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

World Model Failure Classification and Anomaly Detection for Autonomous Inspection

Michelle Ho,[1,2] Muhammad Fadhil Ginting[1,2], Isaac R. Ward[1], Andrzej Reinke[2], Mykel J. Kochenderfer[1], Ali-akbar Agha-Mohammadi[2], and Shayegan Omidshafiei[2]

***Abstract*— Autonomous inspection robots for monitoring industrial sites can reduce costs and risks associated with human-led inspection. However, accurate readings can be challenging due to occlusions, limited viewpoints, or unexpected environmental conditions. We propose a hybrid framework that combines supervised failure classification with anomaly detection, enabling classification of inspection tasks as a success, known failure, or anomaly (i.e., out-of-distribution) case. Our approach uses a world model backbone with compressed video inputs. This policy-agnostic, distribution-free framework determines classifications based on two decision functions set by conformal prediction (CP) thresholds before a human observer does. We evaluate the framework on gauge inspection feeds collected from office and industrial sites and demonstrate real-time deployment on a Boston Dynamics Spot. Experiments show over 90% accuracy in distinguishing between successes, failures, and OOD cases, with classifications occurring earlier than a human observer. These results highlight the potential for robust, anticipatory failure detection in autonomous inspection tasks or as a feedback signal for model training to assess and improve the quality of training data. Project website: `https://autoinspection-classification.github.io/`**

I. Introduction

Industrial inspection robots can offer consistent, reliable monitoring of instruments in large facilities, reducing costs and inconsistencies from human-led inspections [1]. In these environments, accurate readings are critical. However, vision may be limited when instruments are partially occluded by other equipment or truncated in the field of view. Additional challenges arise due to environmental conditions such as glare and shadows, which likely fall outside of the known cases for which the robot was trained. To successfully complete its mission, the robot needs to not only get accurate instrument readings but also detect and react appropriately to these perception failures. In this work, we address the challenge of accurate, efficient online failure detection for an inspection robot, focusing on both known failure cases, such as systemic misreads, and anomalous, out-of-distribution (OOD) cases due to environmental conditions.

Human-led inspections are costly, unreliable, and complex to scale up. Mobile robots address these limitations by providing reliable measurements on a regular schedule. They enable inexpensive, time-efficient inspection without human fatigue and reduce unnecessary exposure in hazardous domains [2]. Legged robots have been tested in offshore high-voltage environments [3] and in uncertain domains requiring risk-aware planning and semantic mapping [4]. These mobile robots have been adopted by industry players for automating inspection tasks such as thermal and acoustic monitoring, 3D mapping of facilities, and gauge inspection [5]–[7].

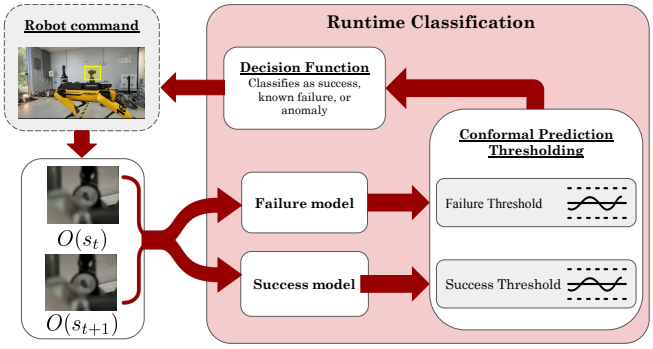


Fig. 1: Our framework classifies successes, known failures, and anomalies with a world model backbone and conformal prediction thresholding, allowing for the robot to take informative actions based on the classification.

Greater reliance on autonomy introduces new risks. For instance, if a robot misses a dangerous reading on a high-pressure system, an anomaly can escalate into a major safety risk. Fault and failure detection have long histories in process systems [8]. Still, modern inspection robots introduce additional challenges due to learned perception and operation in unseen environments. Effective autonomous inspection requires detection methods that can identify known failures but also adapt to novel conditions not encountered during training. To address these challenges, OOD detection provides early warnings when inputs deviate from training distributions [9]. Recent work demonstrates that it is possible to detect failures without explicit labeled failure data with conformal prediction thresholding [10], [11]. However, OOD detection alone is insufficient, as it does not provide informative actions to take upon detection, unlike approaches that leverage knowledge of recognized failure cases.

