From Pixels to Actions: The Hidden Role of SLAM in VLA Model Training and Evaluation

Sandipan Das

January 26, 2026

1 Introduction

Can an autonomous vehicle understand "park next to the red car"? This seemingly simple instruction demands a remarkable confluence of capabilities: seeing the scene, understanding language, knowing where it is, and predicting what happens when it acts. Vision-Language-Action (VLA) models promise to unify these capabilities—but their success quietly depends on solving a much older problem.

This article traces the evolution from early End-to-End driving systems to modern VLAs, revealing a dependency the research community often overlooks: *robust Simultaneous Localization and Mapping (SLAM) is the foundation upon which VLA training, evaluation, and deployment rest.* We show that VLAs are not a complete autonomy stack, but sophisticated sub-systems embedded within a larger stack. They inherit spatial intelligence from classical systems: SLAM provides the pose, maps provide the route, and neural rendering provides the evaluation substrate.

The article is structured as follows. Section 2 traces the architectural evolution from PilotNet to modern VLAs. Section 3 clarifies VLA's role in the autonomy stack for autonomous vehicles and robotics, and its dependencies on upstream systems. Section 4 underlines the connection between VLA evaluation and SLAM, covering pose estimation, neural rendering, and the metric-scale imperative. We conclude that, as more companies embrace this new paradigm of VLAs, we expect to see more evolved neural rendering engines powered by robust SLAM solutions, with the ability to evaluate VLA models seamlessly via a generic interface.

2 Evolution of VLA from End-to-End (E2E) Systems

A VLA model is a type of foundation model that combines visual perception, natural language understanding, and action prediction into a single unified architecture. Building on the success of vision-language models, VLAs extend these capabilities to output actionable decisions—making them particularly suited for robotics and autonomous driving. The core idea is to leverage the rich world knowledge and reasoning abilities encoded in large pretrained models, then fine-tune or prompt them to generate control actions conditioned on both visual observations and language instructions. In the context of autonomous driving, VLAs can interpret complex scenes, understand natural language commands or traffic rules, and directly output driving behaviours, offering a promising path toward more generalizable and interpretable autonomous systems.

Notation. Throughout this article, we use $f_{\text{task}}(\cdot)$ to denote learned neural network functions, where the subscript indicates the function's role (e.g., f_{enc} for encoder, f_{aux} for auxiliary task). The symbol π denotes a policy mapping observations to actions. Bold lowercase letters (e.g., $\mathbf{a}$, $\mathbf{v}$) represent vectors, bold uppercase letters (e.g., $\mathbf{V}$, $\mathbf{T}$) represent matrices or sequences, and calligraphic letters (e.g., $\mathcal{M}$) represent sets.

2.1 End-to-End Learning

Early End-to-End (E2E) systems like PilotNet [1] learn a direct mapping from raw sensor inputs to control outputs, bypassing the traditional modular pipeline of perception, prediction, planning, and control. Formally, an E2E system learns a policy π that maps visual observations to actions as,

$$\mathbf{a} = \pi(\mathbf{V}) \tag{1}$$

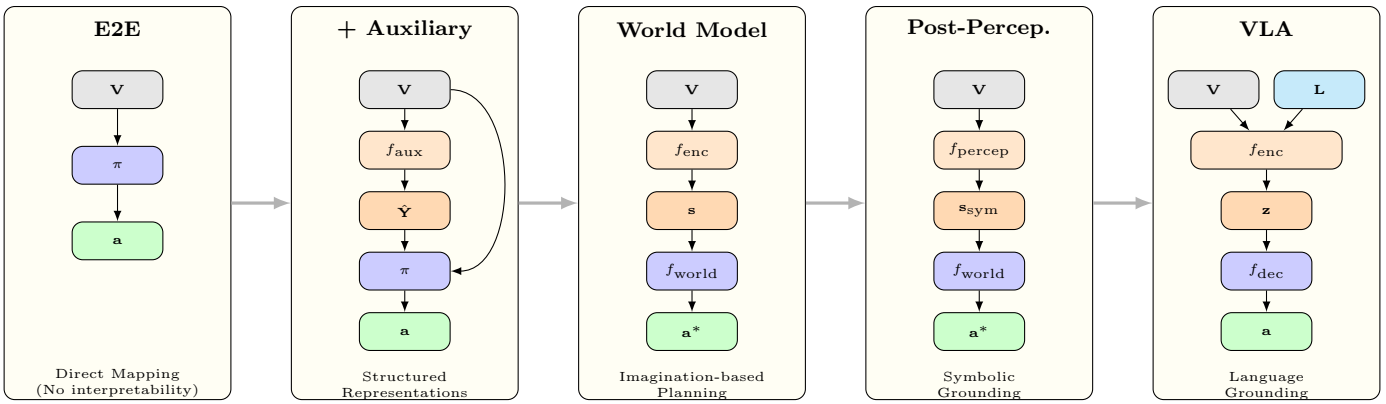

Figure 1: Evolution of autonomous driving architectures. E2E systems directly map vision to action. Auxiliary tasks add structured intermediate supervision. World models enable planning via learned dynamics. Post-perception planning operates on symbolic scene representations. VLAs ground actions in language, enabling semantic instruction following.

where $\mathbf{V} = \{v_1, v_2, \ldots, v_t\}$ represents visual inputs (camera frames) over time and $\mathbf{a}$ represents the output driving actions. Depending on the control interface, actions may be represented as low-level commands $\mathbf{a} = (\delta, \tau, \beta)$ (steering angle, throttle, brake) or as kinematic targets $\mathbf{a} = (\delta, v)$ (steering angle, target velocity). The latter formulation is more common in modern systems as it decouples planning from low-level control. Although simple and elegant, these systems suffered from poor generalization, a lack of interpretability, and difficulty handling rare scenarios not seen during training.

