Demonstration-Based Visual Anomaly Detection in Manufacturing Factory Using Vision-Language Models

Huimin Zhuge[1], Song Wang[2], Huijuan Shao[3], Brian Chien[4], and Dipanjan Ghosh[5]

[1,2,3,4,5] *Hitachi America Ltd, Santa Clara, California, USA*
joy.zhuge@hal.hitachi.com
song.wang@hal.hitachi.com
huijuan.shao@hal.hitachi.com
brian.chien@hal.hitachi.com
dipanjan.ghosh@hal.hitachi.com

ABSTRACT

Flexible anomaly detection is important for identifying process deviations in manufacturing, yet conventional approaches typically depend on hand-coded inspection rules, annotated datasets, task-specific model training, or additional sensing infrastructure, all of which are costly to configure and difficult to adapt. This work investigates demonstration-based visual anomaly detection using vision-language models (VLMs), in which monitoring behavior is specified by example rather than by explicit programming. Instead of enumerating normal operating rules, the model is shown images and videos of nominal operation and infers the expected visual patterns directly from these demonstrations. During monitoring, new observations are assessed against the demonstrated patterns to detect potential abnormal states and to generate interpretable warnings and corrective suggestions. We formalize the detection problem mathematically and evaluate the approach on three representative fault types: sorting faults, where a color category is missing or misplaced; storage faults, where caps violate the expected bottom-to-top filling pattern or appear in an incorrect color region; and conveyor faults, where caps overlap, become stuck, move irregularly, or lose consistent spacing. These scenarios are realized on a small manufacturing cell equipped with pickup arms, a conveyor belt, sorting and storage units, and fixed monitoring cameras, using colored cylinder caps as representative workpieces. Results indicate that visual demonstrations combined with reasoning can substantially reduce the effort required to configure perception, anomaly interpretation, and decision support in manufacturing workflows.

1. INTRODUCTION

Manufacturing systems increasingly rely on visual monitoring to support process supervision, quality inspection, and fault detection. In production environments, even simple deviations such as misplaced parts, blocked material flow, or incorrect assembly order can interrupt operation and require human intervention. However, many existing inspection and monitoring approaches require task-specific models, manually defined rules, annotated datasets, or dedicated sensing infrastructure, which can limit their flexibility in small-scale or rapidly changing manufacturing settings.

Recent advances in vision-language models (VLMs) suggest a potential alternative for configurable visual monitoring. Unlike conventional vision models trained for a fixed set of labels, VLMs can reason jointly over images, videos, and natural-language descriptions. This capability may be useful for manufacturing cells where the expected behavior is difficult to fully encode in advance, but can be demonstrated through examples of normal operation. Rather than explicitly programming every inspection rule, a monitoring system could provide the model with representative nominal images and videos, then ask it to identify whether a new observation appears consistent with the demonstrated process.

This work explores this idea using an industrial testbed designed to represent common manufacturing operations. The testbed includes multiple pickup arms, a conveyor belt, sorting components, storage modules, and fixed cameras for workspace monitoring. Colored cylinder caps are used as representative workpieces with visually distinguishable classes. The goal is to investigate whether visual demonstrations and VLM-based reasoning can support low-effort configuration of anomaly detection tasks in a controlled manufacturing cell.

We study three representative monitoring scenarios. In the sorting component, the expected pattern is that different cap

colors are correctly represented and placed in their corresponding regions; missing colors or misplaced caps indicate potential faults. In the storage component, the expected pattern is that caps are filled from bottom to top within the appropriate color region; filling an upper position first or placing a cap in the wrong region indicates an abnormal state. On the conveyor belt, caps are expected to move with approximately regular spacing and steady motion; overlap, blockage, irregular spacing, or stuck caps indicate possible process deviations.

The proposed setup allows normal operating patterns to be introduced through images and videos rather than through explicit rule programming or retraining. Through the three case studies, we examine how VLM-assisted perception and reasoning can be used to identify abnormal visual states and generate human-readable warnings or corrective suggestions.

2. RELATED WORK

2.1. Visual Inspection and Anomaly Detection in Manufacturing

Visual inspection has been widely studied for manufacturing quality control, assembly verification, and process monitoring. Traditional approaches often rely on handcrafted features, template matching, thresholding, or geometric constraints to detect deviations from a predefined nominal state (Newman & Jain, 1995). More recent methods use deep learning models for defect classification, object detection, segmentation, and anomaly detection (Chibani, Coudert, & Kamsu-Foguem, 2022; Bergmann, Fauser, Sattlegger, & Steger, 2019). These methods can achieve strong performance when sufficient training data are available, but they often require annotated examples, task-specific model tuning, or carefully controlled imaging conditions.

Visual anomaly detection is relevant to manufacturing because abnormal visual patterns can indicate process faults, incorrect system states, or quality issues. Common visual anomalies include missing parts, incorrect placement, unexpected object combinations, blocked material flow, and irregular motion. Many approaches model normal appearance or behavior and then detect deviations from that model (Pang, Shen, Cao, & Hengel, 2021). However, configuring these systems for new tasks can still require data collection, annotation, and model adaptation. This motivates approaches that reduce the effort needed to specify normal operation.

2.2. Demonstration-Based and Reference-Guided Monitoring

Reference-guided monitoring compares a current observation with a known nominal state. This idea appears in template-based inspection, change detection, and robotic scene verification, where a reference image or model is used to determine whether objects are missing, misplaced, or incorrectly arranged (Radke, Andra, Al-Kofahi, & Roysam, 2005; Kragic & Christensen, 2002). Such methods can be effective when the workspace is structured and the desired configuration is known. However, they may be sensitive to viewpoint changes, occlusion, lighting variation, and object appearance changes.

Demonstration-based monitoring extends this idea by using examples of correct operation rather than a single explicitly encoded rule set. In robotics and manufacturing, demonstrations have been used to define desired trajectories, task constraints, and normal process behavior (Argall, Chernova, Veloso, & Browning, 2009). For visual monitoring, demonstrations can provide examples of acceptable spatial arrangements or temporal patterns. This is useful in small or reconfigurable manufacturing cells, where new products or tasks may be introduced frequently and manual rule engineering can become burdensome.

