Neural Processing Letters

https://doi.org/10.1007/s11063-026-11860-3

Article in Press

Assessment of a Fine-Tuned Vision-Language-Action Model for Robotic Feature-Following Inspection

Received: 12 Jan 2026

Accepted: 14 May 2026

Published online: 29 May 2026

Cite this article as: Krüger, M., Salem, M., Reischl, M. *et al.* Assessment of a Fine-Tuned Vision-Language-Action Model for Robotic Feature-Following Inspection. *Neural Process Lett* (2026). https://doi.org/10.1007/s11063-026-11860-3

Martin Krüger, Mahmoud Salem & Markus Reischl

Assessment of a Fine-Tuned Vision-Language-Action Model for Robotic Feature-Following Inspection

Martin Krüger[1*], Mahmoud Salem[1] and Markus Reischl[1]

[1]Institute for Automation and Applied Informatics (IAI), Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, Eggenstein-Leopoldshafen, 76344, Baden-Württemberg, Germany.

*Corresponding author(s). E-mail(s): martin.krueger2@kit.edu;
Contributing authors: mahmoud.salem@kit.edu; markus.reischl@kit.edu;

Abstract

Modern manufacturing increasingly relies on robotics to achieve high throughput and quality, especially as production lines become more flexible and parts more customized. Robotic inspection is a critical enabler for quality assurance as it supports repeatable measurements while reducing human workload and variability. Recent vision-language-action (VLA) models have advanced robotic manipulation by integrating visual perception and language understanding for autonomous control. However, the application to robotic inspection, which requires accurate movement without altering the environment, remains underexplored. This work investigates the feasibility of adapting manipulation-pretrained VLA models to an inspection-oriented feature-following task and presents the following contributions: Tailored to the requirements of inspection problems, two approaches for assessing VLA performance are introduced: A trajectory-based evaluation metric to quantify performance in rollouts as well as an action-level metric, useful during the fine-tuning process. In addition, an open-source, manipulation-pretrained VLA model is fine-tuned for a feature-following task. This task represents a simplified 2D inspection setting, designed to capture core aspects of inspection problems encountered in domains such as manufacturing and infrastructure. The model successfully executes these complex feature-following trajectories with competitive performance relative to a human operator in a real-world robotic setup, demonstrating effective transfer from manipulation to this class of inspection tasks. While the study is limited in scale, the results provide initial evidence that VLA models can be extended beyond manipulation to support feature perception and motion generation in automated, robotic inspection. This suggests

their potential to support more consistent and automated inspection processes, motivating further investigation into robustness and generalization.

Keywords: Vision-Language-Action Models, Robotic Inspection, Robot Learning, Computer Vision, Embodied AI.

1 Introduction

Modern industrial domains, driven by flexibility and human-robot collaboration, increasingly require systems that combine artificial intelligence (AI) with robotics to cope with variability, uncertainty, and rapid changes in production demands. In this context, embodied AI refers to the integration of AI capabilities into a robotic system, enabling the robot to perceive, reason, and act in the physical world through its sensors and actuators. As a result, robotics systems have evolved beyond repetitive routines toward adaptive systems that can operate in unstructured environments [1], collaborate safely with humans [2], and respond to changing task requirements [3]. A key driver of this progress is robot learning, in which control policies are derived from machine-learning models rather than being engineered by hand. This learning approach enables generalization across objects, scenes, and variations in execution. Within robot learning, vision-language-action (VLA) models have recently emerged as a promising direction. VLAs integrate the semantic reasoning capabilities of large language models (LLMs) with the perceptual grounding of vision-language models (VLMs) to produce robotic actions. This integration allows robots to interpret natural-language instructions and translate visual observations directly into action sequences [4, 5].

VLAs are often VLM models that are retrained on large corpora of robotic trajectories covering different embodiments and tasks. The training data typically consists of image-prompt pairs sampled along demonstration trajectories, where the prompt remains constant throughout each trajectory. A typical example of such a prompt is "Pick up the Coke can" [4, 6, 7]. One approach used by some VLA models to generate motion is to predict a 7D action vector, where the first three components correspond to Cartesian position offsets $(\Delta x, \Delta y, \Delta z)$, the next three to orientation offsets $(\Delta\text{roll}, \Delta\text{pitch}, \Delta\text{yaw})$, and the final component to the gripper state. The pose offsets are applied additively to the current end-effector pose for execution.

Recent VLAs have demonstrated notable generalization, adapting to unseen visual conditions, handling new objects, and transferring policies across different robotic platforms. Despite this progress, VLAs are most widely studied for manipulation tasks such as picking, placing, or rearranging items. For instance, RT-2 [7] presents VLM-based VLA models adapted for end-to-end robotic control and evaluates them through manipulation rollouts, i.e., full task executions obtained by sequentially applying predicted actions. Similarly, the OpenVLA model [6] positions itself as a generalist VLA for multi-robot control, with evaluation centered on tabletop manipulation tasks and success rate. Recent VLAs demonstrate that coupling perception with action enables

safer driving decisions [8] and smoother, more stable robot trajectories [9] under complex conditions. Despite recent developments towards generalization over embodiments and tasks, fine-tuning as a form of transfer learning is often essential to run VLAs in new settings, as the domain gap is often too big [4, 6].

Recent surveys primarily organize progress around manipulation datasets, policies, and benchmarks, reflecting what the community most frequently measures and optimizes [10]. Another line of work applies VLA-style systems to autonomous driving [8, 11]. While most VLA-centered studies focus on manipulation, humanoid robot control, and mobile robot navigation tasks [12], other high-impact industrial challenges remain less explored. A key example is robotic inspection and quality assurance, a domain that is crucial for reliable and cost-effective manufacturing.

Inspection differs fundamentally from manipulation tasks: While the target of manipulation includes altering the scene, inspection requires perception, scene understanding, and generating motion that keeps sensors consistently aligned with the inspected feature without disturbing the scene. Traditional human-centered inspection is often labor-intensive, slow, and inconsistent. In contrast, robotic automation enables persistent perception, smart viewpoint selection, and repeatable motion along surfaces and features [13]. Accordingly, robotic automation can improve inspections across different tasks, geometries, and operating conditions [14, 15].

Robotic inspection research has traditionally evolved along largely separate methodological pipelines. Most approaches formulate inspection as view planning, coverage planning, or data acquisition planning [16], rather than as language-conditioned, end-to-end visuomotor policies. Such traditional methods often separately plan sensor poses or trajectories to maximize visibility or coverage under reachability, collision avoidance, and sensing constraints [17, 18]. These planning-based inspection approaches typically require substantial manual setup and domain-specific engineering, relying on structured priors such as known geometries, predefined viewpoints, or curated training data. Classical approaches to generate inspection paths are time-consuming and can require predefined inspection poses and the collection of training images, which becomes impractical when targets vary [19]. As pretraining data for inspection tasks is sparse, and data generation is expensive, data efficient methods and transfer learning from domains such as manipulation are attractive.

VLAs pretrained on large manipulation corpora offer a natural starting point, as they already encode visuomotor reasoning that can potentially be redirected toward inspection motion through targeted fine-tuning. Nevertheless, the domain gap between manipulation and inspection tasks makes this transfer non-trivial. This motivates studying whether manipulation-pretrained VLAs can be adapted for inspection tasks, reducing reliance on rigid, handcrafted routines.

Recent studies have started to explore related directions, particularly through the use of VLMs for inspection tasks [20–22]. Applications range from inspection of construction sites [23] to detecting defects on diverse items [24], utilizing the VLM's semantic scene understanding to perform tasks that previously required human inspectors.

More specialized approaches extend these ideas toward robotics. For example, Kilsby et al. [25] explore the use of VLMs for railway infrastructure inspection, while

[26] propose a VLA for bridge inspection in combination with an unmanned aerial vehicle. Similarly, Tasneem et al. [27] integrate an LLM into human–robot collaborative visual inspection, enabling natural-language interaction and flexible inspection planning.

However, the proposed systems are not demonstrating VLA use for inspection-style tasks, as they either do not realize end-to-end visuomotor policies or use hybrid approaches, where an LLM is used for high-level code generation and task sequencing, while perception, motion planning, and trajectory execution are handled by a classical robotic backend. As a result, the potential of VLAs for fully integrated, inspection-oriented control remains largely unexplored.

In addition, transferring VLAs from manipulation to inspection introduces new requirements, such as the need to incorporate further sensory inputs into the model. As the model must perceive the inspection target using sensors such as cameras or force sensors mounted on the end-effector, it is potentially beneficial to include these measurements directly in the model input. With manipulation-focused models often restricted to fixed input modalities such as one RGB input image of fixed size [6], the integration of additional sensory data becomes a promising direction.

Another challenge concerns VLA performance assessment: existing VLA evaluation protocols are tuned for manipulation tasks and are dominated by end-task success rates, occasionally allowing partial credit [4–7]. For manipulation tasks, the success rate can often be derived from changes in the environment. For inspection tasks, however, success cannot be defined based on environmental change. Instead, it can be assessed by the consistency of the recorded sensor data and the stability and completeness of the inspection trajectory. This concern aligns with broader critiques that attest success-based metrics limited diagnostic power as they miss execution quality, robustness, and confidence [28]. Therefore, an appropriate metric is needed to assess VLA performance in the inspection domain.