2.2 Auxiliary Task Supervision

Researchers found that adding intermediate supervision with auxiliary tasks—such as depth estimation, semantic segmentation, or object detection—acted as implicit regularizers, forcing the network to learn meaningful representations with improved performance. The policy learning framework thus became,

$$\hat{\mathbf{Y}} = f_{\text{aux}}(\mathbf{V}) \tag{2}$$

$$\mathbf{a} = \pi(\mathbf{V}, \hat{\mathbf{Y}}) \tag{3}$$

where $\hat{\mathbf{Y}}$ represents the auxiliary task outputs that provide structured intermediate representations.

2.3 World Models

World models [2] were built on top of the previous philosophy by learning a predictive model of environment dynamics to simulate future states from sampled action distributions. Actions are then derived through planning within this learned simulator based on reward maximization:

$$\mathbf{s} = f_{\text{enc}}(\mathbf{V}) \tag{4}$$

$$\hat{\mathbf{s}} = f_{\text{world}}(\mathbf{s}, \mathbf{a}) \tag{5}$$

$$\mathbf{a}^* = \arg\max_{\mathbf{a}} \mathbb{E}\left[R(\hat{\boldsymbol{\tau}})\right] \tag{6}$$

where $\mathbf{s}$ is the current latent state, $\hat{\mathbf{s}}$ represents the predicted next state, $\hat{\boldsymbol{\tau}} = \{\hat{\mathbf{s}}', \hat{\mathbf{s}}'', \ldots\}$ represents predicted state roll-outs based on the action distribution, and $R(\hat{\boldsymbol{\tau}})$ is the cumulative reward over the roll-outs. This enables the agent to simulate future trajectories and plan within an imagined latent space, rather than relying solely on reactive mappings.

2.4 Post-Perception Planning

In the autonomous driving and robotics domains, similar principles have been applied with a key modification: instead of operating in an abstract latent space, earlier approaches use *post-perception symbolic outputs*—such

as detected objects, lane graphs, or scene descriptions—as the state representation. This adaptation aligns with the existing modular architecture (Perception → Localization → Prediction & Planning → Control) prevalent in industry:

$$\mathbf{s}_{\text{sym}} = f_{\text{enc}}(\mathbf{V}_{\text{perception}}) \tag{7}$$

$$\hat{\mathbf{s}}_{\text{sym}} = f_{\text{world}}(\mathbf{s}_{\text{sym}}, \mathbf{a}) \tag{8}$$

$$\mathbf{a}^* = \arg\max_{\mathbf{a}} \mathbb{E}\left[R(\hat{\boldsymbol{\tau}})\right] \tag{9}$$

The post-perception planning paradigm evolved from early works like ChauffeurNet [3] and NMP [4], which operated on structured scene representations, followed by explicit prediction-planning factorization in ST-P3 [5] and unified probabilistic models like MP3 [6]. Recent approaches such as DIPP [7] and UniAD [8] achieve tighter integration through differentiable optimization and transformer-based unification. These approaches improve explainability and generalization to out-of-distribution (OOD) edge cases. However, these representations remained non-verbal and lacked semantic grounding.

2.5 Vision-Language Alignment

The success of CLIP [9] and subsequent vision-language models demonstrated that aligning visual and textual representations creates powerful, transferable features through a shared embedding space:

$$\mathbf{z}_v = f_{\text{vision}}(\mathbf{V}) \in \mathbb{R}^d \tag{10}$$

$$\mathbf{z}_l = f_{\text{language}}(\mathbf{L}) \in \mathbb{R}^d \tag{11}$$

The alignment is achieved through contrastive learning, where matching vision-language pairs are pulled together while non-matching pairs are pushed apart:

$$\mathcal{L}_{\text{CLIP}} = -\frac{1}{2}\left[\log \frac{\exp(\mathbf{z}_v \cdot \mathbf{z}_l/\tau)}{\sum_j \exp(\mathbf{z}_v \cdot \mathbf{z}_l^{(j)}/\tau)} + \log \frac{\exp(\mathbf{z}_l \cdot \mathbf{z}_v/\tau)}{\sum_j \exp(\mathbf{z}_l \cdot \mathbf{z}_v^{(j)}/\tau)}\right] \tag{12}$$

where τ is a learned temperature parameter. This showed that language could serve as both a supervisory signal and a means of encoding world knowledge.

2.6 Vision-Language-Action Models

Building on the vision-language alignment, the SOTA Vision-Language-Action (VLA) models [10, 11, 12, 13, 14, 15, 15, 16, 17, 18, 19, 20, 21, 22] learn a policy π that maps visual observations and language instructions directly to actions:

$$\mathbf{a} = \pi(\mathbf{V}, \mathbf{L}) \tag{13}$$

where $\mathbf{V}$ represents visual inputs, $\mathbf{L}$ represents language context (navigation commands, traffic rules, or scene descriptions), and $\mathbf{a}$ represents the output driving actions. Recent work has also proposed incorporating auxiliary tasks within this framework for improved safety and interpretability:

$$\hat{\mathbf{Y}} = f_{\text{aux}}(\mathbf{V}) \tag{14}$$

$$\mathbf{a} = \pi(\mathbf{V}, \hat{\mathbf{Y}}, \mathbf{L}) \tag{15}$$

2.7 VLA Architecture

A VLA model typically encodes both modalities into a shared representation space before decoding actions:

$$\mathbf{z} = f_{\text{enc}}(\mathbf{V}, \mathbf{L}) \tag{16}$$

$$\mathbf{a} = f_{\text{dec}}(\mathbf{z}) \tag{17}$$

This joint embedding allows the network to reason about relationships between what it sees and what it's told, enabling behaviours like "turn left at the red building" by grounding the language instruction in visual features. Language is important in VLAs for several key reasons:

