R. Joon, S. Cui, I. Nasiriziba, M. StÃ¶lzle, C.-H. C. Shih, and M. Gazzola, "Gazzolalab/pyelastica: v0.3.2," Mar. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.10883271

[29] A. Marafioti, O. Zohar, M. FarrÃ©, M. Noyan, E. Bakouch, P. Cuenca, C. Zakka, L. B. Allal, A. Lozhkov, N. Tazi, V. Srivastav, J. Lochner, H. Larcher, M. Morlon, L. Tunstall, L. von Werra, and T. Wolf, "Smolvlm: Redefining small and efficient multimodal models," 2025. [Online]. Available: https://arxiv.org/abs/2504.05299

[30] G. Comanici *et al.*, "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," 2025. [Online]. Available: https://arxiv.org/abs/2507.06261

[31] P. T. Phan, M. T. Thai, T. T. Hoang, N. H. Lovell, and T. N. Do, "Hfam: Soft hydraulic filament artificial muscles for flexible robotic applications," *IEEE Access*, vol. 8, pp. 226 637–226 652, 2020.