A more general challenge for current VLAs is the evaluation during the training process. VLAs are commonly trained with cross-entropy losses and monitored using action token accuracy, which assesses correctness at the level of discretized action tokens [6]. These metrics ignore the geometry of the continuous action space, treating physically similar or functionally valid actions as equally incorrect and failing to capture harmful directional errors. Consequently, the calculated values do not necessarily correlate with rollout performance, motivating direction-based vector metrics that better reflect action quality.

In accordance with the identified research gaps, this work pursues four objectives: (1) an inspection-aligned evaluation metric is proposed that captures observation completeness and trajectory quality, rather than relying solely on end-task success; (2) a metric for model quality assessment at the action level is introduced; (3) an inspection-oriented VLA pipeline is developed, aiming to reduce reliance on rigid, handcrafted inspection routines within the considered setting; (4) the approach is implemented and validated in a custom experimental setup designed to represent key aspects of inspection constraints.

Importantly, this work focuses on motion planning for inspection-style tasks, rather than on downstream data quality assessment, as accurate and repeatable motion is a

prerequisite for consistent data acquisition. Downstream data quality assessment, such as defect detection, is therefore considered a subsequent step and left for future work.

The remainder of this paper is organized as follows. First, the work packages conducted in this study are outlined. Afterwards, a trajectory-level metric to assess VLA performance in inspection tasks, as well as an action-level metric to evaluate the quality of individual predictions, are introduced. The subsequent part of this work investigates the real-world application of a VLA to an inspection-style task. To this end, an experimental setup and a VLA model are selected and applied to a task designed to capture core characteristics of inspection scenarios. A semiautomatic data generation pipeline is developed, and based on the collected data, three different models are fine-tuned, exploring alternative approaches for integrating a second camera as model input. Additionally, a human benchmark is established and compared against the three models using the proposed trajectory-level metric. Finally, the results are presented and discussed, followed by conclusions and directions for future work.

2 Methodology

2.1 Overview

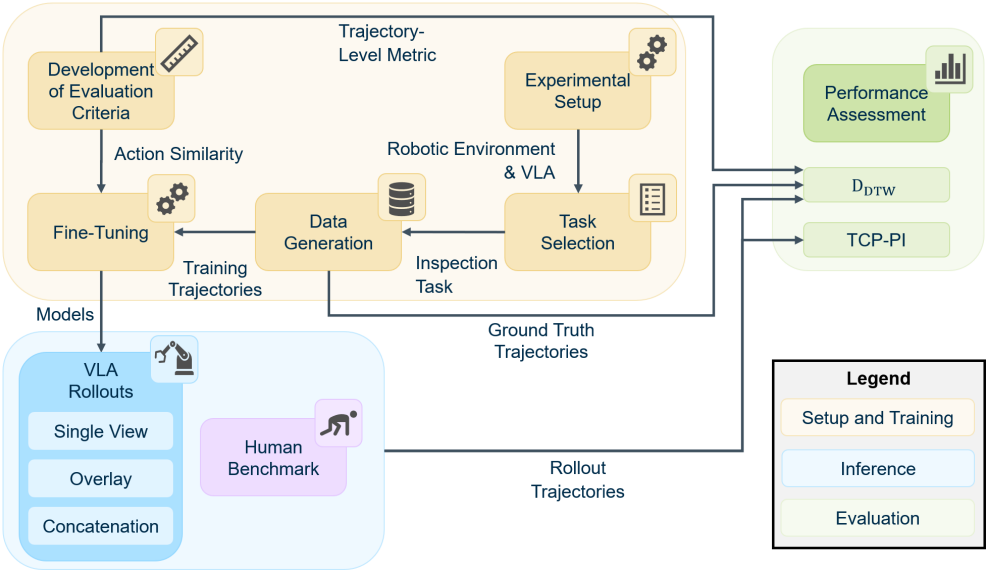

Fig. 1 Overview of the work packages and their interaction throughout this paper. The work packages include the entire path from creating an Experimental Setup to selecting a Task, Data Generation, and Fine-Tuning to assessing the performance of Rollouts to a Human Benchmark with the developed Trajectory-Level Metric. Additionally, a Human Benchmark is created for comparison, and an Action-Level Metric is developed to assist Fine-Tuning.

Figure 1 gives an overview of the work conducted. The real-world Experimental Setup comprises both the Robotic Environment and the selection of a VLA model used. A Task Selection is performed to yield a task representative for different inspection problems. Based on the selected Task, a semiautomatic system is implemented for Data Generation. A key methodological component of this work is the Development of Evaluation Criteria, where an Action-Level Metric is introduced to optimize Fine-Tuning and a Trajectory-Level Metric is introduced for the evaluation of inspection task performance. During Fine-Tuning, the manipulation-pretrained VLA is trained on the collected Training Trajectories, resulting in distinct Models. The Models explore different strategies for incorporating an additional camera stream as input, with the goal of improving performance on visual inspection tasks. The resulting policies, namely a Single View, Overlay, and Concatenation approach, are run in real-world rollouts to generate Rollout Trajectories. During the Performance Assessment, these trajectories are subsequently evaluated against Ground Truth Trajectories and a Human Benchmark using the proposed trajectory-level metric D_{DTW} as well as the Trajectory Position Instability $TCP - PI$ proposed in [28].

2.2 Trajectory Level Metric

To obtain a robust metric for VLA rollout quality, three complementary approaches can be considered: (1) modifying the success rate through task-specific thresholds for acceptable behavior; (2) evaluating the resulting measurement values received during inspection; (3) assessing the characteristics of the executed trajectory.

Modifying the success rate is challenging, as defining thresholds for all possible deviations is a tedious process. Evaluation based on measurement values faces similar challenges, as it requires defining thresholds for acceptable outcomes, while poor measurements can arise from factors unrelated to the executed motion. Therefore, VLA assessment based on the executed trajectory provides a more reliable approach and is proposed as a novel application of trajectory comparison methods, applicable whenever the ground truth trajectory can be defined. Following this idea, the quality of a rollout is measured by calculating the similarity between the ground truth trajectory and the executed trajectory. To compare trajectories in this manner, a suitable trajectory-based metric must (i) reward partial task completion while penalizing incomplete executions; (ii) be robust to variations in execution speed; (iii) distinguish temporally distinct but geometrically similar poses, while accounting for the full end-effector pose, including orientation.

Based on these requirements, Dynamic Time Warping (DTW) is used as the backbone for trajectory comparison in this work. DTW is a distance measure between two time series that optimally aligns their discrete time steps [29]. DTW naturally handles partial matches and speed variations. DTW also effectively captures scenarios where trajectories pass the same points multiple times. While standard DTW trajectory metrics typically only consider the positional difference [30, 31], it is possible to define custom distances between trajectory time steps [32], allowing orientation to be included. While DTW is a general method for time-series alignment and has been widely applied in robotics for trajectory comparison [32–35], most existing evaluation protocols for vision-language-action (VLA) systems rely on task success rates. In

this work, DTW is therefore adapted by defining a configuration-space distance that jointly captures translational and rotational deviations and by resampling trajectories to enable fair comparison across different lengths. The following section shows the exact DTW formulation utilized in this work.

To apply DTW to VLA trajectory comparison, our metric defines a configuration-space distance that jointly considers translational and rotational components of the manipulator, enabling a more meaningful comparison for inspection-relevant motions.

Two trajectories $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$ and $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ are compared, where each element $\mathbf{a}_i$ and $\mathbf{b}_j$ is a configuration of the robotic arm along the trajectory, defined as

$$\mathbf{a}_i = (\mathbf{p}_i, \mathbf{R}_i), \qquad \mathbf{b}_j = (\mathbf{p}_j, \mathbf{R}_j),$$

with position vectors $\mathbf{p}_i, \mathbf{p}_j \in \mathbb{R}^3$ and the rotation matrices are represented as $\mathbf{R}_i, \mathbf{R}_j \in \mathrm{SO}(3) \subset \mathbb{R}^{3\times3}$.[1] The distance between two configurations is defined as a weighted sum of translational and rotational differences. While multiple metrics exist to compare poses, the chosen formulation prioritizes transparency and interpretability and allows task-specific weighting. For instance, SE(3) logarithmic distances provide a unified but less transparent treatment of pose differences, and Quaternion-based distances introduce ambiguity requiring additional normalization [36, 37].

The Euclidean distance is used for the translational difference, and the rotational difference is measured using the geodesic distance on SO(3):

$$\mathrm{dist}(\mathbf{a}_i, \mathbf{b}_j) = w_t \, \|\mathbf{p}_i - \mathbf{p}_j\|_2 + w_r \, \theta(\mathbf{R}_i, \mathbf{R}_j),$$

$w_t, w_r > 0 \in \mathbb{R}$ are weighting factors and

$$\theta(\mathbf{R}_i, \mathbf{R}_j) = \arccos\left(\frac{\mathrm{tr}(\mathbf{R}_i^\top \mathbf{R}_j) - 1}{2}\right)$$

is the minimal rotation angle between the two orientations. The weighting factors w_t and w_r allow for balancing the different scales of position and rotation and are fixed prior to evaluation based on the geometry of the experimental setup, further described in Section 2.4.1. For a camera 0.2 m from the feature, a rotation of 15° produces approximately the same displacement in the image as a lateral shift of 0.054 m. The parameters are therefore selected as $w_t = 1$ and $w_r = 0.25$, which relates a rotation of about 10°–15° to a similar significance as a translation of about 0.04 to 0.06 m. This choice scales the rotation, measured in radians, to be comparable with the translational distance in meters.