In this paper, we present a failure detector for real-time anomaly monitoring for an autonomous gauge inspection problem (see Figure 1). Our method combines supervised failure classification with sequential OOD detection through conformal prediction–based thresholding. The detector builds on previous work that uses a world-model backbone to predict observations, from which anomaly scores are derived with prediction errors [11]. Unlike prior work that focused

[1]Stanford University, Stanford, CA 94305 USA {mtho, ginting, irward, mykel}@stanford.edu
[2]Field AI, Mission Viejo, CA 92691 USA

solely on using success data only to classify anomalies [10], [11], we incorporate explicit classification of known failure cases. We make the following technical contributions:

1) A hybrid detector that combines failure classification, for providing informed, corrective actions, with OOD detection, for failures not seen during training;
2) Calibrated conformal prediction thresholds based on standard CP metrics, enabling classification of successes, known failures, and OOD readings;
3) Evaluation on gauge inspection data gathered in both office and industrial settings;
4) Real-time, online deployment of the detector in a gauge inspection scenario on a Boston Dynamics Spot robot.

These contributions not only enable robust real-time failure detection but can also assess training data quality to guide targeted data collection for large-scale model training.

II. Related Work

A. Out-of-Distribution Detection

Detecting anomalies is critical for safe, robust autonomous systems, especially in safety-critical domains such as industrial inspection. Anomalies are often defined as out-of-distribution from what the robot has seen during training. OOD detection flags anomalies, since they could indicate possible failure modes [9]. Approaches to OOD detection are vast. Statistical and reconstruction methods can be simple and interpretable but struggle with high dimensionality, temporal dependencies, or noisy data [9], [12], [13]. Embedding-based [14], [15], distribution modeling [16]–[20], ensemble [21], and Bayesian approaches [22] scale better to complex settings but require large, diverse datasets.

Recent work has focused on detecting failures without failure data. One method, FAIL-Detect [10], learns scalar signals correlated with failures and calibrates them with time-varying conformal prediction thresholds [10]. Another recent work builds on this method by using a world model backbone to detect failures in a highly dimensional problem. It also incorporates history to anticipate future failures before they occur [11]. Though these methods do not require labeled data for training, in many cases, failure data can be readily collected during the training process and could be used to improve performance.

B. Failure Classification

While anomaly detection identifies deviations from nominal behavior, failure classification can categorize these deviations if they have been seen before. For industrial inspection, these failure classes can include mobility and actuation [8], [23], perception [24], and task-level planning and execution [25]. Residual-based approaches for classification are interpretable and leverage a system model to generate diagnostic signals, but they are sensitive to noise or approximately modeled dynamics [26], [27]. Supervised learning methods are useful when the model is not easily defined [28], [29]. Large language models provide semantic understanding of the scene beyond recognizing pixel patterns [30], [31]. Hybrid approaches for detecting different known failure cases include adding a supervised head to an autoencoder [32] or combining separate frameworks for detecting various types of failures [33]. In contrast, our approach detects multiple categories of perception failures using a single pretrained world model.

Though failure detection is useful for prescribing informed, corrective actions for known cases, it is not possible to train a robot to recognize every future failure it may ever encounter. Therefore, we address this by creating a unified framework that combines failure classification with anomaly detection.

C. Hybrid Anomaly Detection and Failure Classification

Some anomaly detection methods can be combined with failure classification when labeled data for specific cases is available. For instance, one approach combines principal component analysis with supervised classifiers [34], another adds classification heads to autoencoders [32], and another uses ensembles for detecting anomaly signals and making class predictions [21]. In contrast, we combine anomaly detection and failure classification using a single world model backbone to reduce architecture complexity for deployment.

Moreover, this combination not only supports timely, more informed detection but also supports intervention strategies for specific failure modes. Some prior work focuses on defining failure beyond collision and overriding actions to prevent failure, such as with latent space binary failure classification using Hamilton–Jacobi reachability [35]. However, our focus on combining anomaly detection with multi-class failure classification not only flags unseen deviations but also distinguishes among known failure modes, enabling more informative responses. We test our framework on a real-time hardware setup and trigger appropriate intervention strategies given the classification.