Table 1: Evolution of autonomous driving architectures

Paradigm	Formulation	State	Representation	Main Idea
E2E	$\mathbf{a} = \pi(\mathbf{V})$	Implicit	Latent	Direct sensor-to-action mapping
E2E + Auxiliary	$\mathbf{a} = \pi(\mathbf{V}, \hat{\mathbf{Y}})$	Explicit	Latent + Grounding	Auxiliary tasks as regularization
World Models	$\mathbf{s} = f_{\text{enc}}(\mathbf{V});\ \mathbf{a}^* = \arg\max_{\mathbf{a}} \mathbb{E}[R(\hat{\boldsymbol{\tau}})]$	Implicit	Latent	Imagination-based planning
Post-Perception	$\mathbf{s} = f_{\text{enc}}(\mathbf{V}_{\text{perception}});\ \mathbf{a}^* = \arg\max_{\mathbf{a}} \mathbb{E}[R(\hat{\boldsymbol{\tau}})]$	Explicit	Symbolic	Structured scene representation
VLA	$\mathbf{a} = \pi(\mathbf{V}, \mathbf{L})$	Implicit	Latent + Language	Language as world knowledge
VLA + Auxiliary	$\mathbf{a} = \pi(\mathbf{V}, \hat{\mathbf{Y}}, \mathbf{L})$	Explicit	Latent + Language + Grounding	Safety via structured outputs

Knowledge Injection. Language provides a mechanism to transfer human knowledge directly into the model without requiring exhaustive training data for every scenario.

Generalization. A model trained with language grounding can generalize to novel instructions and scenarios, leveraging semantic understanding from language pretraining.

Interpretability. Language enables explainable decisions through reasoning traces like "braking because pedestrian is stepping onto road."

Unified Representation. Language provides a common embedding space that enables zero-shot transfer, cross-modal reasoning, and leveraging internet-scale pretrained knowledge.

2.8 Language Input Categories

As an example, in the autonomous driving domain, language input $\mathbf{L}$ decomposes into:

Table 2: Language input categories in autonomous driving VLAs

Category	Symbol	Example	Function
Navigation Commands	$\mathbf{L}_{\text{nav}}$	"turn left ahead"	Goal specification
Traffic Rules	$\mathbf{L}_{\text{rules}}$	"yield to pedestrians"	Constraint definition
Scene Descriptions	$\mathbf{L}_{\text{scene}}$	"cyclist approaching on right"	Visual grounding
Chain-of-Thought	$\mathbf{L}_{\text{cot}}$	"slowing down because..."	Reasoning trace

$$\mathbf{L} = \{\mathbf{L}_{\text{nav}}, \mathbf{L}_{\text{rules}}, \mathbf{L}_{\text{scene}}, \mathbf{L}_{\text{cot}}\} \tag{18}$$

Each component shapes the output action $\mathbf{a}$ differently—goals define *what* to achieve, rules define *what not* to do, scene descriptions provide *context*, and reasoning ensures *interpretability*.

3 Role of VLA in the autonomy stack

The preceding section traced the evolution from PilotNet's pixel-to-steering mapping to today's language-conditioned VLAs. But architectural elegance does not imply operational independence. In this section, we step back from *how VLAs work* to ask *where VLAs fit* within the broader autonomy stack. We examine two primary domains—autonomous driving and robotics—and reveal a nuanced dependency landscape. For vehicles navigating open roads, VLAs remain tightly coupled to SLAM and mapping infrastructure. For arms manipulating objects on a tabletop, much of this dependency dissolves. For mobile robots that must do both, the full complexity returns. Recognizing these domain-specific roles is essential for realistic benchmarking and deployment.

3.1 VLA in Autonomous Vehicles

Despite the impressive capabilities of VLA models, it is essential to recognize their *current practical role* in the autonomy stack for autonomous vehicles: In *current publicly documented* deployed systems, VLAs appear to function

primarily as local planners. However, the boundary between local and global reasoning is not sharp, and proprietary production systems may integrate VLAs more deeply than published research suggests. While some research works like Alpamayo-R1 [10] explore end-to-end global reasoning, production systems typically use VLAs for reactive, short-horizon decision-making—responding to immediate visual stimuli and language commands within the context provided by upstream modules.

The distinction between local and global planning clarifies this role. A VLA answers: "*How* do I execute this maneuver safely?" It does not typically answer: "*Which* road leads to the airport?" These are fundamentally different questions requiring different representations—and current VLA deployments depend on traditional mapping and routing systems for the latter.

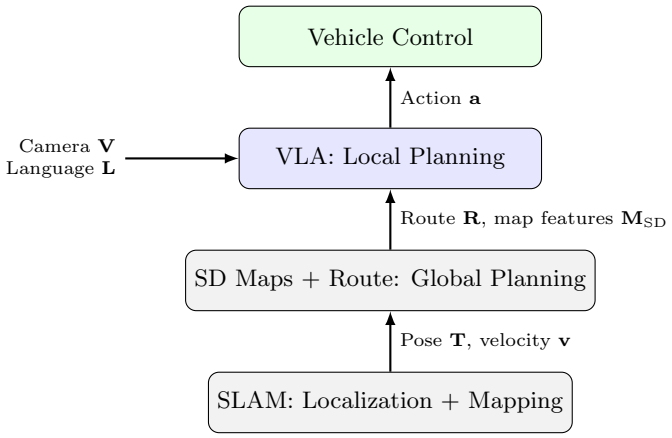

Figure 2: VLA as a local planner within the full autonomy stack for autonomous vehicles. SLAM and SD maps provide essential upstream inputs.