2.3. Vision-Language Models for Industrial Reasoning

Vision-language models have shown increasing ability to interpret images and videos using natural-language prompts, perform visual question answering, describe scenes, and reason about object relationships (Zhang, Huang, Jin, & Lu, 2024; Radford et al., 2021; OpenAI, 2023). These capabilities have encouraged early exploration of VLMs for robotics, inspection, planning, and human-machine interaction (Ahn et al., 2022; Xia, Li, Li, & Song, 2024). In manufacturing, VLMs are attractive because they may combine visual perception with semantic reasoning, allowing users to ask questions about a scene or request explanations of abnormal conditions.

Despite this potential, VLMs also have limitations. Their outputs can be sensitive to prompt design, image quality, viewpoint, and ambiguity in the scene. They may also produce uncertain or inconsistent responses when fine-grained spatial reasoning or temporal tracking is required (Guan et al., 2023). Therefore, VLM-based industrial monitoring should be studied carefully in controlled settings before deployment in safety-critical applications. The testbed used in this work provides such a setting, allowing different visual fault cases to be examined while keeping the physical process interpretable and repeatable.

2.4. Manufacturing Monitoring and Decision Support

Visual anomaly detection can support process monitoring, post-maintenance validation, assembly correctness inspection, and operator decision support. A system that can recognize deviations from demonstrated normal operation may help reduce configuration effort for small manufacturing cells, laboratory platforms, or educational testbeds. Rather than focusing only on high-accuracy defect classification,

this work emphasizes interpretable monitoring: identifying what appears abnormal, where the abnormality occurs, and what corrective action may be appropriate. This supports the broader goal of improving system awareness and enabling faster response to abnormal operating conditions.

3. METHODOLOGY

In this section, we first describe the manufacturing testbed setup. We then formulate the monitoring task as normal-demonstration-conditioned anomaly detection and fix the notation used throughout the paper. Finally, we describe each scenario in turn, including cap sorting, cap storage, and belt transport, giving for each the operating rules that define normal behavior and the perception pipeline that extracts the corresponding state from images or video.

3.1. Manufacturing Testbed

The experimental platform is implemented using a fischertechnik factory simulation system, as shown in Figure 1. The setup integrates several modular manufacturing components, including a vacuum gripper robot, an automated high-bay warehouse, and a sorting line. Together, these components form a small-scale manufacturing cell that reproduces key operations commonly found in automated production systems, such as material handling, processing, sorting, conveying, and storage. In a nominal workflow,

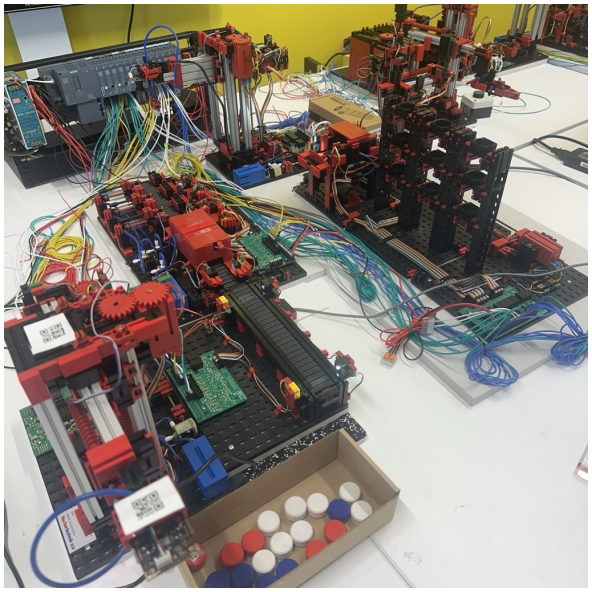


Figure 1. Miniature manufacturing factory testbed.

workpieces are first retrieved from a box by the vacuum gripper robot and transferred to the transport belts. The workpieces are then transported to the sorting line, where they are sorted according to color. After processing, a second gripper robot inserts each sorted workpiece into a black socket carrier and moves it to the automated high-bay warehouse for storage.

Colored cylinder caps are used as simplified workpieces, where each color represents a distinct object class. Although the testbed is smaller and less complex than a production-scale manufacturing system, it provides a controlled environment for studying visual monitoring across object placement, spatial arrangement, and motion-related conditions.

In this work, the manufacturing process is divided into three monitored stations: sorting, storage, and conveyor transport. The sorting station organizes caps according to their visual class. The storage station arranges caps in vertical slots or regions, where correct operation follows a consistent filling pattern. The conveyor station transports caps along the belt with expected spacing and motion. These stations are selected because they require different forms of visual reasoning, including object presence and color consistency, spatial ordering, and temporal motion regularity.

3.2. Problem Formulation

We formulate the monitoring task as few-shot, normal-demonstration-conditioned anomaly detection.

Let the full event space be partitioned into a set of normal events

$$\mathcal{N} = \{n_1, n_2, \ldots, n_P\}$$

and a set of abnormal events

$$\mathcal{M} = \{m_1, m_2, \ldots, m_Q\},$$

where each event may be an image, a video clip, or a short sequence of observations from a manufacturing process. A query event $e \in \mathcal{N} \cup \mathcal{M}$ may belong to either set.

Rather than requiring access to the full normal distribution $\mathcal{N}$, our approach operates on a small demonstration subset

$$\mathcal{N}' = \{n_1, n_2, \ldots, n_k\} \subset \mathcal{N}, \quad k \ll P,$$

where k is a small number of normal reference examples.

From this subset, a reasoning model infers a latent normal rule

$$R = g_\theta(\mathcal{N}'),$$

where g_θ denotes the rule-inference process of the vision-language reasoning model and R captures the implicit pattern of correct operation.

Different instantiations of g_θ correspond to different reasoning strategies, which we compare in our evaluation.

Conditioned on the inferred rule R, the model determines whether the query event e is consistent with normal operation:

$$\hat{y} = f_\theta(e \mid R),$$

where

$$\hat{y} = \begin{cases} 0, & \text{if } e \text{ is classified as normal,} \\ 1, & \text{if } e \text{ is classified as abnormal.} \end{cases}$$

Equivalently, the model may produce a consistency score

$$S(e, R) \in [0, 1],$$

where a higher value indicates stronger agreement with the demonstrated normal behavior.