While standard DTW formulations typically do not address alignment biases of different lengths [30, 31, 38], the longer trajectory is resampled to ensure a fair comparison. The trajectories $\mathbf{A}$ and $\mathbf{B}$ are ordered such that $m \leq n$. $\mathbf{B}$ is then evenly resampled to match the length of the shorter trajectory $\mathbf{A}$, resulting in $\mathbf{B}' = (\mathbf{b}'_1, \ldots, \mathbf{b}'_m)$ with m

[1] Only the end-effector pose is considered; the gripper state is not included, as it is not relevant for the inspection tasks considered in this work.

configurations. Resampling employs linear interpolation for positions and spherical linear interpolation (SLERP) [39] for orientations. Using the defined distance $dist(\mathbf{a}_i, \mathbf{b}_j)$ and two trajectories of equal length m, the DTW distance is computed as follows:

$$DTW(\mathbf{A}, \mathbf{B}') = \begin{cases} 0, & \text{if } \mathbf{A} \text{ and } \mathbf{B}' \text{ are empty,} \\ \infty, & \text{if } \mathbf{A} \text{ or } \mathbf{B}' \text{ is empty,} \\ dist^2(\mathbf{a}_1, \mathbf{b}'_1) + \min \left\{ \begin{matrix} DTW(\mathbf{A}_{2:m}, \mathbf{B'}_{2:m}), \\ DTW(\mathbf{A}, \mathbf{B}'_{2:m}), \\ DTW(\mathbf{A}_{2:m}, \mathbf{B}') \end{matrix} \right\}, & \text{otherwise,} \end{cases}$$

where $\mathbf{A}_{i:j}$ denotes the subsequence of trajectory $\mathbf{A}$ from index i to j inclusive. Finally, the calculated distance is normalized by the length of the trajectories m. The trajectory similarity measure, referred to as the trajectory alignment distance D_{DTW}, is then given by

$$D_{\mathrm{DTW}}(\mathbf{A}, \mathbf{B}) = \sqrt{\frac{\mathrm{DTW}(\mathbf{A}, \mathbf{B}')}{m}}. \tag{1}$$

In practice, values close to zero indicate high-quality execution, whereas larger values correspond to partial task completion or significant deviations from the reference trajectory.

2.3 Action-Level Metric

While the trajectory-level metric evaluates the quality of complete rollouts, it does not provide feedback on individual action predictions, which is crucial for guiding model fine-tuning. To address this limitation, a second metric is introduced that assesses the model performance at the level of singular 6D action vectors.[2] Important requirements for an action-level metric include sensitivity to all six components, the ability to capture combined positional and orientational errors, and interpretability of the resulting score. For action comparison, the direction of the movement is considered more relevant than its magnitude, as the same trajectory can be correctly followed with different speed profiles. The direction is unaffected by acceleration phases, sampling rates, and velocity profiles.

Common choices for comparing 6D vectors are the Mean Absolute Error (L1) and the Mean Squared Error (L2). While L1 provides a robust measure that is less sensitive to outliers, L2 penalizes larger deviations more strongly. Consequently, L2 is preferred if significant errors should carry more weight in the evaluation [40]. Another option for comparing vectors is cosine similarity, which compares the directional alignment of two vectors regardless of their magnitude. As cosine similarity can be applied to 6D vectors and yields an interpretable direction-based metric, it is selected for action-level assessment. While cosine similarity is widely used for vector comparison in domains such as essay assessment, genomic data analysis, and music similarity measurement [41–43], the application to the assessment of VLA action vectors is novel. The following section details the application of cosine similarity to the 6D action vectors: Both the

[2] While many VLAs predict a 7D action vector, the component of the gripper is not considered in this work, as it is not relevant to inspection tasks.

predicted and ground-truth action vectors, $\mathbf{v}_i^{\text{pred}}$ and $\mathbf{v}_i^{\text{true}} \in \mathbb{R}^6$, are decomposed into a translational component $\mathbf{t}_i = (\mathbf{v}_{i,1:3})$ and a rotational component $\boldsymbol{\gamma}_i = (\mathbf{v}_{i,4:6})$. Following Section 2.2, these components are weighted to account for their different scales.

$$\tilde{\mathbf{v}}_i = \begin{bmatrix} w_t \cdot \mathbf{t}_i \\ w_r \cdot \boldsymbol{\gamma}_i \end{bmatrix},$$

with $w_t = 1$ and $w_r = 0.25$.

$$\mathrm{CS}(\tilde{\mathbf{v}}_i^{\text{true}}, \tilde{\mathbf{v}}_i^{\text{pred}}) = \begin{cases} 1, & \tilde{\mathbf{v}}_i^{\text{true}} = \mathbf{0} \text{ and } \tilde{\mathbf{v}}_i^{\text{pred}} = \mathbf{0}, \\ -1, & \text{exactly one is } \mathbf{0}, \\ \dfrac{\tilde{\mathbf{v}}_i^{\text{true}} \cdot \tilde{\mathbf{v}}_i^{\text{pred}}}{\|\tilde{\mathbf{v}}_i^{\text{true}}\|_2 \|\tilde{\mathbf{v}}_i^{\text{pred}}\|_2}, & \text{otherwise.} \end{cases} \tag{2}$$

The angular deviation given as

$$\theta_i = \arccos(\mathrm{CS}) \tag{3}$$

is linearly scaled to $[0, 1]$ as

$$s_i = 1 - \frac{\theta_i}{\pi}, \tag{4}$$

The resulting score $s_i \in [0, 1]$ is directly interpretable: values close to 1 indicate strong directional agreement between predicted and ground-truth actions, $s_i = 0.5$ corresponds to orthogonal action directions, and values close to 0 indicate opposing directions. Over N steps s_i can be averaged to obtain the mean action similarity

$$\overline{S} = \frac{1}{N} \sum_{i=1}^{N} s_i. \tag{5}$$

Overall, this yields a method to assess the quality of single actions in a way that is both explainable and related to the usability of single action vectors.

2.4 VLA Adaptation to Inspection Task

2.4.1 Experimental Setup

To support the experimental investigations in this work, the real-world setup shown in Figure 2 is used for VLA training and inference. A Static RGB-D Camera (RealSense D435i) observes the scene featuring a Robotic Arm (Franka Panda). In front of the robot, a Workspace of 50 cm by 40 cm is used to conduct experiments. A Wrist Camera of the same type is mounted on the robot, allowing for close inspection of an Object within the Workspace by moving the end-effector. The robot and the cameras are connected to a Control PC. The Control PC communicates with a separate VLA PC, which is used to fine-tune the VLA and run model inference.

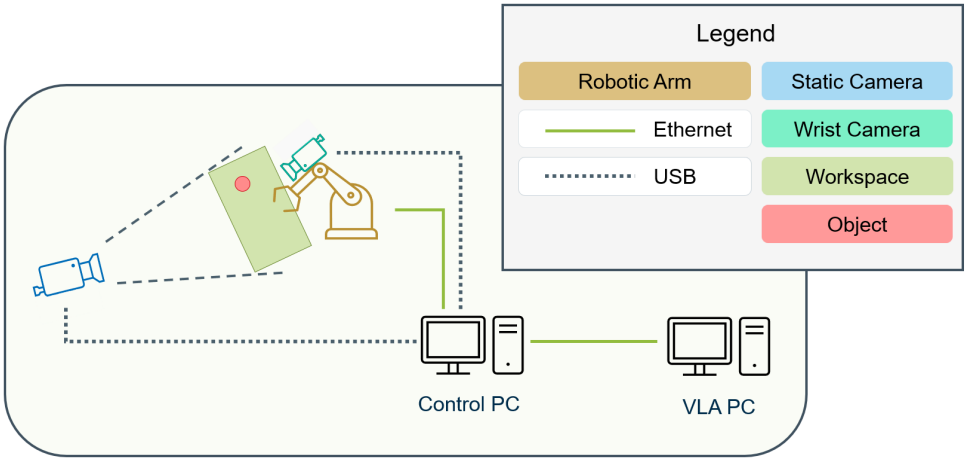

Fig. 2 The hardware setup used in this work consists of two PCs, a Static Camera, a Wrist Camera, and a Robotic Arm, which are connected via USB and Ethernet. The Robotic Arm and the cameras are used to generate data and to run inference on the model.

OpenVLA is used for the experiments in this work, as it provides a fully open-source code base including model weights, fine-tuning scripts, and examples for inference. OpenVLA can be fine-tuned to new tasks and environments by training on demonstrations from the target domain. Input to the model is a single image and a prompt. Based on the input, OpenVLA predicts 7D action vectors. The seventh dimension for the gripper is omitted in this work, and the 6D vector of position and rotation differences is considered. Training as well as action prediction is performed on the VLA PC. The predicted actions are then converted into executable movements and sent to the robot by a low-level C++ program [44] on the Control PC. Additional details on the system implementation and hardware are provided in Appendices B and C, respectively.