The complete VLA formulation, accounting for these dependencies, in methods such as — Alpamayo-R1 [10], SimLingo [11], Poutine [12], Orion [13], as shown in Fig. 2, becomes:

$$\mathbf{a} = \pi(\mathbf{V}, \mathbf{L}, \mathbf{R}, \mathbf{M}_{\text{SD}}, \mathbf{T}, \mathbf{v}) \tag{19}$$

This expanded formulation reveals the truth: *VLA is not a complete autonomous driving solution—it is a learned local planner conditioned on a rich set of upstream inputs.*

Table 3: Auxiliary inputs required by VLA systems

Input	**Source**	**Purpose**
Visual observations $\mathbf{V}$	Cameras	Scene understanding
Language context $\mathbf{L}$	User commands, rules	Goal and constraint specification
SD Map features $\mathbf{M}_{\text{SD}}$	Pre-built maps	Lane topology, traffic rules
High-level route plan $\mathbf{R}$	SD Map + Routing	Global navigation intent
Ego state $\mathbf{T}, \mathbf{v}$	SLAM system	Spatial grounding, motion context

The Role of SD Maps. Standard Definition (SD) maps provide coarse but crucial information: road topology, lane connectivity, intersection structure, speed limits, and traffic rules. The high-level route plan $\mathbf{R}$—a sequence of road segments from origin to destination—is derived from these maps:

$$\mathbf{R} = \texttt{RoutePlanner}(\mathbf{M}_{\text{SD}}, \mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}}) \tag{20}$$

Without this global plan, a VLA has no notion of *where to go*. It can react to "turn left" but cannot decide *when* to turn left to reach a destination. Language provides local intent; maps provide global context.

The Role of Pose. High-quality pose estimation $\mathbf{T} \in SE(3)$ is not optional—it is essential for:

- **Map alignment:** Querying relevant SD map features requires knowing where the ego vehicle is on the map.
- **Temporal consistency:** Fusing observations across time requires accurate ego-motion estimation.
- **Action grounding:** Predicted actions are executed relative to the current pose; errors compound over time.

The Overlooked Truth. Much of the VLA literature focuses on architecture innovations: better vision encoders, larger language models, more sophisticated action decoders. But the truth is that these models inherit their spatial intelligence from classical systems. The language model knows "turn left at the intersection" has meaning only because a SLAM system knows where the intersection is, and an SD map knows which lane leads where. VLAs are not replacing the autonomy stack—they are a sophisticated component within it. Recognizing this dependency is essential for realistic evaluation, meaningful benchmarking, and successful deployment.

This has practical implications. When a VLA-based vehicle fails, the root cause may lie upstream—in map errors, localization drift, or routing mistakes—rather than in the VLA itself. Debugging requires examining the *entire* stack, not just the learned policy. Similarly, reported benchmark performance is only meaningful if the quality of upstream inputs (poses, maps, routes) is held constant or explicitly characterized.

3.2 VLA in Robotics: A Different Dependency Landscape

However, in the robotics domain—particularly for manipulation tasks such as robotic arms—the dependency landscape shifts significantly. Unlike autonomous vehicles that navigate large-scale, dynamic environments, manipulation robots operate in spatially constrained workspaces with limited mobility. This fundamental difference reshapes the role of SLAM and its relationship to VLA models. Fig. 3 illustrates the simplified dependency stack for fixed-base manipulation.

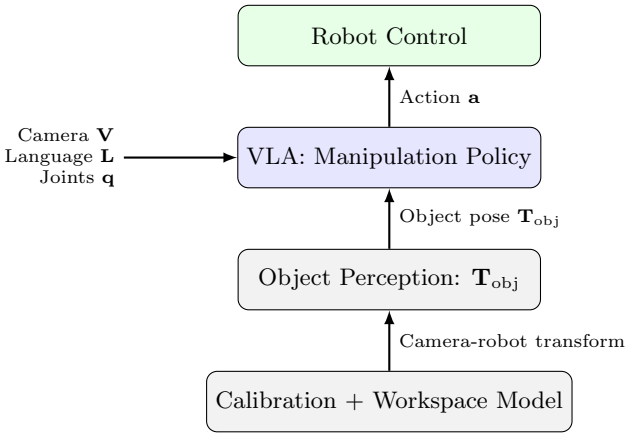

Figure 3: VLA in fixed-base manipulation. SLAM is replaced by calibration and object perception; the robot's kinematics provide proprioceptive state.

3.2.1 Fixed-Base Manipulation

Reduced Mapping Requirements. A robotic arm mounted on a fixed base does not traverse kilometers of roads or encounter novel environments during operation. Its workspace—typically a tabletop, conveyor belt, or assembly station—can be characterized once during setup. The "mapping" component of SLAM, which constructs and maintains representations of large-scale environments for navigation and route planning, becomes largely unnecessary. Instead, the scene can be represented through:

- **Fixed calibration:** Camera-to-robot transforms established during installation
- **Workspace models:** Pre-defined geometric boundaries and known fixture locations
- **Real-time perception:** Object detection and pose estimation within the known workspace

State Estimation Remains Essential. While mapping requirements diminish, state estimation remains critical—though its scope changes. For manipulation, the relevant "pose" is not the robot's position in a global coordinate frame, but rather:

$$\mathbf{T}_{\text{ee}} = \text{FK}(\mathbf{q}) \quad \text{(end-effector pose from joint angles)} \tag{21}$$

$$\mathbf{T}_{\text{obj}} = f_{\text{percep}}(\mathbf{V}) \quad \text{(object pose from visual observation)} \tag{22}$$

where $\mathbf{q} \in \mathbb{R}^n$ represents joint configurations, FK is the forward kinematics function, and $\mathbf{T}_{\text{ee}}, \mathbf{T}_{\text{obj}} \in SE(3)$ are the end-effector and object poses respectively. The VLA must reason about the spatial relationship between gripper and target:

$$\mathbf{T}_{\text{rel}} = \mathbf{T}_{\text{ee}}^{-1} \cdot \mathbf{T}_{\text{obj}} \tag{23}$$