The anomaly decision is then made by thresholding:

$$\hat{y} = \begin{cases} 0, & S(e, R) \geq \tau, \\ 1, & S(e, R) < \tau, \end{cases}$$

where τ is a task-dependent decision threshold.

The key property of this formulation is that rule R is derived from $\mathcal{N}'$ alone, without access to the full normal distribution or any labeled abnormal examples at inference time.

3.3. Testbed Setup

Figure 2 shows the common monitoring pipeline shared by the three stations. Each station supplies raw observations, either static images or video clips. The dashed path is the end-to-end configuration evaluated in Phase 1, in which the raw observation is passed directly to the reasoning model with no front-end. This was our initial design. It did not prove reliable enough for monitoring use, most notably at the belt transport station, which is what motivated the front-end. We use SAM3 to extract the relevant state in JSON format. Phase 2 then evaluates whether the model can infer the operating rule R for each station from k normal demonstrations alone. Phase 3 supplies the rule explicitly, isolating rule application from rule inference so that we can measure whether the model separates normal from abnormal states once it is no longer required to recover the rule itself. Each phase is described in detail in the Results section.

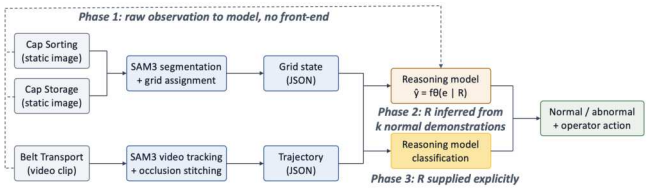
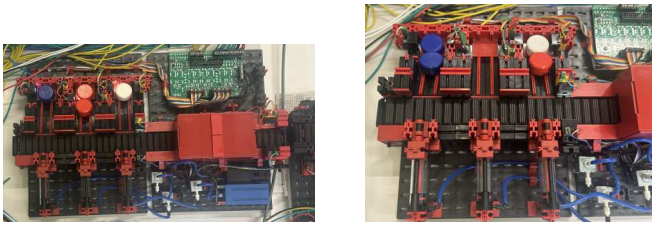


Figure 2. Common monitoring pipeline across the three stations.

3.3.1. Station 1: Cap Sorting

Figure 3 illustrates the sorting station, which detects caps of three different colors and routes them into corresponding storage bins via a conveyor belt. A push arm directs each cap toward its designated bin; since the three bins are tilted, caps fall and stack sequentially from the top of each bin downward. When a fault is detected, operator intervention is required: if a cap is in the wrong color line, the second vacuum gripper picks it up and transfers it to the correct line; if a line is empty, a signal is sent to the first arm to retrieve and deliver the missing cap. The sorting station is modeled as a 3×3 grid, where each of the nine positions can hold one of four states: red, white, blue, or empty, yielding 4^9 possible configurations. We randomly sampled 113 configurations from this space, of which 61 are incorrect and 52 are correct.



(a) Correct sorting configuration.

(b) Incorrect sorting configuration.

Figure 3. Examples of correct and incorrect cap sorting states, showing an empty column and a color misplacement.

Operation Rules. A sorting configuration is considered *correct* when all of the following hold:

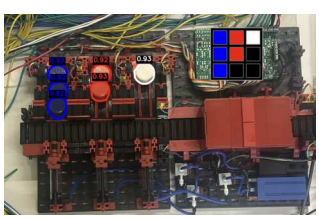
- Each column contains only caps of its designated color.
- No column that should contain caps has an empty slot.

A configuration is flagged as *incorrect* when at least one of the following is observed:

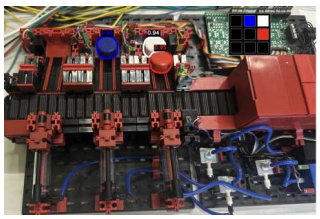
- A cap occupies a column whose designated color does not match the cap's color (color misplacement).
- A whole column is empty, indicating no workpiece waiting in the sorting line.

Image Processing. Each image is processed using SAM3 (Carion et al., 2025) in semantic segmentation mode with the text prompts "blue round cap", "red round cap", and "white round cap". Images are loaded with EXIF orientation disabled so that all frames share a consistent spatial reference before detection. SAM3 returns an instance segmentation mask and bounding box for each detected cap, labeled with its color class. Because the camera is hand-held and not fixed across captures, the spatial positions of the nine grid slots vary per image. A grid assignment step projects each detected centroid onto the 3×3 layout by clustering centroids into rows and columns based on their pixel coordinates, dynamically determining slot boundaries from the current frame rather than from fixed calibration data. Each slot is assigned one of four states: blue, red, white, or empty. The resulting

grid is exported in two formats: a JSON file recording the 3×3 grid rows as a nested list, per-slot color assignments keyed by column-row index, color counts, and total cap count; and a plain-text summary for human inspection. The JSON representation is the input fed to the vision-language model for anomaly evaluation.



(a) Correct case: all caps in designated columns.



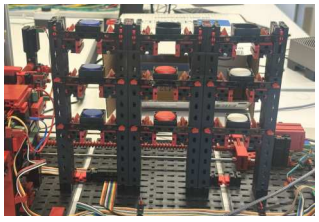
(b) Incorrect case: color misplacement detected.

Figure 4. SAM3 segmentation and grid assignment output for the sorting station. Colored overlays indicate detected cap instances; slot labels (top right) show the assigned state used as input to the reasoning model.

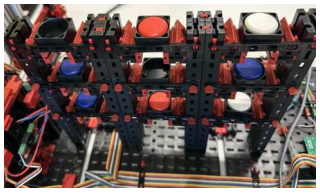
3.3.2. Station 2: Cap Storage

Immediately following color sorting, the second vacuum gripper picks each cap from the sorting line, retrieves a black socket from the high-bay warehouse, inserts the cap into the socket, and returns the assembly to the shelf. The warehouse contains nine storage spots arranged in a 3×3 layout.