2.4.2 Task Design

To evaluate whether a manipulation-pretrained VLA can be adapted to inspection-relevant skills, this work selects a focused and experimentally tractable task isolating core challenges of inspection. Inspection tasks across domains such as manufacturing and infrastructure often involve elongated features, including cables, welds, and seams on technical textiles such as airbags. All these tasks share three distinct characteristics: The system has to (1) perceive the feature, (2) understand its geometry, and (3) generate motion that follows the trajectory while keeping the sensors aligned with the feature. Feature-following is therefore an excellent demonstration task to assess the capabilities of a VLA in the inspection domain: It requires understanding and adaptive action generation, while avoiding domain-specific sensor requirements. The task also enables generating flexible and reproducible experiments and evaluation by trajectory comparison.

Based on these considerations, this work introduces the task of following a cord from a start to an end position. Concretely, a red cord is placed in a random shape and position in a flat workspace. The goal is to follow the cord from one end to the other while keeping the cord centered in the wrist camera images and the distance between the camera and the cord constant. To make the task unambiguous, one end is marked with a green circle, and the other end is marked with a yellow square. The prompt for the VLA is formulated as: "Move along the red line from the yellow square to the green circle." Figure 3 illustrates the feature-following task used.

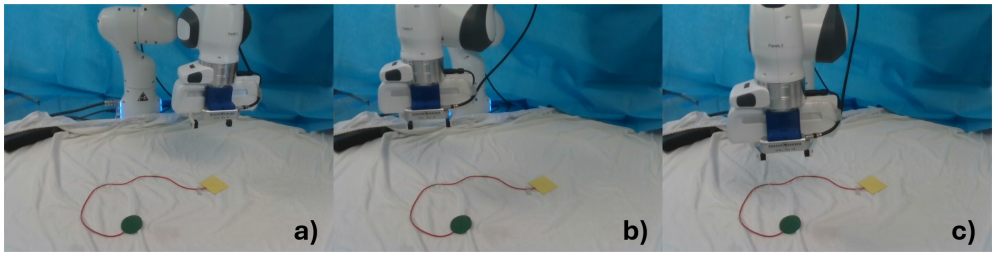

Fig. 3 Feature-following task. From (a) to (c), the images demonstrate different positions along the taken trajectory during the task execution. Starting from a position where the wrist camera is over the yellow marker, the end-effector has to follow the cord until it reaches the green end marker.

2.4.3 Data Collection

An approach is needed to generate a fine-tuning dataset and ground-truth trajectories for the assessment of rollout performance using the trajectory alignment distance D_{DTW}. Since teleoperation, which is often used to generate VLA training data, is slow and produces imperfect trajectories that are not well-suited as ground truth, the following semiautomatic approach is developed: Before recording a trajectory, the cord is placed in a random shape within the workspace, and the camera-equipped end-effector is positioned over the start point, marked by the yellow square. The wrist camera's optical axis is perpendicular to the workspace surface, and the camera lens is located 200 mm above the workspace ground plane. To follow the cord with the end-effector, the cord's shape is extracted from the wrist camera images, and the endpoints are identified using classical computer vision [45] (for more information, see Appendix A). Based on the extracted shape and its endpoints, 7D action vectors with only the Δx and Δy components having nonzero entries are calculated and used to move the end-effector along the cord trajectory in 2 cm increments. Based on the actions, the robot moves along the path of the cord. At the same time, images are collected from both cameras and associated with the actions for training. A total of 100 fine-tuning trajectories with different cord shapes and positions is collected for training. Additionally, ten more validation trajectories are recorded to provide validation actions for the training process. The cord shapes are manually arranged within the workspace, resulting in varied curvatures and spatial positions without enforcing specific constraints. Image augmentations provided by the OpenVLA framework are applied during fine-tuning.

2.4.4 Fine-Tuning

To control the fine-tuning process and select the best-performing model efficiently, early stopping is implemented based on the action similarity. Therefore, a set of $N = 30$ ground-truth actions with corresponding images is sampled from the unseen demonstrations and used as a validation dataset to assess the current quality of the model. The current model state is used to predict action vectors for the images in the validation dataset every 20 training steps. Based on the predicted and the ground-truth action vectors, the mean action similarity $\overline{S}$ is calculated. If $\overline{S}$ does not improve over 600 training steps, training is stopped.[3]

Following the requirement for additional sensory model input, the utilization of the wrist camera as VLA input is explored. As OpenVLA only supports a fixed-size input image, these approaches aim to expand the model's usability for multiple cameras without architectural changes or heavy retraining. In total, three models are fine-tuned based on the collected data.

- **Static Camera**
  Only the images from the static camera are used as the input image.
- **Overlay**
  The images from the wrist camera are combined with the static camera images by overlaying the two frames. The overlay image is created by averaging corresponding pixels across all RGB channels.
- **Concatenation**
  The images from the wrist camera are combined with the static camera images by horizontal concatenation and then rescaled to the model's input size.

2.4.5 Rollouts

To evaluate the usability of the adapted VLA on the inspection tasks, the performance is assessed in real-world rollouts using the defined trajectory alignment distance D_{DTW}. As reproducing the exact cord shape is difficult, all models, as well as the human baseline, are evaluated sequentially on the same three randomly generated cord shapes. Figure 4 depicts the three cord shapes and the corresponding starting positions of the robot. Each cord shape is tested three times, yielding a total of nine rollouts per method. For each cord shape, the semiautomatic following approach is used to generate a ground truth trajectory for the evaluation using D_{DTW}. Additionally, the rollout duration as well as the Trajectory Position Instability $TCP - PI$ proposed in [28] are used to assess the rollouts. $TCP - PI$ is based on the second-order differences of the positions. Lower $TCP - PI$ values correspond to smoother acceleration and consistent trajectory execution [28].

[3] The choice of 600 was derived from preliminary tests conducted beforehand, where it was shown to be a suitable value.

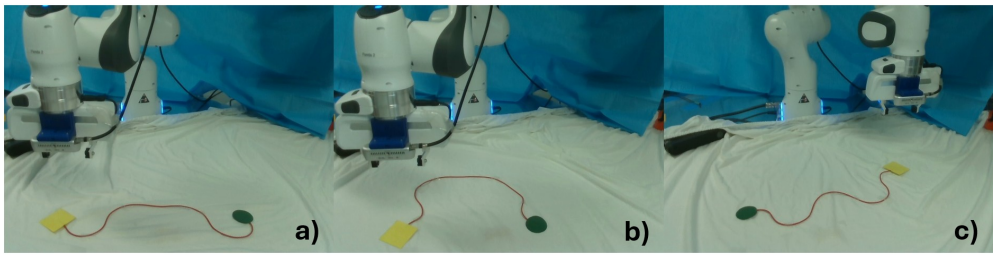

Fig. 4 Three cord shapes of the robot and the cord used for evaluation, each sub-figure shows a distinct shape of the feature and a different starting position of the end-effector.

2.4.6 Human Benchmark

Since complex inspection tasks of elongated features are traditionally performed by humans, benchmarking robotic performance against humans is natural. Therefore, an operator uses a camera to follow the cord on the same shapes presented in Section 2.4.5. To record human demonstrations, the operator manually moves the robotic arm in hand-guiding mode, where the robot passively follows externally applied forces without additional assistance. The end-effector positions along the executed trajectory are recorded. The benchmark employs a single operator with extensive prior experience using the robotic hardware, which allows us to establish a competitive benchmark rather than measuring broad performance over multiple users with varying skill levels. The operator is given three practice trials to follow the cord shapes before benchmark data are recorded on previously unseen shapes. The current images from the wrist camera are streamed to a screen to assist the operator with the movement. Additionally, the distance to the cord is displayed to assist the operator, providing additional information not available to the VLA models.

3 Results and Discussion

Figure 5 shows the best and worst feature-following performance across all rollouts for the evaluated VLA models as well as the human operator, quantified by the trajectory alignment distance D_{DTW}. The plots show both the Ground Truth trajectory as well as the executed Rollout. For clarity, only the position is shown. The gray lines indicate the configurations associated by D_{DTW}.

While the human-generated trajectories generally follow the ground truth with small deviations in all directions, the trained VLA models predict nonzero values solely for the Δx and Δy components, resulting in planar motion. The VLA-based rollouts demonstrate that all models except the concatenation-based approach successfully learn to follow the cord. The concatenation-based trajectories, however, exhibit substantial noise, with only the best rollout approximately following the correct direction. Both the static camera and overlay models consistently execute the task, with worst-case rollouts still moving in the correct direction, capturing the overall shape of the cord.