VLA Models in Manipulation. VLA models trained for manipulation—such as RT-1 [14], RT-2 [15], OpenVLA [16], Octo [17], and π_0 [18] - $\pi^*_{0.6}$ [19]—learn policies of the form:

$$\mathbf{a} = \pi(\mathbf{V}, \mathbf{L}, \mathbf{q}) \tag{24}$$

where actions $\mathbf{a}$ may be end-effector velocities, joint velocities, or target poses. The language input $\mathbf{L}$ typically specifies task-level goals ("pick up the red block", "place the cup on the coaster") rather than navigation commands. Crucially, these models benefit from:

- **Simpler geometry:** The workspace is bounded and largely static
- **Known kinematics:** Robot joint configurations provide precise proprioceptive state
- **Controlled environments:** Lighting, backgrounds, and camera viewpoints can be standardized
- **Dense supervision:** Teleoperation enables collection of expert demonstrations at scale

3.2.2 Mobile Manipulation

The distinction between driving and manipulation blurs for *mobile manipulation* platforms—robots that navigate environments *and* interact with objects (e.g., home robots, warehouse robots, humanoids). For these systems, the full SLAM dependency returns:

$$\mathbf{a} = \pi(\mathbf{V}, \mathbf{L}, \mathbf{q}, \mathbf{R}, \mathbf{M}, \mathbf{T}_{\text{base}}, \mathbf{v}) \tag{25}$$

where $\mathbf{T}_{\text{base}}$ is the mobile base pose from SLAM along with motion primitives $\mathbf{v}$, $\mathbf{q}$ is the arm configuration, $\mathbf{R}$ is the high-level route based on the stored map, and $\mathbf{M}$ is the environment map. Mobile manipulation VLAs must jointly reason about *where to go* and *how to interact*—combining the challenges of both domains. Systems like TidyBot [20], OK-Robot [21], and Mobile ALOHA [22], and emerging humanoid platforms (e.g., Figure 01, Tesla Optimus) represent early steps toward this unified capability, and their success critically depends on robust SLAM foundations.

3.3 Domain-Dependent Dependencies of VLA

Table 4: SLAM-VLA dependency across robotic domains

Aspect	**Driving VLA**	**Manipulation VLA**	**Mobile Manipulation VLA**
Mapping required	Yes (large-scale)	No (workspace model)	Yes (environment map)
Localization required	Yes (global + local)	Partial (kinematics)	Yes (base + arm)
Metric scale source	SLAM / sensor fusion	Robot geometry	SLAM / sensor fusion
Primary spatial reasoning	Ego-to-world	Gripper-to-object	Both
SLAM dependency	**High**	**Low**	**High**

The SLAM-VLA dependency is not universal but *domain-dependent*. For autonomous driving, SLAM is foundational—providing poses, enabling map queries, and grounding actions in metric space. For fixed-base manipulation,

SLAM is largely replaced by calibration and forward kinematics, though object pose estimation remains critical. For mobile manipulation, the full dependency re-emerges as robots must navigate *and* interact. Understanding this spectrum is essential for designing VLA systems appropriate to each application context, and for interpreting benchmark results with appropriate nuance.

4 The Connection Between VLA evaluation and SLAM

We've traced the evolution from PilotNet to VLA—from raw pixels to language-grounded actions. But a critical question remains: *how do we evaluate these models?* Here's the uncomfortable truth. You cannot evaluate a VLA like you evaluate a classifier. There's no ImageNet for embodied AI. When a VLA predicts "steer left," the agent moves, the camera sees something new, and that new observation becomes the input for the next decision. Evaluation becomes simulation. Simulation requires rendering. Rendering requires knowing *exactly* where you are. This dependency reveals an unexpected connection: cutting-edge VLA systems are fundamentally constrained by a classical robotics problem—SLAM. Without robust SLAM, we cannot build the simulators needed to evaluate VLAs. Without evaluation, we cannot trust them. The connection is hidden in plain sight. While many VLA benchmarks sidestep this dependency by using ground-truth poses from privileged sensor suites, this evaluation paradigm is unrealistic for deployment. Preliminary studies suggest that even small pose errors can cause significant downstream degradation in closed-loop performance, though systematic characterization remains an open problem.

4.1 The Importance of Pose Estimation

At the core of any autonomous system lies the ability to answer a deceptively simple question: *where am I?* Accurate pose estimation—determining the position and orientation of the agent in the environment—is foundational to both robotics and autonomous driving. Formally, pose estimation seeks to recover the transformation:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3) \tag{26}$$

where $\mathbf{R} \in SO(3)$ represents rotation and $\mathbf{t} \in \mathbb{R}^3$ represents translation in 3D space. The velocity vectors, $\mathbf{v} \in \mathbb{R}^3$, are also coupled with pose to provide the motion context to the VLA models. For VLA models, pose estimation serves multiple critical functions:

Action Grounding. The predicted action $\mathbf{a} = (\delta, \tau, \beta)$ must be executed relative to the agent's current pose. Without accurate localization, even perfect action predictions result in erroneous trajectories.

Temporal Consistency. VLA models process sequences of observations $\mathbf{V} = \{v_1, v_2, \ldots, v_t\}$. Consistent pose estimates across frames enable the model to reason about motion and change over time.

Scene Understanding. Language instructions like "turn left at the intersection" require understanding the agent's position relative to semantic landmarks—a task that fundamentally depends on localization.