Figure 5 shows representative correct and incorrect configurations. When a fault is detected, corrective action is required: the affected caps must be relocated by the gripper robot to their designated column and correct vertical position before operation can resume. Each of the nine spots can hold one of four states (red, white, blue, empty), giving 4^9 possible configurations. We randomly sampled 108 configurations, of which 60 are incorrect and 48 are correct.



(a) Correct storage configuration.



(b) Incorrect storage configuration, wrong order and color misplacement.

Figure 5. Examples of correct and incorrect cap storage states. Correct operation requires each color to occupy its designated column and spots to be filled from the bottom upward.

Operation Rules. A storage configuration is considered *correct* when all of the following hold:

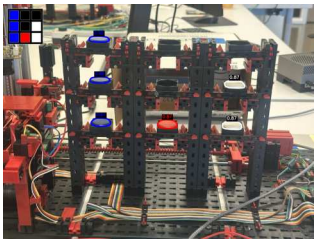
- Each column is dedicated to a single cap color.
- Spots within each column are filled from the bottom row upward, with no gaps between occupied slots.

A configuration is flagged as *incorrect* when at least one of the following is observed:

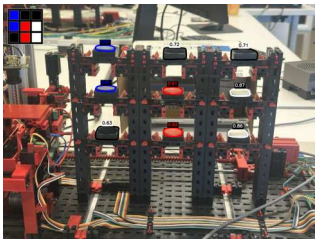
- A cap is placed in a column whose designated color does not match the cap's color (color misplacement).
- A lower slot within a column is empty while a higher slot in the same column is occupied, violating the bottom-up fill order.

Image Processing. The same SAM3-based pipeline used for the sorting station is applied here. The text prompts "blue round cap", "red round cap", and "white round cap" are issued to the semantic predictor, and each detected cap is assigned to one of the nine warehouse slots via dynamic centroid clustering. The 3×3 grid state is serialized to a JSON file with the same schema as the sorting station (slot assignments, color counts, total cap count) and passed to the vision-language model for evaluation.

Figure 6 shows representative SAM3 segmentation outputs for the storage station, illustrating both a correct bottom-up filling pattern and an incorrect configuration with a fill-order violation.



(a) Correct case: bottom-up fill order preserved.



(b) Incorrect case: fill-order violation detected.

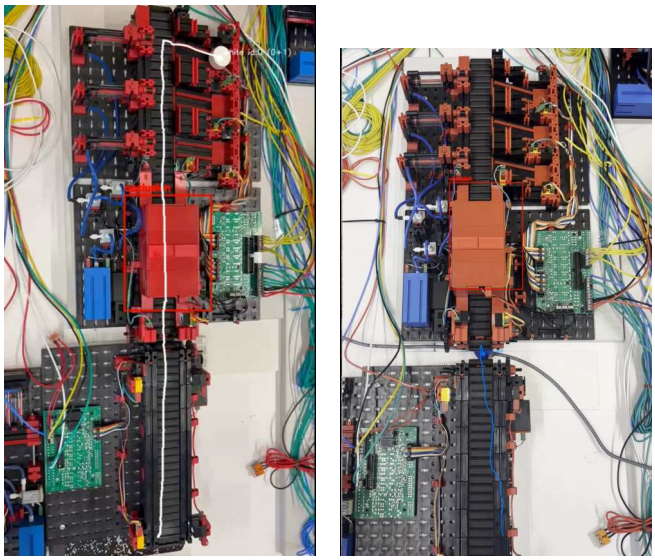
Figure 6. SAM3 segmentation and grid assignment output for the storage station. Slot labels (top left) indicate assigned cap color and row position passed to the reasoning model.

3.3.3. Station 3: Belt Transport

In this station, the input modality is video rather than a single image. We record video of the conveyor belt system during operation, and each video may contain one to five workpieces. The first vacuum gripper places a cap at the start of the belt; the belt consists of two connected units, and the cap is expected to travel either to the end of the belt or onward to the sorting station.

When a fault is detected, the belt system must be halted and the affected cap manually repositioned or removed before operation can resume.

A red occlusion box is present on the belt as part of the physical setup; it does not affect the operational outcome and is handled explicitly in our processing pipeline. We recorded 33 videos, containing 91 trajectories in total, of which 48 are incorrect and 43 are correct.



(a) Normal trajectory: cap goes to sorting station.

(b) Abnormal trajectory: cap stuck mid-belt.

Figure 7. Examples of normal and abnormal cap trajectories on the conveyor belt. The stitched trajectory is shown with the red occlusion box region indicated.

Operation Rules. A belt transport event is considered *correct* when all of the following hold:

- The cap moves continuously from the start of the belt to the end, or to the sorting station, without stopping for more than 10 seconds.
- The cap's motion direction along the belt axis is consistent with the expected transport direction throughout the trajectory.
- The cap does not interact with other caps (no collision or jamming).

A transport event is flagged as *incorrect* when at least one of the following is observed:

- The cap halts or leaves the belt at the junction between the two belt units, indicating a misalignment or poor connection.
- The cap's trajectory reverses direction or remains stationary for an extended period (10 seconds), indicating a belt direction fault or a stuck cap.
- Two or more caps are detected in close proximity in a vertical configuration, indicating a collision and jam.

Trajectory Extraction and Occlusion Stitching. Cap trajectories and the red occlusion box are extracted from video using SAM3's video semantic predictor with text prompts "blue round cap", "red round cap", "white round cap", and "red rectangle box".

For each frame, SAM3 returns bounding boxes with persistent object IDs; the centroid (c_x, c_y) of each detection is recorded.

The dominant belt axis (horizontal or vertical) is determined automatically by comparing the range of c_x versus c_y across all detections: the axis with the larger range is selected as the belt position coordinate.

Since the camera is not fixed, the pixel coordinates of the red occlusion box differ across recordings. When a cap passes behind the occlusion box, the SAM3 tracker loses the object and assigns a new ID upon re-acquisition, producing two or more disjoint track segments. These fragments are stitched back into a single continuous track using a cost-based greedy matching algorithm.