D. Conformal Prediction

Conformal prediction (CP) is a statistical framework that quantifies uncertainty by calibrating thresholds [36]. For a trained model and a calibration dataset, CP produces prediction intervals that contain the true label (or nominal behavior) by some user-specified metric [36]. Since CP can be applied to black-box models, it is beneficial for safety-critical robotic applications. It is common to separate available data into distinct training and calibration sets [37]. Typical scoring methods include probability-based scores (e.g., margin score, inverse probability score, negative log-likelihood), residual-based scores (e.g., regression residuals, reconstruction error), and distance scores (e.g., Mahalanobis distance, latent standard deviation, clustering distance) [37]. Other scoring methods require policy outputs or perform better with an exocentric view [10]. We do not consider these in our setting because our inspection pipeline operates on robot-egocentric visual feeds for a black-box policy.

CP is typically used to construct prediction sets with some guaranteed coverage. For anomaly detection, CP can

be applied to calibrate thresholds on scalar scores, setting approximate bounds that separate nominal and OOD inputs rather than interval coverage. In line with recent work on failure detection [10], [11], we adopt this thresholding method, applying CP bands to data gathered from a camera feed. We are not aware of prior work in robotics that uses CP in a hybrid manner to both classify failure modes and reject out-of-distribution inputs within a single framework.

III. Methodology

A. Problem Formulation

Our goal is to determine if and when a robot will fail its task and to distinguish between a known failure and an anomaly. For a given task, we represent the robot's trajectory over a finite horizon T as the sequence $\tau_T = \{s_0, a_0, \ldots, s_T, a_T\}$. The objective is to determine whether there exists a failure time $k \leq T$, and if so, to identify k.

We assume an observation function $O(s_t)$ that is noise-free and fully observable, such that observations provide a direct mapping to the underlying state and thereby reveal whether the robot is failing its task. The policy $\pi(s_t) = a_t$ is tele-operated, making it unknown to the framework. Therefore, we work with the observable trajectory $\tau_t \triangleq \{o_0, \ldots, o_t\}$ which is fully informative about the underlying states.

Like in Xu et al. [10], we also frame our detector as a decision function $D(\tau_t; \theta)$, where θ contains the function parameters. This function decides whether a failure (1) occurs or not (0) at time t in the trajectory. An ideal detector minimizes the delay between the true failure time k and the detected time $\hat{k}$.

B. Failure Classification and Anomaly Detection Framework

Given observation trajectories, our full decision function is comprised of two decision functions, $D_{\text{success}}(\tau_t; \theta_s)$ and $D_{\text{fail}}(\tau_t; \theta_f)$ parameterized by θ_s and θ_f respectively. A trajectory is classified as OOD only if both decision functions return 1, i.e.,

$$D_{\text{OOD}}(\tau_t) = \mathbf{1}\big(D_{\text{success}}(\tau_t; \theta_s) = 1 \ \wedge \ D_{\text{fail}}(\tau_t; \theta_f) = 1\big). \tag{1}$$

Under this formulation, our detected time is

$$\hat{k} = \min\left\{t \leq T \ : \ D_{\text{success}}(\tau_t; \theta_s) = 1 \ \vee \ D_{\text{fail}}(\tau_t; \theta_f) = 1\right\}. \tag{2}$$

We label the classification at $\hat{k}$ by

$$\hat{c}(\tau_{\hat{k}}) = \begin{cases} \text{success}, & \text{if } D_{\text{success}}(\tau_{\hat{k}}; \theta_s) = 0 \\ & \wedge \ D_{\text{fail}}(\tau_{\hat{k}}; \theta_f) = 1, \\ \text{known failure}, & \text{if } D_{\text{fail}}(\tau_{\hat{k}}; \theta_f) = 0 \\ & \wedge \ D_{\text{success}}(\tau_{\hat{k}}; \theta_s) = 1, \\ \text{anomalous}, & \text{if } D_{\text{OOD}}(\tau_{\hat{k}}) = 1. \end{cases} \tag{3}$$

Figure 2 shows the decision function.

The two underlying decision functions are designed with the following framework:

1) Train two predictive models on observation pairs: $M_{\text{success}}(\cdot; \phi_s)$ using successful data and $M_{\text{fail}}(\cdot; \phi_f)$ using failure data.