One might argue that sufficiently large foundation models will learn implicit geometric reasoning, subsuming SLAM capabilities. While promising, current evidence suggests otherwise: VLAs still require explicit pose inputs for consistent action grounding, and no published VLA has demonstrated robust metric-scale reasoning without external localization. The question is not whether geometric reasoning can be learned, but whether it can be learned *reliably enough* for safety-critical deployment.

4.2 Scene Reconstruction: From Point Clouds to Neural Rendering

Beyond localization, autonomous systems require rich representations of their environment. SLAM systems construct sparse or dense point cloud maps:

$$\mathcal{M} = \{(\mathbf{p}_i, \mathbf{c}_i)\}_{i=1}^{N} \tag{27}$$

where $\mathbf{p}_i \in \mathbb{R}^3$ represents 3D point positions and $\mathbf{c}_i$ represents associated attributes (colour, semantics, etc.). While sufficient for localization and obstacle avoidance, these representations lack the photorealistic detail needed for modern learning-based systems.

Neural Radiance Fields (NeRF). NeRF [23] revolutionized scene reconstruction by representing scenes as continuous volumetric functions:

$$F_\theta : (\mathbf{x}, \mathbf{d}) \to (\mathbf{c}, \sigma) \tag{28}$$

where $\mathbf{x} \in \mathbb{R}^3$ is a spatial position, $\mathbf{d} \in \mathbb{S}^2$ is a viewing direction, $\mathbf{c} \in \mathbb{R}^3$ is the emitted colour, and $\sigma \in \mathbb{R}^+$ is the volume density. Images are rendered via volumetric integration along camera rays:

$$\hat{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})\, dt \tag{29}$$

where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))\, ds\right)$ is the accumulated transmittance.

3D Gaussian Splatting (3DGS). More recently, 3D Gaussian Splatting [24] offers an explicit representation using anisotropic 3D Gaussians:

$$G(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \tag{30}$$

where $\boldsymbol{\mu} \in \mathbb{R}^3$ is the mean position and $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ is the covariance matrix. Each Gaussian carries additional attributes including opacity α and view-dependent colour represented via spherical harmonics. The key advantage of 3DGS is real-time rendering through efficient rasterization, making it particularly suitable for robotics applications.

The Common Requirement. Both NeRF and 3DGS share a fundamental prerequisite: *accurate camera poses* and *sparse point cloud map of the scene*. These neural rendering techniques optimize scene representations given known viewpoints. The quality of the reconstruction is directly bounded by the accuracy of the input poses:

$$\mathcal{L}_{\text{render}} = \sum_i \left\| \hat{I}(\mathbf{T}_i) - I_i \right\|^2 \tag{31}$$

where $\hat{I}(\mathbf{T}_i)$ is the rendered image at pose $\mathbf{T}_i$ and I_i is the ground truth image. Errors in $\mathbf{T}_i$ propagate directly into reconstruction artifacts.

4.3 VLA Evaluation: The Closed-Loop Challenge

The evaluation of VLA models presents a unique challenge that exposes the deep connection to SLAM. Unlike traditional perception tasks where evaluation is straightforward (compare predictions to ground truth labels), VLA evaluation requires *closed-loop simulation*—the predicted actions must affect future observations along with reactive dynamic agents.

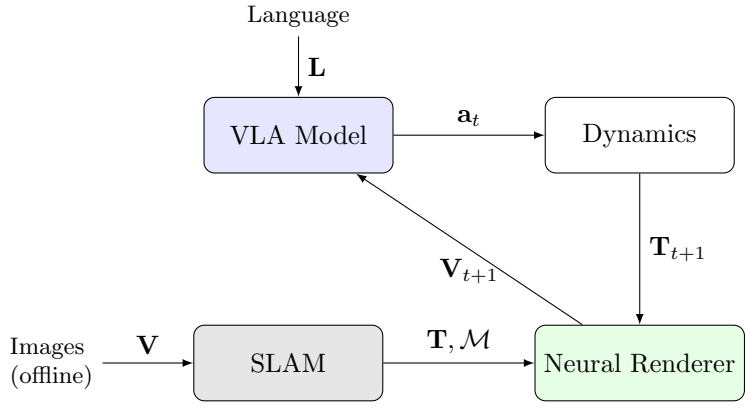

Figure 4: Closed-loop VLA evaluation requires rendering novel views based on predicted actions. SLAM provides accurate poses $\mathbf{T}$ and scene reconstruction $\mathcal{M}$ to enable neural rendering. Language context $\mathbf{L}$ conditions the policy throughout the evaluation loop.

The Evaluation Loop. Consider the VLA inference cycle:

$$\mathbf{a}_t = \pi(\mathbf{V}_t, \mathbf{L}) \tag{32}$$

$$\mathbf{T}_{t+1} = f_{\text{dynamics}}(\mathbf{T}_t, \mathbf{a}_t) \tag{33}$$

$$\mathbf{V}_{t+1} = f_{\text{render}}(\mathcal{M}, \mathbf{T}_{t+1}) \tag{34}$$

At each timestep, the VLA predicts an action $\mathbf{a}_t$ based on current visual observations $\mathbf{V}_t$ and language context $\mathbf{L}$. This action updates the agent's pose via a dynamics model. Critically, the next observation $\mathbf{V}_{t+1}$ must be *rendered from the new viewpoint*—a viewpoint that may never have been observed in the original data, which is illustrated in Fig. 4.

Novel View Synthesis. This requirement for extrapolated view synthesis is where neural rendering becomes essential:

$$\mathbf{V}_{t+1} = \texttt{Render}(\mathcal{M}_{\text{neural}}, \mathbf{T}_{t+1}) \tag{35}$$

The rendered image at the extrapolated pose—a process termed *Extrapolated View Synthesis* (EVS)—serves as input for the next action inference, thereby closing the evaluation loop. The fidelity of EVS directly determines evaluation validity: rendering artifacts, geometric distortions, or lighting inconsistencies can cause VLA models to exhibit behaviours that would not occur in the real world.