For each candidate pair of segments (s_a, s_b) ordered by time, a stitching cost is computed as

$$c(s_a, s_b) = |p_b^{\text{start}} - \hat{p}_b| + 0.05 \cdot \Delta f,$$

where $\hat{p}_b = p_a^{\text{end}} + v_a \cdot \Delta t$ is the predicted belt position extrapolated from segment s_a's velocity v_a, p_b^{start} is the observed starting position of s_b, and Δf is the frame gap between the two segments.

If either endpoint lies within or near the occlusion box region, the cost is discounted by a factor of 0.35 to favor merges that pass through the known occluder.

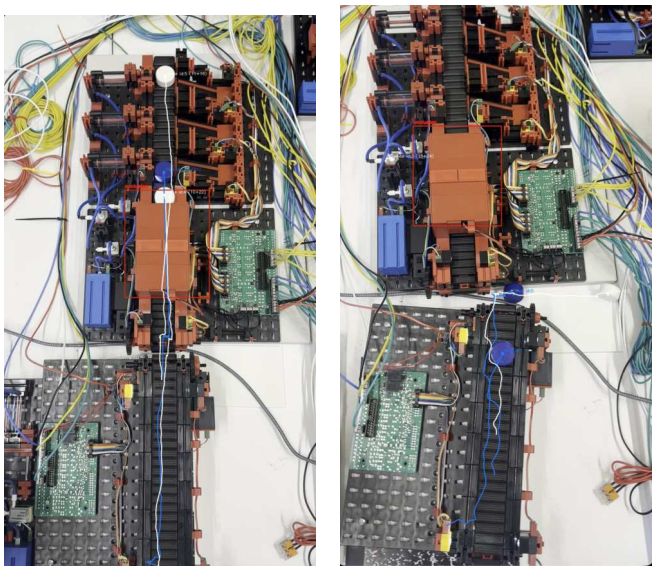
Candidate pairs with a gap exceeding a maximum threshold or with conflicting motion directions are rejected outright.

Gap frames between stitched segments are filled by linear interpolation and marked as non-visible in the output.

The merged track for each cap is exported as a JSON file containing the video metadata (path, frame rate, confidence threshold, text prompts) and, for each track, the per-frame timestamps, belt-axis positions in pixels, raw centroids, visibility flags, and mean detection score. This JSON representation is passed to the reasoning model to determine whether the observed trajectory constitutes a normal or abnormal transport event.

Figure 7 and Figure 8 both show representative stitched-trajectory visualizations for normal and abnormal transport

events, with the occlusion box region and interpolated segments indicated.



(a) Normal trajectories: continuous end-to-end motion.

(b) Abnormal trajectories: caps fall off.

Figure 8. Stitched trajectory output for the belt transport station.

4. RESULTS

4.1. Experimental Setup

We evaluate our approach on three manufacturing stations: cap sorting, cap storage, and belt transport, across three experimental phases described below.

The reasoning models evaluated are:

- **Qwen3.5** (Qwen Team, 2026): A 0.8 B open-weight model from Alibaba's Qwen3 family with hybrid thinking mode support.
- **Nemotron3 Nano** (NVIDIA et al., 2025): A 30 B reasoning-optimized model from NVIDIA, distilled for high accuracy at a compact scale.
- **GPT-5.4 mini** (OpenAI, 2026): OpenAI's lightweight multimodal model with strong instruction-following capability.
- **Gemini 3.5 Flash** (Google DeepMind, 2025): Google DeepMind's multimodal model with an extended context window and enhanced chain-of-thought reasoning.
- **Claude Sonnet 4.6** (Anthropic, 2025): Anthropic's hybrid reasoning model with extended-thinking mode for multi-step tasks.

Phases 2 and 3 use the JSON grid representation for the sorting and storage stations, and the stitched trajectory JSON for the belt transport station.

4.2. Phase 1: End-to-End Rule Inference

Our initial design placed no intermediate representation between the camera and the reasoning model. This phase reports that configuration, corresponding to the dashed path in Figure 2: raw static images (sorting, storage) and raw video clips (belt transport) are supplied to the model directly, and the model is asked to infer the operating rule from nominal examples. No segmentation, grid assignment, or trajectory extraction is applied. Each model receives $k = 10$ observations of correct operation in the same raw modality as the query, so the demonstration budget matches the largest setting evaluated in Phase 2. Table 1 summarizes the outcome.

Model	Cap Sorting	Cap Storage	Belt Transport
Qwen3.5	N	N	N
Nemotron3 Nano	N	N	N
GPT-5.4 mini	N	N	N
Gemini 3.5 Flash	N	Y	N
Claude Sonnet 4.6	Y	N	N

Table 1. Rule inference from raw observations at $k = 10$. **Y** = all rules correctly inferred; **N** = rule incomplete or incorrect.

Only Claude Sonnet 4.6 on cap sorting and Gemini 3.5 Flash on cap storage yield a complete rule, and no model recovers the belt transport rule.

Every model reliably identifies the color-to-column constraint at both static stations, stating without exception that caps of a given color belong to a single designated line or column. The constraints that are missed are of a different kind. At the sorting station, models fail to infer that a designated lane may not be left empty. At the storage station, the models frequently fail to infer that slots must be filled from the bottom upward with no gaps, and their descriptions are additionally disturbed by caps resting on the background surface behind the shelf: rather than restricting attention to the nine shelf slots, several models incorporate background objects into the arrangement they describe. At the belt station, most models cannot track a single cap from the start of the belt to its destination, and identity is most often lost where the cap passes behind the red occlusion box, after which the model treats the re-emerging cap as a separate object.

Because a correct rule is a precondition for the classification step that follows it, and rule inference already failed at the largest demonstration budget we evaluate, we did not proceed to query classification in this configuration. We take this as evidence that end-to-end monitoring from raw observations is not yet practical at this level of spatial and temporal granularity. Rather than prompting the models harder, we delegate perception to a promptable segmentation front-end and retain the reasoning model for the tasks it performs well. Phases 2 and 3 report that configuration.

4.3. Phase 2: Rule Inference from Normal Demonstrations

In this phase, for each station we supply the model with k normal demonstration events drawn from $\mathcal{N}'$ (correct examples only) and ask it to infer the operational rule $R = g_\theta(\mathcal{N}')$ before classifying each query event.