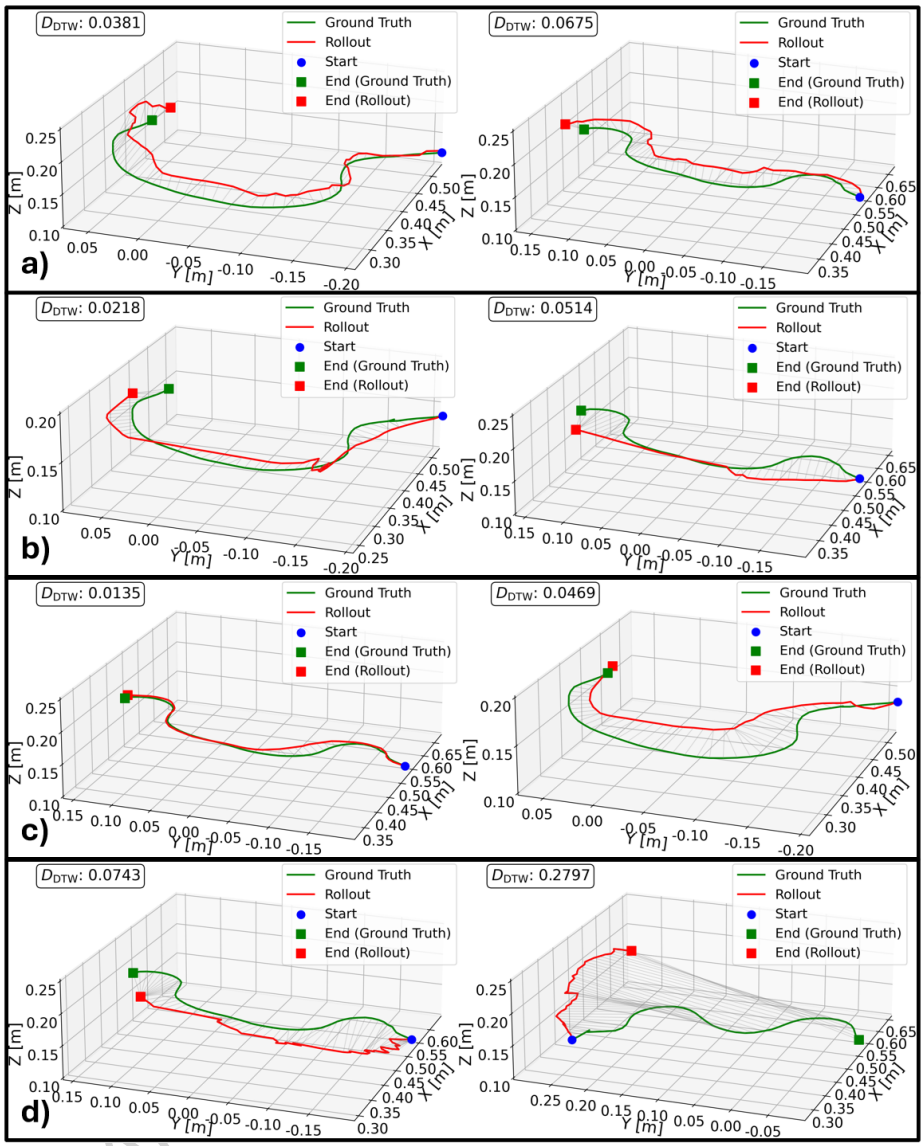

Fig. 5 Trajectories of best (left) and worst (right) rollouts according to the trajectory alignment distance D_{DTW}. From top to bottom: (a) human demonstrations, (b) VLA using the static camera input, (c) VLA with the overlay approach, and (d) VLA with the concatenation approach.

This qualitative observation from the best and worst rollouts is further supported by the quantitative results across all rollouts, shown in Figure 6, which reports the D_{DTW} values for the three VLA models and the human benchmark. The Mann–Whitney U test [46] was used to compute p-values for pairwise comparisons. To account for multiple comparisons, the Holm–Bonferroni correction [47] was applied separately for each metric. All pairwise comparisons between the four groups (human,

static camera, overlay, concatenation) were treated as a single family of tests, resulting in six comparisons. Both the static camera and overlay models follow the feature in the evaluated rollouts and achieve lower D_{DTW} scores than the human benchmark, with the overlay approach yielding the best results. The concatenation approach mostly fails to move in the correct direction, resulting in high D_{DTW} values. This poor performance likely arises from several compounding factors. The concatenation halves the resolution of both images, leading to smaller features. This input format differs significantly from the image distribution OpenVLA was trained on. In contrast to the overlay approach, which preserves the original image geometry, the concatenation approach requires the model to implicitly solve a problem that is not present during pretraining: distinguishing a meaningful second view from ordinary image content such as obstacles or background structure.

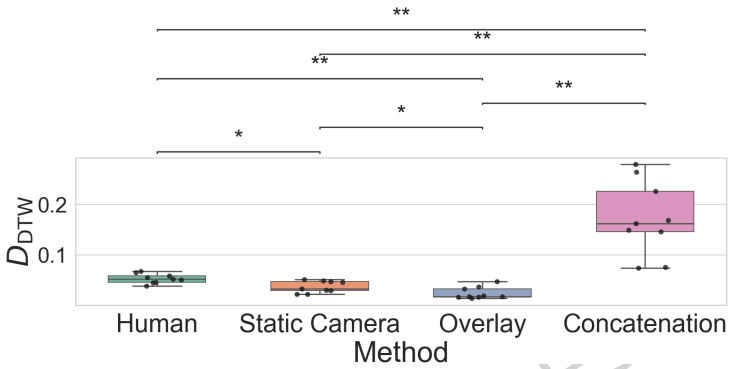

Fig. 6 Achieved trajectory alignment distance D_{DTW} for feature-following models benchmarked against human following. Each point depicts the score of a single rollout, where lower values indicate better trajectory alignment.

Figure 7 shows the performance quality measured by the Trajectory Position Instability $TCP - PI$. For the executed rollouts, the metric exhibits behavior consistent with the behavior observed with the D_{DTW} metric. The concatenation approach exhibits substantially higher values than all other approaches. While the differences between the groups of human, static camera, and overlay rollouts are comparable, the static camera approach shows a noticeably higher variance in $TCP - PI$, whereas both the human and the overlay approach demonstrate more consistent behavior. The overlay approach achieves a slightly lower median score than the human and static camera rollouts, but the differences are not statistically significant.

These results indicate that, in terms of trajectory smoothness as measured by TCP-PI, the static camera and overlay approaches achieve consistency comparable to human performance.

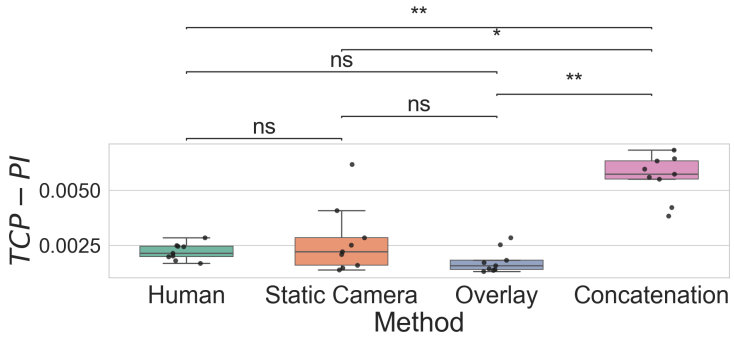

Fig. 7 Trajectory Position Instability $TCP - PI$ for all conducted rollouts. Lower values indicate smoother execution and superior performance.

Figure 8 shows the rollout duration for each execution to either successfully complete the task or terminate due to workspace violations or collisions. It can be seen that the VLA, following a static camera and overlay approach, executes the task in a similar time as the human operator. The small temporal difference between the slowest and the fastest overlay rollouts highlights the consistent behavior of this approach.

The rollout duration of the concatenation approach is similar; however, as most of these rollouts fail, direct comparisons are limited.

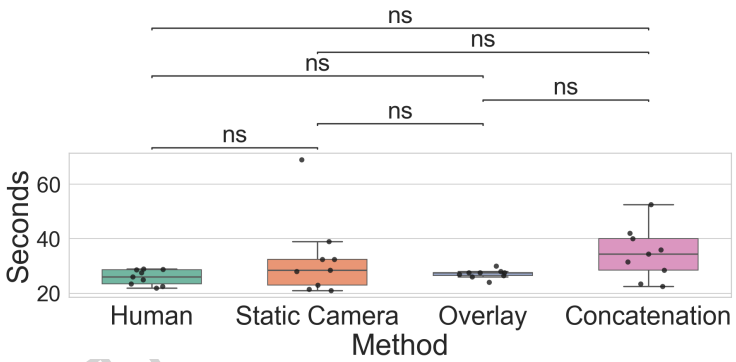

Fig. 8 Rollout duration of the individual rollouts. Each point depicts the time for one execution to either fail or complete the task.

Overall, the results demonstrate that OpenVLA can be fine-tuned to achieve lower trajectory-alignment errors than the benchmarked human operator in the evaluated feature-following rollouts, while maintaining comparable smoothness and execution times. This highlights that VLAs can be adapted to inspection-style tasks that require continuous, geometry-aware motion. This suggests that the learned representations are not limited to manipulation tasks but can be repurposed for tasks that emphasize spatial consistency and trajectory adherence without altering the environment. The strong performance of the overlay approach suggests that VLAs can effectively leverage additional sensor inputs without changing the architecture. By combining

global scene context with local geometries observed from the moving end-effector, the model gains complementary information assisting in feature recognition and trajectory alignment. In contrast, the poor concatenation performance indicates that additional sensor information has to be carefully merged to the input representation to use the potential of this sensor fusion approach.

While the evaluated task is limited to 2D feature-following, the findings suggest potential for transferring VLAs to richer inspection scenarios, such as three-dimensional inspection or problems involving more complex geometries. Such transfer, however, will likely require additional and diverse training data to ensure robustness under varying conditions.