EVS Simulators. Recognising this importance, the community has developed several EVS-capable simulators. Open-source options include HUGSIM [25] and AlpaSim (Nvidia) [26]. Industry players—including Tesla, Aurora, Wayve, Waymo, and GM—have deployed proprietary variants built on similar principles. Public urban EVS benchmarks [27] now enable a standardized evaluation of rendering quality in extrapolated views.

The Reactive Agent Gap. While EVS for static scenes is maturing, true closed-loop evaluation requires more: dynamic agents (vehicles, pedestrians, cyclists) must *react* to ego behaviour. Current simulators typically replay recorded trajectories or use rule-based reactive models, neither of which captures realistic multi-agent interactions. Bridging this gap—combining photorealistic EVS with behaviourally realistic agent simulation—is an active and critical research direction.

The Metric Scale Gap. The current EVS benchmarks heavily focus on the rendering side of things. However, an equally important aspect would be the metric scale scene reconstruction, which will help in evaluating VLA models with proper scale intuition.

4.4 The Metric Scale Imperative

A subtle but critical requirement for VLA evaluation is *metric scale reconstruction*. Many visual SLAM and structure-from-motion systems recover geometry only up to an unknown scale factor:

$$\tilde{\mathbf{t}} = s \cdot \mathbf{t}, \quad \tilde{\mathbf{p}} = s \cdot \mathbf{p} \tag{36}$$

where $s \in \mathbb{R}^+$ is the scale ambiguity. This ambiguity is problematic for VLA evaluation for several reasons:

Action Consistency. VLA models predict actions that must be converted to ego-motion updates. If the reconstructed scene is at an incorrect scale, the dynamics model produces incorrect pose updates. Formally, the pose update follows:

$$\mathbf{T}_{t+1} = \mathbf{T}_t \cdot \exp(\boldsymbol{\xi}_t \cdot \Delta t) \tag{37}$$

where $\boldsymbol{\xi}_t = g(\mathbf{a}_t, \mathbf{v}_t) \in \mathfrak{se}(3)$ is the twist (instantaneous velocity) derived from the predicted action and current velocity via a kinematic or dynamic vehicle model. A scale error $s \neq 1$ causes the agent to "teleport" too far ($s > 1$) or too little ($s < 1$) with each action, leading to compounding trajectory errors.

Physical Plausibility. Language instructions often contain implicit metric information: "stop 3 meters before the crosswalk" or "maintain safe following distance." Evaluating such instructions requires metric-accurate reconstruction.

Cross-Sequence Consistency. Evaluating VLA generalization requires testing across multiple scenes. Without a metric scale, comparing performance across sequences becomes meaningless.

Solutions for Metric Scale. Recovering metric scale typically requires additional sensors or constraints:

- **IMU Integration:** Inertial measurements provide metric acceleration, enabling Visual-Inertial Odometry (VIO) to recover true scale.
- **Known Objects:** Detecting objects of known size (e.g., standard traffic signs, lane widths) provides scale anchors.
- **LiDAR Fusion:** Direct depth measurements from LiDAR inherently provide metric scale.
- **Ground Plane Constraint:** Assuming known camera height above ground constrains the scale.

4.5 The Full Dependency Stack

We can now synthesize the dependencies identified throughout this article. VLA training requires accurately posed sequences and map annotations (Section 3). VLA evaluation requires closed-loop simulation with metric-scale novel view synthesis (Section 4.1—4.4). Both requirements bottom out at SLAM as shown in Fig. 5.

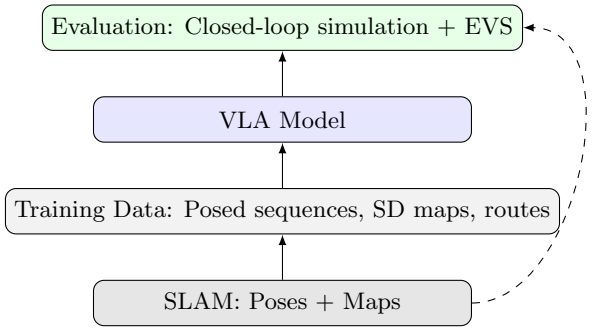

Figure 5: The full dependency stack: SLAM underpins both training data and evaluation infrastructure.

Implications for Research. This dependency stack has important implications:

1. **Dataset requirements:** VLA training datasets must include high-quality pose, SD map features, and route annotations—not just images and actions.
2. **Evaluation fidelity:** Benchmarks that omit map or pose inputs evaluate a simplified—and unrealistic—version of the VLA problem.
3. **Deployment readiness:** A VLA model is only as deployable as the localization and mapping systems it depends on. State-of-the-art VLA performance in simulation may not transfer to vehicles with inferior SLAM.
4. **Failure attribution:** When a VLA-based vehicle fails, the root cause may lie upstream—in map errors, localization drift, or routing mistakes—rather than in the VLA itself.

5 Conclusion

The journey from PilotNet's simple pixel-to-steering mapping to today's VLA models represents a remarkable evolution in autonomous systems. VLAs promise to leverage the rich world knowledge encoded in large pretrained models, enabling systems that can interpret complex scenes, understand natural language commands, and output driving behaviours with unprecedented generalizability and interpretability. VLAs are not fully autonomous solutions—they are sophisticated sub-systems embedded within a larger stack. They inherit their spatial intelligence from classical systems: SLAM provides the pose, maps provide the route, and neural rendering provides the evaluation substrate. Strip away these foundations, and even the most capable VLA cannot answer the question every

robot must answer: *where am I, and where am I going?* The connection between VLA and SLAM is not incidental but fundamental.