We compare models to assess whether they can reliably derive the correct rule solely from normal examples.

We vary $k \in \{3, 5, 10\}$ and report whether each model correctly derives all rules for each station. For sorting and storage, k counts grid JSON files (one per image); for belt transport, k counts trajectories (one video may contribute multiple). We verified that the JSON files for these events are correct. Table 2 summarizes the results, where **Y** indicates all rules correctly inferred and **N** indicates an incomplete or incorrect rule. For sorting and storage, models consistently identify the color-to-column assignment rule even at small k, but require more demonstrations to infer the secondary constraints: that no column may be empty (sorting) and that slots must be filled bottom-up (storage). For belt transport, the trajectory can terminate at the belt end or at any of the three sorting lines, so k must be large enough to cover all destination variants before a reliable rule can be inferred.

Station	Model	$k=3$	$k=5$	$k=10$
Cap Sorting	Qwen3.5	N	Y	Y
	Nemotron3 Nano	N	N	Y
	GPT-5.4 mini	N	Y	Y
	Gemini 3.5 Flash	Y	Y	Y
	Claude Sonnet 4.6	N	Y	Y
Cap Storage	Qwen3.5	N	N	Y
	Nemotron3 Nano	N	Y	Y
	GPT-5.4 mini	N	N	Y
	Gemini 3.5 Flash	N	Y	Y
	Claude Sonnet 4.6	N	Y	Y
Belt Transport	Qwen3.5	N	N	N
	Nemotron3 Nano	N	N	N
	GPT-5.4 mini	N	N	Y
	Gemini 3.5 Flash	N	N	Y
	Claude Sonnet 4.6	N	N	Y

Table 2. Rule inference correctness across models and number of normal demonstrations k. **Y** = all rules correctly inferred; **N** = rule incomplete or incorrect.

4.4. Phase 3: Classification under Explicit Rule Prompting

In this phase, the ground-truth operational rule for each station is provided directly in the prompt, as described in the Operation Rules paragraphs of Section 3. Each model is given the rule together with the JSON representation of a query event and asked to classify it as normal or abnormal.

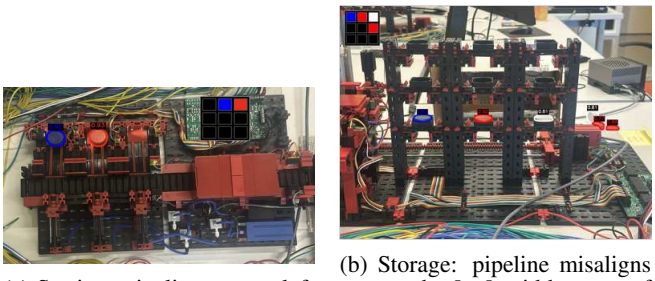
The 10 normal demonstrations used in Phase 2 are excluded, leaving the following evaluation sets: sorting, 103 events (61 incorrect, 42 correct); storage, 98 events (60 incorrect, 38 correct); belt transport, 81 trajectories (48 incorrect, 33 correct). This phase isolates classification capability from rule-inference capability, providing an upper-bound reference for Phase 2.

Since the JSON inputs fed to each model are identical across all models, any classification errors reflect either model reasoning failures or inaccuracies introduced upstream in the image-processing pipeline.

Table 3 reports the confusion matrix for each model on each station. The three stations behave quite differently, so we discuss each in turn in the paragraphs below, covering in each case what the numbers show and whether the residual errors originate in the perception front-end or in the reasoning model.

Station	Model	TP	TN	FP	FN
Cap Sorting	Qwen3.5	61	42	0	0
	Nemotron3 Nano	61	42	0	0
	GPT-5.4 mini	61	42	0	0
	Gemini 3.5 Flash	61	42	0	0
	Claude Sonnet 4.6	61	42	0	0
Cap Storage	Qwen3.5	57	35	3	3
	Nemotron3 Nano	59	35	3	1
	GPT-5.4 mini	59	35	3	1
	Gemini 3.5 Flash	59	35	3	1
	Claude Sonnet 4.6	59	35	3	1
Belt Transport	Qwen3.5	43	28	5	5
	Nemotron3 Nano	42	29	4	6
	GPT-5.4 mini	45	29	4	3
	Gemini 3.5 Flash	46	29	4	2
	Claude Sonnet 4.6	46	30	3	2

Table 3. Confusion matrix for Phase 3 (explicit rule prompting) across models and stations. TP = correctly flagged abnormal; TN = correctly accepted normal; FP = incorrectly flagged abnormal; FN = missed fault.



(a) Sorting: pipeline reports left column empty instead of right.

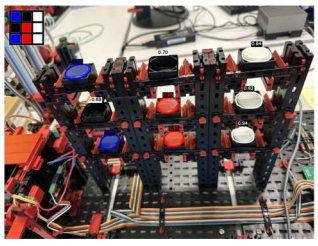
(b) Storage: pipeline misaligns caps to the 3×3 grid because of caps detected in the background.

Figure 9. Representative image-processing pipeline errors in the sorting (left) and storage (right) stations. In both cases the pipeline misidentifies cell occupancy, yet the underlying events are genuinely abnormal regardless of the specific error reported.

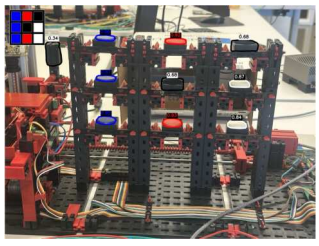
Station 1: Cap Sorting. All five models achieve perfect classification on the sorting station (61 TP, 42 TN, 0 FP, 0 FN). The sorting rule is relatively straightforward: no column of the 3×3 grid may be entirely empty, and each colored cap must occupy its designated color area. Given an explicit rule, all models consistently apply this criterion in their chain-of-thought reasoning. The image-processing pipeline does introduce one misidentification, reporting the left column as empty when in fact the right white-cap column is empty (Figure 9, left). This case remains a genuinely abnormal placement regardless of which column is incorrectly reported, so the classification label is unaffected.