While the human benchmark is based on a single operator, limited variability across repeated trials can be observed in D_{DTW}, $TCP - PI$, and rollout duration, even under controlled conditions with an experienced operator.

The key contribution of this work is a methodology for adapting manipulation-pretrained VLAs to inspection tasks. To support this, two metrics (trajectory alignment distance D_{DTW} and action similarity) are introduced, which provide an interpretable and quantitative assessment of VLA performance. These tools enable the development and evaluation of inspection VLAs, potentially reducing deployment time, improving model performance, and allowing precise performance assessment. This contribution is accompanied by a proof-of-concept on a focused experiment, including different sensor fusion strategies.

Moreover, the proposed metrics can also be applied to manipulation VLAs, offering a means for fine-grained performance analysis across different robotic tasks.

4 Conclusion and Future Work

Vision-Language-Action Models hold significant promise for automating a variety of tasks and have demonstrated strong generalization across different embodiments and tasks. However, VLAs have so far been primarily applied to manipulation tasks. While the inspection domain can potentially benefit from VLA capabilities, it has not yet been considered in research. Inspection poses unique challenges, meaning that models pretrained on manipulation data cannot be directly applied to inspection tasks. The most common metric for assessing VLA performance is task success rate. However, this metric is difficult to apply to inspection tasks: It does not allow fine-grained analysis and cannot capture whether the inspection trajectory was followed correctly.

To address these research gaps, this work introduced a methodology for the adaptation and evaluation of VLAs for inspection-oriented tasks. Two metrics were introduced that provide a novel perspective on VLA performance and enable the assessment of VLAs for inspection tasks: Our trajectory alignment distance D_{DTW} allows easy evaluation of VLA rollouts by comparison with ground-truth reference trajectories. The proposed action similarity offers an interpretable metric for evaluating individual actions by quantifying the directional differences between the predicted and ground-truth action vectors. Additionally, a focused feature-following task, isolating important challenges of inspection tasks, is proposed. A manipulation-pretrained VLA

was fine-tuned on inspection trajectories from a feature-following task. In the evaluated rollouts, the fine-tuned VLA achieved lower trajectory-alignment errors than the benchmarked human operator, demonstrating that a manipulation-pretrained VLA can be fine-tuned to a focused inspection task. However, due to the limited dataset size and number of evaluation rollouts, these findings should be interpreted as an initial proof-of-concept. Further large-scale studies are required to assess robustness, generalization, and transferability across a broader range of inspection scenarios. The results indicate that the VLA's performance on the inspection task can be significantly improved by overlaying two camera streams as model input, demonstrating that intelligent sensor fusion can enhance VLAs and accommodate additional sensors beyond the model's original design.

This study is limited to two-dimensional features; future work could extend the approach to three-dimensional inspection tasks, explore different inspection scenarios, and investigate the influence of various prompts. On the application side, integrating defect detection into VLA systems, either as a separate module or directly within the VLA model, represents a promising direction for further research.

Appendix A Computer Vision Cord Shape Extraction

To follow the cord with the end-effector as discussed in section 2.4.3, the cord's shape is extracted from the background on the wrist camera images, and the endpoints are identified using classical computer vision. Figure A1 shows the main image processing steps performed to extract the path of the cord for an example image. The figure illustrates the sequential transformation of the input image during processing. Each subfigure visualizes one intermediate step, beginning with the raw RGB input and ending with the extracted skeleton of the cord. First, the RGB image (subfigure (a)) is smoothed using Gaussian smoothing (subfigure (b)) to reduce potential noise or small irrelevant structures. Afterwards, the image is binarized to get an initial segmentation of foreground and background. To improve the robustness of the segmentation, two complementary binarization approaches are applied and subsequently combined. The first approach relies on a static threshold, computed from the mean value across all three RGB channels (subfigure (c)). The second approach exploits color information by defining a subspace of the HSV color space that corresponds to different shades of red (subfigure (d)). Both methods produce separate binary masks, which are then merged (subfigure (e)) so that all regions identified as potential candidates by either method are retained in the final binarized image. Morphological operations are applied to remove artifacts, such as the start and end markers, and to fill gaps in the line structure. Afterwards, the skeleton of the binarized cord is calculated (subfigure (f)). This reduces the cord to a one-pixel-wide centerline. The skeleton is then further processed to remove small branches, resulting in a single centerline. Based on the skeleton, the endpoints of the skeleton are detected.

The pipeline is implemented in Python using OpenCV and scikit-image. Gaussian smoothing is applied with a 21×21 kernel and $\sigma = 5$. The static binarization threshold is set to 85 (on the mean grayscale value), with automatic inversion if the background is lighter than the foreground. The HSV red segmentation uses two hue ranges: $H \in [0, 20]$ with $S \in [40, 255]$, $V \in [50, 255]$, and $H \in [150, 180]$ with $S \in [50, 255]$, $V \in [60, 255]$. Morphological cleanup consists of an initial closing with a 7×7 kernel followed by an opening with a 3×3 kernel, and a multi-scale gap-filling closing pass using kernels of widths 5, 10, and 15 pixels in horizontal, vertical, and diagonal orientations. Large blobs (markers) are suppressed by subtracting the result of a morphological opening with a 20×20 kernel. Skeletonization is performed via scikit-image's skeletonize, followed by iterative pruning of branches shorter than 45 pixels and removal of all connected components except the largest. Endpoints are detected by neighbor-counting and selecting pixels with exactly one neighbor. The system runs at 2 Hz on the Intel RealSense camera.

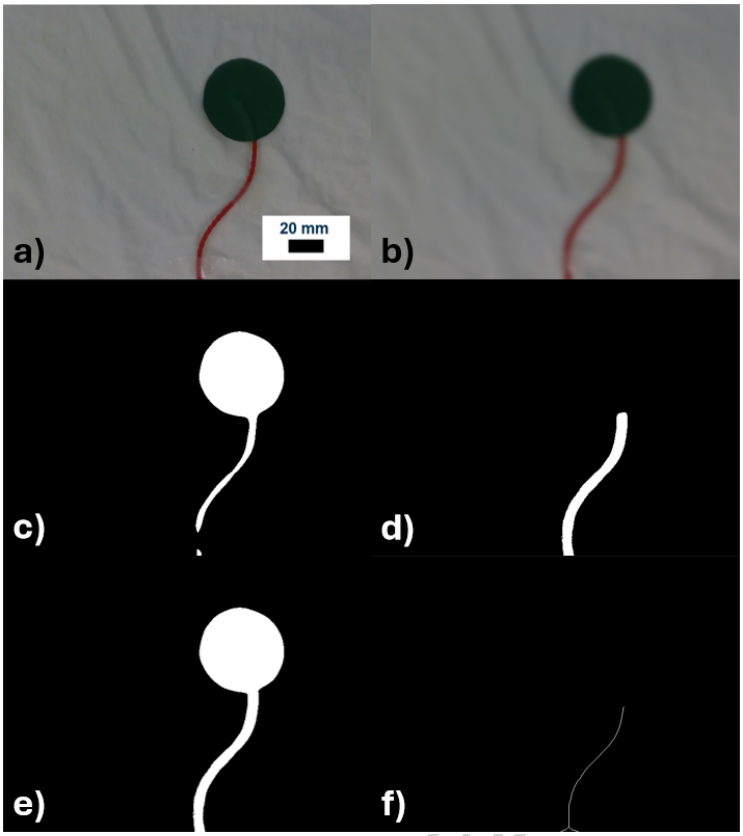

Fig. A1 The subfigures (a)–(f) show different steps in the image processing. (a) RGB image, (b) smoothed RGB image, (c) binarized image mean value across all three RGB channels, (d) binarized image based on the color in HSV format, (e) merged binary image of (c) and (d), and (f) the final skeleton.

Appendix B Implementation Details of the VLA Pipeline

Figure B2 illustrates the distributed execution pipeline of the proposed VLA robotic system, which is organized into three functional layers: `VLA_PC`, `Bridge`, and `Executor`. In the `VLA_PC` layer, camera images are captured, preprocessed, and passed to the OpenVLA model to predict a 7-dimensional action command. The predicted action is then serialized and transmitted over TCP, followed by an echo-based verification step to confirm communication integrity. In the `Bridge` layer, the received TCP message is echoed back to the sender and then converted into a file-based representation through a lock-controlled write operation to ensure safe synchronization between processes. In the `Executor` layer, the robot-side controller polls the shared file, reads the latest command, determines whether it corresponds to a joint-space or Cartesian

action, and executes the motion accordingly, including the associated gripper operation. Solid arrows indicate the forward data flow and file-synchronization path, whereas the dashed arrow denotes the return echo handshake used for transmission verification. The detailed sequence of operations across these three concurrent processes is summarized in Algorithm 1. In particular, the algorithm formalizes one complete action cycle, from visual inference and TCP-based transmission to lock-synchronized file exchange and final robot motion execution.