Looking ahead, we anticipate—and advocate for—tighter integration between these historically separate communities. The question is no longer *whether* VLA and SLAM will converge, but *how elegantly* they can be unified. The most sophisticated language-conditioned policy is only as good as its ability to know where it is and what the world looks like from where it's going. In this sense, the future of autonomous systems lies not in VLA *or* SLAM, but in their synthesis—systems that jointly reason about perception, language, action, and geometry.

Key Takeaways.

1. VLAs represent a paradigm shift by grounding actions in language and world knowledge.
2. VLAs are sophisticated subsystems that do local planning—practically, they still depend on upstream systems for pose, maps, and global routes.
3. Closed-loop evaluation requires metric-scale 3D reconstruction and novel view synthesis for rendering extrapolated views. However, there is a current gap in reactive agent synthesis, which researchers are actively working on.
4. Robust SLAM is the hidden foundation upon which VLA training, evaluation, and deployment depend.
5. The future lies in unified systems that elegantly bridge language, action, and geometry.

Final Thought. A purist E2E perspective might view explicit SLAM as an unnecessary constraint—the network should learn localization implicitly. In principle, this is true. In practice, current VLAs have not demonstrated this capability at the reliability required for deployment. Our argument is not that SLAM *must* remain explicit forever, but that *today's* VLAs depend on it—and evaluation frameworks should reflect this reality rather than assume it away. The next time you read a VLA paper, look for the localization section. If it's missing, the reported performance is incomplete. If it assumes perfect poses, the results are optimistic. The field needs benchmarks that jointly evaluate VLA reasoning *and* the SLAM systems that enable it. Until then, ask the simple question: *where did the poses come from?*

References

[1] Mariusz Bojarski et al. "End to end learning for self-driving cars". In: *arXiv preprint arXiv:1604.07316* (2016).

[2] David Ha and Jürgen Schmidhuber. "Recurrent world models facilitate policy evolution". In: *Advances in neural information processing systems* 31 (2018).

[3] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst". In: *arXiv preprint arXiv:1812.03079* (2018).

[4] Wenyuan Zeng et al. "End-to-end interpretable neural motion planner". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 8660–8669.

[5] Shengchao Hu et al. "St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning". In: *European Conference on Computer Vision*. Springer. 2022, pp. 533–549.

[6] Sergio Casas, Abbas Sadat, and Raquel Urtasun. "MP3: A Unified Model to Map, Perceive, Predict and Plan". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 14398–14407. DOI: 10.1109/CVPR46437.2021.01417.

[7] Zhiyu Huang et al. "Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving". In: *IEEE transactions on neural networks and learning systems* 35.11 (2023), pp. 15222–15236.

[8] Yihan Hu et al. "Planning-oriented autonomous driving". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 17853–17862.

[9] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PmLR. 2021, pp. 8748–8763.

[10] Yan Wang et al. "Alpamayo-r1: Bridging reasoning and action prediction for generalizable autonomous driving in the long tail". In: *arXiv preprint arXiv:2511.00088* (2025).

[11] Katrin Renz et al. "Simlingo: Vision-only closed-loop autonomous driving with language-action alignment". In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025, pp. 11993–12003.

[12] Luke Rowe et al. "Poutine: Vision-Language-Trajectory Pre-Training and Reinforcement Learning Post-Training Enable Robust End-to-End Autonomous Driving". In: *arXiv preprint arXiv:2506.11234* (2025).

[13] Haoyu Fu et al. "Orion: A holistic end-to-end autonomous driving framework by vision-language instructed action generation". In: *arXiv preprint arXiv:2503.19755* (2025).

[14] Anthony Brohan et al. "Rt-1: Robotics transformer for real-world control at scale". In: *arXiv preprint arXiv:2212.06817* (2022).

[15] Anthony Brohan et al. "Rt-2: Vision-language-action models transfer web knowledge to robotic control. arXiv". In: *arXiv preprint arXiv:2307.15818* (2023).

[16] Moo Jin Kim et al. "OpenVLA: An Open-Source Vision-Language-Action Model". In: *arXiv preprint arXiv:2406.09246* (2024).

[17] Octo Model Team et al. "Octo: An open-source generalist robot policy". In: *arXiv preprint arXiv:2405.12213* (2024).

[18] Kevin Black et al. "π_0: A Vision-Language-Action Flow Model for General Robot Control". In: *arXiv preprint arXiv:2410.24164* (2024).

[19] Physical Intelligence et al. "$\pi^*_{0.6}$: a VLA That Learns From Experience". In: *arXiv preprint arXiv:2511.14759* (2025).

[20] Jimmy Wu et al. "Tidybot: Personalized robot assistance with large language models". In: *Autonomous Robots* 47.8 (2023), pp. 1087–1102.

[21] Peiqi Liu et al. "Ok-robot: What really matters in integrating open-knowledge models for robotics". In: *arXiv preprint arXiv:2401.12202* (2024).

[22] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. "Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation". In: *Conference on Robot Learning (CoRL)*. 2024.

[23] Ben Mildenhall et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.

[24] Bernhard Kerbl et al. "3D Gaussian splatting for real-time radiance field rendering." In: *ACM Trans. Graph.* 42.4 (2023), pp. 139–1.

[25] Hongyu Zhou et al. "Hugsim: A real-time, photo-realistic and closed-loop simulator for autonomous driving". In: *arXiv preprint arXiv:2412.01718* (2024).

[26] NVIDIA et al. *AlpaSim: A Modular, Lightweight, and Data-Driven Research Simulator for Autonomous Driving*. Oct. 2025. URL: `https://github.com/NVlabs/alpasim`.

[27] Xiangyu Han et al. "Extrapolated urban view synthesis benchmark". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2025, pp. 28718–28728.