Station 2: Cap Storage. The storage station reveals a small number of misclassifications. Qwen3.5 produces 3 FP and 3 FN, while the remaining four models each yield 3 FP and 1 FN. The shared 3 FP and 1 FN all trace back to pipeline inaccuracies rather than model reasoning errors: caps resting on the background table surface are incorrectly included (Figure 9, right); partially occluded caps are missed by the segmentation pipeline because of the tilted camera angle; and the grid-alignment step sometimes maps caps to the wrong cell in the 3×3 layout.

The two additional false negatives unique to Qwen3.5 stem from a distinct model reasoning failure: the model incorrectly accepts configurations in which the middle row of the shelf is empty while the upper row is occupied, violating the bottom-up filling rule. Representative cases are shown in Figure 10.

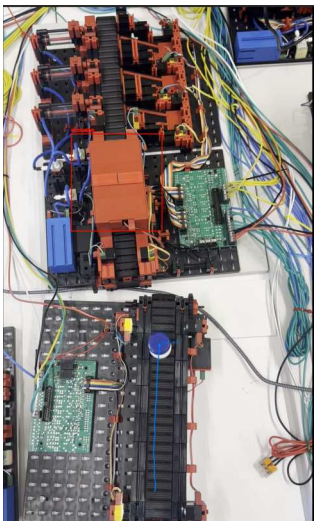


(a) FN case 1: blue upper row occupied, middle row empty.

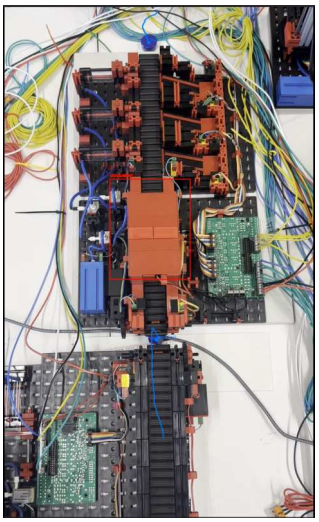


(b) FN case 2: red upper row occupied, middle row empty.

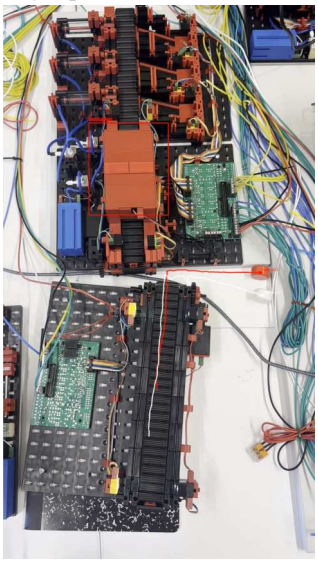
Figure 10. The two additional false negatives produced by Qwen3.5 in the storage station.



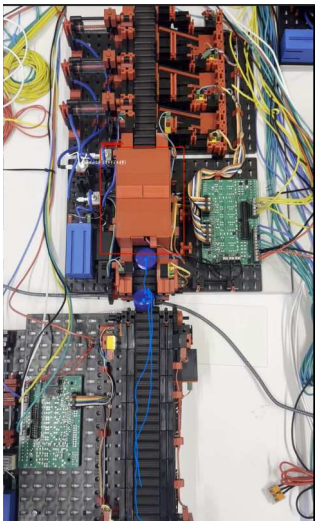
(a) Occlusion FP: white cap hidden beneath blue cap; SAM3 registers only one object until they separate.



(b) End-of-belt FN: blue cap rests at belt exit due to camera shake; trajectory does not originate from belt entry.



(c) Misalignment FN: belt offset causes caps to roll rightward off the first belt, mimicking a valid sorting trajectory.



(d) Belt-gap FP: a minor gap induces brief cap rotation but the cap continues forward without falling.

Figure 11. Representative belt transport classification errors.

Station 3: Belt Transport. The belt transport station is considerably more complex than the preceding stations. We list the most common error sources we observed. In three events, shown in Figure 11a, a white cap travels directly beneath a blue cap on the belt. Because the two caps are vertically stacked, SAM3 detects only a single object until the caps separate, yielding a trajectory that appears to contain one fewer cap than expected. All five models classify these events as abnormal, producing three shared false positives.

Two events, illustrated in Figure 11b, involve a cap that comes to rest in the background region near the belt exit. Because the camera is hand-held and not perfectly steady, the cap's tracked trajectory drifts frame-to-frame and terminates at the belt end rather than following a full entry-to-exit path. GPT-5.4 mini, Gemini 3.5 Flash, and Claude Sonnet 4.6 identify these events as abnormal on the grounds that the trajectory does not originate from the belt entry point, whereas Qwen3.5 and Nemotron3 Nano accept them as normal, contributing two additional false negatives to those models.

Figure 11c shows one of four events arising from a physical misalignment between the two belt sections. Caps fall off the first belt and roll to the right, producing trajectories that superficially resemble a valid path toward the sorting station. Because no rigid lateral boundary is enforced in the rule prompt, a model-dependent subset of these trajectories is accepted as correct, yielding between two and four false negatives across models.

In one further event, shown in Figure 11d, the cap crosses a small gap between the two belts. The gap is not large enough for the cap to fall off; instead the cap rotates briefly and continues moving forward, completing a normal transit. Qwen3.5, Nemotron3 Nano, GPT-5.4 mini, and Gemini 3.5 Flash all flag the rotational perturbation as a failure, adding one false positive each.

5. DISCUSSION

All five models achieve perfect classification on the sorting station and near-perfect on storage, confirming that VLMs can reliably enforce discrete placement rules when the rule prompt closely mirrors the structured JSON input.

Performance degrades on the belt transport station, where the input is a trajectory derived from a sequence of frames rather than a single static image. Trajectory information alone proves insufficient for reliable anomaly detection: temporary cap occlusions cause SAM3 to undercount objects, end-of-belt drift from a hand-held camera produces paths that do not conform to a canonical entry-to-exit shape, and belt misalignment generates lateral deviations that superficially resemble valid sorting paths. These ambiguities are difficult to resolve without richer temporal context or tighter spatial priors, neither of which the current pipeline provides.