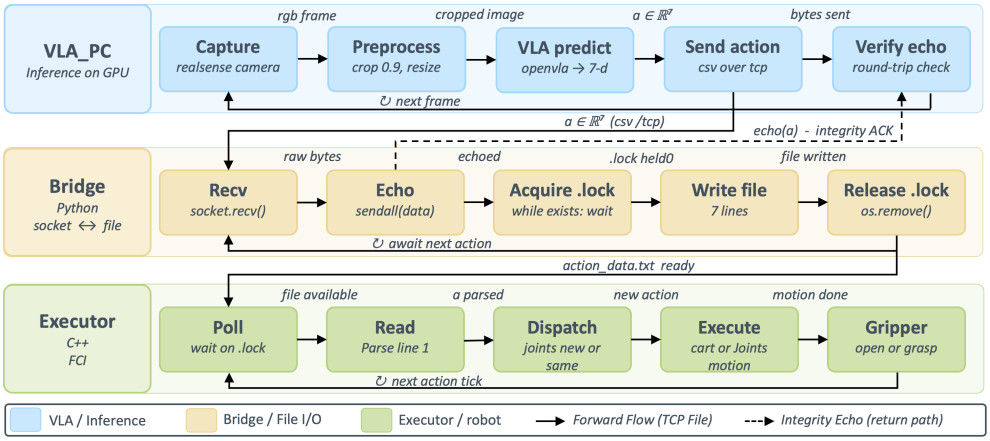

Fig. B2 Execution pipeline of a single VLA rollout cycle. Three concurrent processes: VLA inference on the GPU host, a Python socket-to-file bridge, and a C++ real-time executor are synchronized via TCP and a lock-guarded shared file.

Algorithm 1 End-to-end VLA rollout: one action cycle across three concurrent processes

Require: OpenVLA model *vla*, prompt p, camera $\mathcal{C}$; TCP endpoint (H, P); shared file F, lock L; Franka IP

procedure VLA_PC ▷ GPU host · Python
  $I \leftarrow \textsc{CenterCrop}(\textsc{Capture}(\mathcal{C}), 0.9)$ ▷ matches training aug
  $\mathbf{a} \leftarrow vla.\textsc{PredictAction}(p, I)$ ▷ $\mathbf{a} \in \mathbb{R}^7$: $(\Delta x, \Delta y, \Delta z, \Delta\psi, \Delta\theta, \Delta\phi, g)$
  $\mathbf{a}_{1:3} \mathrel{*}= \varepsilon, \quad \varepsilon \sim \mathcal{U}(0.999, 1.001)$ ▷ anti-dedup jitter
  $\textsc{TcpSend}\big(\textsc{Csv}(\mathbf{a})\big)$; **verify** round-trip echo within 1%
end procedure

procedure BRIDGE ▷ Control_PC · Python
  $d \leftarrow \textsc{TcpRecv}()$; $\textsc{SendAll}(d)$ ▷ echo handshake
  await $\neg\textsc{Exists}(L)$; $\textsc{Create}(L)$ ▷ acquire mutex
  $\textsc{WriteLines}\big(F, \textsc{Split}(d)\big)$; $\textsc{Remove}(L)$ ▷ release mutex
end procedure

procedure EXECUTOR ▷ Control_PC · C++ real-time; $\textsc{StateWriter}(S, 10\,\text{Hz})$ runs in parallel
  $\mathbf{a} \leftarrow \textsc{ReadTxtAction}(F, L)$
  if $\text{line}_1(F) =$ "joint" **then**
    $\textsc{JointMotion}(\mathbf{q}, 0.1)$; **return** ▷ teleport
  end if
  if $\|\mathbf{a} - \mathbf{a}_{\text{prev}}\|_\infty < \epsilon$ **then**
    return ▷ skip identical action
  end if
  $T_{\text{ee}} \leftarrow \textsc{CurrentPose}()$
  $\Delta\mathbf{p}_W \leftarrow R(T_{\text{ee}})\,[a_0, a_1, a_2]^\top$ ▷ local EE → base frame
  $\Delta R \leftarrow R_z(a_3)\,R_y(a_4)\,R_x(a_5)$
  $T^* \leftarrow \big(\,t(T_{\text{ee}}) + \Delta\mathbf{p}_W,\ \ R(T_{\text{ee}})\,\Delta R\,\big)$
  $\textsc{CartMotion}(T^*, 0.4\,\text{s})$
  $a_6 > 0.5$? $\textsc{GripperOpen}() : \textsc{GripperGrasp}()$ ▷ not relevant if following
  $\mathbf{a}_{\text{prev}} \leftarrow \mathbf{a}$
end procedure

Appendix C System Setup and Robot Kinematics

The Franka Emika Panda is typically represented as a seven-degree-of-freedom redundant serial manipulator using the modified Denavit-Hartenberg convention, in which the seven revolute joint variables q_1 through q_7 parameterize the arm configuration, while the constant geometric terms encode the robot's nominal link layout. In this formulation, the principal offsets and link lengths are given by $d_1 = 0.333$ m, $d_3 = 0.316$ m, $d_5 = 0.384$ m, $d_f = 0.107$ m, $a_4 = 0.0825$ m, $a_5 = -0.0825$ m, and

$a_7 = 0.088$ m. These constants define the successive homogeneous transformations between neighboring coordinate frames along the kinematic chain, thereby providing the geometric foundation for forward kinematics, Jacobian derivation, singularity analysis, and inverse kinematics. Owing to its redundant 7-DOF structure, the Panda can achieve a desired end-effector pose with additional flexibility in posture selection, which is particularly useful for dexterous motion generation, obstacle avoidance, and null-space optimization in constrained manipulation tasks.

For control implementation on the Franka Emika Panda, the Franka Control Interface is designed for a 1 kHz real-time control loop. According to the official Panda limits, the maximum joint velocities are $\dot{q}_{\max} = 2.175$ rad/s for joints 1-4 and $\dot{q}_{\max} = 2.610$ rad/s for joints 5-7. In Cartesian space, the maximum translational velocity is $\dot{p}_{\max} = 1.7$ m/s.

For camera calibration, our setup utilizes the intrinsic calibration of the Intel RealSense D435 camera. It is defined using the factory-provided parameters and exposed through the Intel RealSense SDK.

Supplementary information. Not applicable for this manuscript.

Acknowledgment. We would like to acknowledge the support provided by the KIT-Publication Fund of the Karlsruhe Institute of Technology.

Declarations

- Funding
  The publication of this work was supported by the KIT-Publication Fund of the Karlsruhe Institute of Technology.
- Conflict of Interest and Competing Interests
  All authors confirm that there are no conflict of interest and/or competing interests.
- Ethics Approval and Consent to Participate
  All authors approve and consent to participate in this manuscript.
- Consent for Publication
  All authors read the final version of the manuscript and approve to publish this work.
- Data Availability
  Data is available upon request.
- Materials
  All materials are available upon request.
- Code Availability
  The code is available upon request.
- Author Contribution
  Martin Krüger: Conceptualization, Methodology, Investigation, Validation, Visualization, Writing Original Draft, Reviewing, and Editing. Mahmoud Salem: Supervision, Conceptualization, Methodology, Validation, Writing Original Draft, Reviewing, and Editing. Markus Reischl: Supervision, Project Administration, Conceptualization, Validation, Proofreading, Reviewing, and Editing.

References

[1] Nambiar, S., Wiberg, A., Tarkian, M.: Automation of unstructured production environment by applying reinforcement learning. Frontiers in Manufacturing Technology **3** (2023) https://doi.org/10.3389/fmtec.2023.1154263

[2] Dhanda, M., Rogers, B.A., Hall, S., Dekoninck, E., Dhokia, V.: Reviewing human-robot collaboration in manufacturing: Opportunities and challenges in the context of industry 5.0. Robotics and Computer-Integrated Manufacturing **93**, 102937 (2025) https://doi.org/10.1016/j.rcim.2024.102937

[3] Mukherjee, D., Gupta, K., Chang, L.H., Najjaran, H.: A survey of robot learning strategies for human-robot collaboration in industrial settings. Robotics and Computer-Integrated Manufacturing **73**, 102231 (2022) https://doi.org/10.1016/j.rcim.2021.102231

[4] Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., et al.: Octo: an open-source generalist robot policy. arXiv preprint (2024). https://doi.org/10.48550/arXiv.2405.12213

[5] Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., et al.: pi_0: A Vision-Language-Action Flow Model for General Robot Control. arXiv preprint (2024). https://doi.org/10.48550/arXiv.2410.24164

[6] Kim, M.J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., et al.: OpenVLA: An Open-Source Vision-Language-Action Model. arXiv preprint (2024). https://doi.org/10.48550/arXiv.2406.09246

[7] Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., et al.: RT-2: Vision-language-action models transfer web knowledge to robotic control. arXiv preprint (2023). https://doi.org/10.48550/arXiv.2307.15818

[8] Zhou, X., Han, X., Yang, F., Ma, Y., Knoll, A.C.: OpenDriveVLA: towards end-to-end autonomous driving with large vision language action model. arXiv preprint (2025). https://doi.org/10.48550/arXiv.2503.23463

[9] Jiang, A., Gao, Y., Sun, Z., Wang, Y., Wang, J., Chai, J., et al.: DiffVLA: vision-language guided diffusion planning for autonomous driving. arXiv preprint (2025). https://doi.org/10.48550/arXiv.2505.19381

[10] Sapkota, R., Cao, Y., Roumeliotis, K.I., Karkee, M.: Vision-language-action models: concepts, progress, applications and challenges. arXiv preprint (2025). https://doi.org/10.48550/arXiv.2505.04769