The end-to-end configuration is weaker still. When the same demonstrations are supplied as raw images and video with no intermediate representation, only two of fifteen model-station combinations recover a complete operating rule, and none does so for belt transport. The limitation therefore lies in grounding rather than in rule reasoning: a model that cannot fix a stable spatial or temporal reference cannot state a rule about what should be present and is not. Current VLMs do not reliably ground fine-grained spatial relationships, such as exact grid-cell occupancy or precise trajectory shape, from unprocessed imagery, which is consistent with benchmark findings that multimodal models trail specialized methods on fine-grained industrial perception ("MMAD: A Comprehensive Benchmark for Multimodal Large Language Models in Industrial Anomaly Detection", 2025). The explicit segmentation and annotation steps are therefore necessary to make the task tractable, but they add engineering overhead and introduce their own failure modes, as the pipeline errors in the storage and belt stations demonstrate.

Binary classification is stable, but root-cause reasoning is not. When prompted to explain why an event is abnormal and how to remedy it, model responses become inconsistent. This is most evident in the belt transport station, where a cap falling off or stalling can arise from several distinct physical causes: misaligned belt sections, an inter-belt gap, reversed belt direction, or an imprecise gripper placement. Identifying the true cause requires the model to reason backwards from a trajectory to the underlying mechanism, a substantially harder inference problem than binary classification, and none of the current models handles it reliably.

Overall, this work is best understood as a preliminary study. It demonstrates that VLMs can monitor and evaluate manufacturing workflows by internalizing explicit rules expressed in natural language, but the conditions are deliberately simple and controlled, and the limitations above point to concrete open problems. Table 4 summarizes what each phase tests and what it shows.

Phase	What it tests	Outcome
Phase 1	Rule inference from raw images and video	Fails: 2 of 15 model–station pairs, none on belt transport
Phase 2	Rule inference from extracted symbolic state	Succeeds at $k = 10$ for all stations, except belt for the two smallest models
Phase 3	Rule application, rule supplied explicitly	Sorting solved (100%), storage near-solved, belt degraded

Table 4. Summary of what each phase tests and what it shows.

6. FUTURE WORK

The most direct extension is to remove the preprocessing front-end and explore pipelines that accept raw images or short video clips as input, allowing the model to learn the relevant spatial and temporal structure end-to-end. Removing the preprocessing layer would substantially reduce engineering complexity and eliminate a class of pipeline-induced errors.

A second direction is root-cause reasoning and automated remediation: identifying the physical cause of a fault and proposing a corrective action, for instance instructing the

gripper to reposition a cap or alerting an operator to realign the belt sections. Developing reliable prompting strategies and evaluation protocols for this open-ended task is itself a non-trivial challenge.

A third is to quantify the trade-off between VLM-based monitoring and explicit rule-based checking. A deterministic checker is fast, reproducible, and auditable, whereas a VLM incurs a per-event inference cost, is stochastic, and may shift in behavior when a hosted model is updated, which complicates revalidation. Set against this, writing a checker presupposes that someone already knows the correct rule. In practice that knowledge is often held by a small number of domain experts, is tacit rather than documented, and is incomplete until failures have been observed. The demonstration-based route requires no such input: an operator need only supply examples of correct operation, without being able to articulate what makes them correct. A hybrid arrangement is therefore worth investigating, in which the VLM induces a candidate rule offline from demonstrations and proposes revisions as new failure modes appear, while the induced rule is compiled into a deterministic checker that runs online, giving the configuration effort of one approach and the cost and auditability of the other.

Finally, all three stations here involve rules that are concise and easy to articulate, which is the regime in which handwritten logic is most competitive. The potential advantage of VLMs lies in constraints that are difficult to verbalize but clear from visual demonstration, and future work will target assembly and logistics tasks in that regime, where rules emerge implicitly from example sequences rather than explicit descriptions.

REFERENCES

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., ... others (2022). Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Anthropic. (2025). *Introducing claude sonnet 4.6.*

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). Robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469–483.

Bergmann, P., Fauser, M., Sattlegger, D., & Steger, C. (2019). Mvtec ad: A comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 9592–9600).

Carion, N., Gustafson, L., Hu, Y.-T., Debnath, S., Hu, R., Suris, D., ... others (2025). Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*.

Chibani, A., Coudert, T., & Kamsu-Foguem, B. (2022). A review of deep learning in the study of materials degradation. *Engineering Applications of Artificial Intelligence*, *113*, 104940.

Google DeepMind. (2025). *What's new in gemini 3.5 flash.*

Guan, T., Liu, F., Wu, X., Xian, R., Li, Z., Liu, X., ... others (2023). Hallusionbench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. *arXiv preprint arXiv:2310.14566*.

Kragic, D., & Christensen, H. I. (2002). A survey of vision-based robotic manipulation. In *Ieee international conference on robotics and automation* (pp. 1810–1815).

MMAD: A comprehensive benchmark for multimodal large language models in industrial anomaly detection. (2025). In *International conference on learning representations (iclr).* (arXiv:2410.09453)

Newman, T. S., & Jain, A. K. (1995). Machine vision applications in manufacturing. *Computer Vision and Image Understanding*, *61*(2), 231–262.

NVIDIA, :, Blakeman, A., Grattafiori, A., Basant, A., Gupta, A., ... Yan, Z. (2025). *Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning.* Retrieved from `https://arxiv.org/abs/2512.20848`

OpenAI. (2023). Gpt-4v(ision) system card. *OpenAI Technical Report*.

OpenAI. (2026). *Introducing gpt-5.4 mini and nano.* (Accessed June 2026)

Pang, G., Shen, C., Cao, L., & Hengel, A. v. d. (2021). Deep learning for anomaly detection: A survey. *ACM Computing Surveys*, *54*(2), 1–38.

Qwen Team. (2026, February). *Qwen3.5: Towards native multimodal agents.*

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... others (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning* (pp. 8748–8763).

Radke, R. J., Andra, S., Al-Kofahi, O., & Roysam, B. (2005). Change detection techniques. *IEEE Transactions on Image Processing*, *14*(3), 294–307.

Xia, Z., Li, H., Li, W., & Song, B. (2024). Large language models in manufacturing: Opportunities and challenges. *Journal of Manufacturing Systems*.

Zhang, J., Huang, J., Jin, S., & Lu, S. (2024). Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.