[11] Jiang, S., Huang, Z., Qian, K., Luo, Z., Zhu, T., Zhong, Y., et al.: A survey on vision-language-action models for autonomous driving. arXiv preprint (2025). https://doi.org/10.48550/arXiv.2506.24044

[12] Kawaharazuka, K., Oh, J., Yamada, J., Posner, I., Zhu, Y.: Vision-language-action models for robotics: A review towards real-world applications. IEEE Access **13**, 162467–162504 (2025) https://doi.org/10.1109/ACCESS.2025.3609980

[13] Chen, Z., Foong, S., Chen, X., Ghorbel, F., Tang, J.: Focused section on robotics enabled inspection. International Journal of Intelligent Robotics and Applications **9** (2025) https://doi.org/10.1007/s41315-025-00493-9

[14] Khan, A., Mineo, C., Dobie, G., MacLeod, C., Pierce, G.: Vision guided robotic inspection for parts in manufacturing and remanufacturing industry. Journal of Remanufacturing **11**(1), 49–70 (2020) https://doi.org/10.1007/s13243-020-00091-x

[15] Liu, J., Wu, Y., Wu, Y., Wang, Y., Jia, X.: A novel laser vision-based method for robotic curved welding seam tracking. Measurement Science and Technology **36**(4), 046203 (2025) https://doi.org/10.1088/1361-6501/adbc0a

[16] Eckholdt Andersen, R., Yding Brogaard, R., Boukas, E.: Remote inspection techniques: A review of autonomous robotic inspection for marine vessels. IEEE Transactions on Field Robotics **2**, 1–20 (2025) https://doi.org/10.1109/TFR.2024.3499916

[17] Liu, Y., Zhao, W., Liu, H., Wang, Y., Yue, X.: Coverage path planning for robotic quality inspection with control on measurement uncertainty. IEEE/ASME Transactions on Mechatronics **27**(5), 3482–3493 (2022) https://doi.org/10.1109/TMECH.2022.3142756

[18] Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robotics and Autonomous Systems **61**(12), 1258–1276 (2013) https://doi.org/10.1016/j.robot.2013.09.004

[19] Tasneem, O., Pieters, R.: Automatic robot path planning for visual inspection from object shape. arXiv preprint (2023). https://doi.org/10.48550/arXiv.2312.02603

[20] Ding, Y.: Die-to-prompt: visual language model-based defect inspection and anomaly detection. In: Metrology, Inspection, and Process Control XXXIX, vol. 13426, pp. 483–487. SPIE, San Jose, CA, USA (2025). https://doi.org/10.1117/12.3052174

[21] Lin, S., Wang, C., Ding, X., Wang, Y., Du, B., Song, L., et al.: A VLM-based method for visual anomaly detection in robotic scientific laboratories. arXiv preprint (2025). https://doi.org/10.48550/arXiv.2506.05405

[22] Fan, H., Liu, C., Janvisloo, N.E., Bian, S., Fuh, J.Y.H., Lu, W.F., *et al.*: MaViLa: Unlocking new potentials in smart manufacturing through vision language models. Journal of Manufacturing Systems **80**, 258–271 (2025) https://doi.org/

10.1016/j.jmsy.2025.02.017

[23] Adil, M., Lee, G., Gonzalez, V.A., Mei, Q.: Using vision language models for safety hazard identification in construction. arXiv preprint (2025). https://doi.org/10.48550/arXiv.2504.09083

[24] Ueno, S., Hayashi, Y., Nakatsuka, S., Yamada, Y., Aizawa, H., Kato, K.: Vision-language in-context learning driven few-shot visual inspection model. arXiv preprint (2025). https://doi.org/10.48550/arXiv.2502.09057

[25] Kilsby, P., Kun, L.A.: Enabling intelligent robotic visual inspection in the railway industry with generative AI. In: 2024 Eighth IEEE International Conference on Robotic Computing (IRC), pp. 275–277. IEEE, Tokyo, Japan (2024). https://doi.org/10.1109/IRC63610.2024.00051

[26] Chen, Z., Zou, Y., González, V.A., Ingham, J., Wotherspoon, L.M.: Bridge inspection using a multi-modal vision language model. In: Proceedings of the 6th International Conference on Civil and Building Engineering Informatics, Hong Kong (2025). https://doi.org/10.29007/m7wj

[27] Tasneem, O., Pieters, R.: Human–robot collaborative visual inspection with large language models **98**, 103154 (2026) https://doi.org/10.1016/j.rcim.2025.103154

[28] Valle, P., Lu, C., Ali, S., Arrieta, A.: Evaluating uncertainty and quality of visual language action-enabled robots. arXiv preprint (2025). https://doi.org/10.48550/arXiv.2507.17049

[29] Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition **26**(1), 43–49 (1978) https://doi.org/10.1109/TASSP.1978.1163055

[30] Zang, Y., Wang, P., Zha, F., Guo, W., Li, C., Sun, L.: Human skill knowledge guided global trajectory policy reinforcement learning method. Frontiers in Neurorobotics **18** (2024) https://doi.org/10.3389/fnbot.2024.1368243

[31] Tao, Y., Both, A., Silveira, R.I., Buchin, K., Sijben, S., Purves, R.S., et al.: A comparative analysis of trajectory similarity measures **58**(5), 643–669 (2021) https://doi.org/10.1080/15481603.2021.1908927

[32] Vaughan, N., Gabrys, B.: Comparing and combining time series trajectories using dynamic time warping. Procedia Computer Science **96**, 465–474 (2016) https://doi.org/10.1016/j.procs.2016.08.106

[33] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., *et al.*: Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 262–270. Association for

Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2339530.2339576

[34] Mueller, C., Venicx, J., Hayes, B.: Robust robot learning from demonstration and skill repair using conceptual constraints. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6029–6036 (2018). https://doi.org/10.1109/IROS.2018.8594133

[35] De Lazzari, D., Terreran, M., Giacomuzzo, G., Jain, S., Falco, P., Carli, R., *et al.*: PACE: Proactive assistance in human-robot collaboration through action-completion estimation. In: 2025 IEEE International Conference on Robotics and Automation (ICRA), pp. 6725–6731 (2025). https://doi.org/10.1109/ICRA55743.2025.11127399

[36] Park, F.C.: Distance metrics on the rigid-body motions with applications to mechanism design **117**(1), 48–54 (1995) https://doi.org/10.1115/1.2826116 . Accessed 2025-08-16

[37] Huynh, D.Q.: Metrics for 3D Rotations: Comparison and Analysis. Journal of Mathematical Imaging and Vision **35**(2), 155–164 (2009) https://doi.org/10.1007/s10851-009-0161-2

[38] Sousa, R.S.D., Boukerche, A., Loureiro, A.A.F.: Vehicle trajectory similarity: Models, methods, and applications **53**(5), 1–32 (2021) https://doi.org/10.1145/3406096

[39] Shoemake, K.: Animating rotation with quaternion curves. In: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '85, pp. 245–254. ACM Press, San Francisco (1985). https://doi.org/10.1145/325334.325242

[40] Chai, T., Draxler, R.R.: Root mean square error (RMSE) or mean absolute error (MAE)? – arguments against avoiding RMSE in the literature **7**(3), 1247–1250 (2014) https://doi.org/10.5194/gmd-7-1247-2014

[41] Lahitani, A.R., Permanasari, A.E., Setiawan, N.A.: Cosine similarity to determine similarity measure: Study case in online essay assessment. In: 2016 4th International Conference on Cyber and IT Service Management, pp. 1–6 (2016). https://doi.org/10.1109/CITSM.2016.7577578

[42] Xie, J., Wang, M., Xu, S., Huang, Z., Grant, P.W.: The unsupervised feature selection algorithms based on standard deviation and cosine similarity for genomic data analysis **12** (2021) https://doi.org/10.3389/fgene.2021.684100

[43] Sheikh Fathollahi, M., Razzazi, F.: Music similarity measurement and recommendation system using convolutional neural networks **10**(1), 43–53 (2021) https://doi.org/10.1007/s13735-021-00206-5

[44] Salem, M., Elkaseer, A., Müller, T., Scholz, S.: On the development of a reconfigurable platform for the control of multiple collaborative robots from a software engineering perspective. In: Arai, K. (ed.) Proceedings of the Future Technologies Conference (FTC) 2023, Volume 1, pp. 74–91. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-47454-5_6

[45] Salem, M., Elkaseer, A., Müller, T., Scholz, S.: Intelligent collaborative robots augmented with vision perception for flexible manufacturing system. In: Jolly, M., Scholz, S.G., Howlett, R.J., Setchi, R. (eds.) Sustainable Design and Manufacturing 2024, pp. 67–78. Springer, Singapore (2025). https://doi.org/10.1007/978-981-96-4459-9_7

[46] MacFarland, T.W., Yates, J.M.: Mann–whitney u test. In: Introduction to Nonparametric Statistics for the Biological Sciences Using R, pp. 103–132. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30634-6_4

[47] Holm, S.: A Simple Sequentially Rejective Multiple Test Procedure. Scandinavian Journal of Statistics **6**(2), 65–70 (1979) https://www.jstor.org/stable/4615733