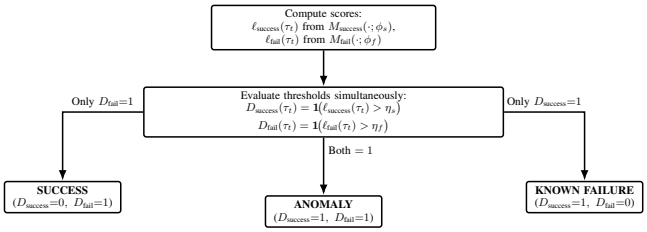


Fig. 2: Decision function for failure classification.

2) For a set of success and failure calibration sets, use the respective models to assign scores to each trajectory:

$$\ell_{\text{success}}(\tau_t), \qquad \ell_{\text{fail}}(\tau_t). \tag{4}$$

3) Use the scores to derive fixed thresholds η_s and η_f using conformal prediction (CP). These thresholds specify cutoff values for anomalies for each decision function:

$$D_{\text{success}}(\tau_t; \theta_s) = \mathbf{1}\big(\ell_{\text{success}}(\tau_t) > \eta_s\big), \tag{5}$$
$$D_{\text{fail}}(\tau_t; \theta_f) = \mathbf{1}\big(\ell_{\text{fail}}(\tau_t) > \eta_f\big). \tag{6}$$

4) Combine the two detectors to obtain the consensus OOD decision using Equation (1).

With its CP basis, this framework is policy-agnostic, distribution-free, and compatible with many scoring methods.

C. Classifying Failures for Gauge Detection

We apply our framework to autonomous gauge inspection in both office and industrial sites. The site contains numerous gauges that vary in location, size, and shape. A trajectory is classified as:

- **Success:** The robot centers and orients itself and its onboard pan-tilt-zoom (PTZ) correctly to obtain an accurate gauge reading.
- **Failure:** The robot produces an inaccurate gauge reading, such as when the gauge is partially or fully occluded or viewed at a poor angle. The gauge cannot be centered because of the limited PTZ configuration space. These can be dangerous if the robot misses a reading that could indicate abnormal equipment behavior or unsafe conditions.
- **OOD:** The robot is unable to produce a reading due to suboptimal but non-safety-critical conditions (e.g., shadows, glare, blurriness, poor lighting). These cases are not unsafe since no incorrect measurement is reported, but they should not be considered successful.

Examples of all of these cases can be seen in Figure 4.

The objective is to detect the earliest time $k \leq T$ at which a trajectory leads to failure, such that corrective actions can be taken early. The robot is tele-operated and assumes noise-free camera observations.

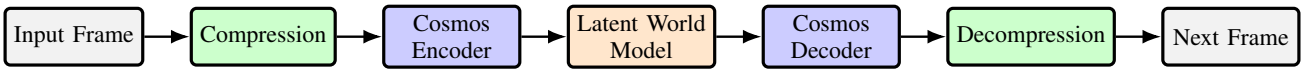


Fig. 3: Model pipeline: Input frame from video feed is optionally compressed, tokenized by Cosmos, propagated through a latent world model, decoded by Cosmos, optionally decompressed, and reconstructed into the next frame.

(a) Success (b) Failure (c) Anomaly (OOD)

Fig. 4: Examples of Gauge Image Classifications. We assume a fixed robot pose, so only the PTZ camera is controlled.

D. Dataset Creation

Since we needed to train and validate two models, calibrate two CP bands, and evaluate three classifications, we used the following dataset sizes:

- 14 success and 14 failure training videos
- 6 success and 6 failure validation videos
- 45 success and 37 failure CP calibration videos
- 45 success, 37 failure, and 53 OOD test videos,

for a total of 290 videos, each on average 10 seconds long. This choice was inspired by prior work on encoding synchronized camera feeds with a world model [11]. The set of known failures includes missing the gauge in the field of view, partial obstruction of the gauge or its needle, and viewing it at a wide angle that prevents an accurate reading. The OOD dataset consists of conditions that prevent the robot from getting a gauge measurement, such as shadows, glare, and blurriness. Our dataset was limited to what was collected from July 9, 2025, to August 8, 2025, from our two environments, an office and an industrial site.

E. Model Architecture and Training

The model architecture is adapted from prior work [11], which employed a world model trained on a success dataset from multiple synchronized camera feeds. In contrast, we extend the approach by requiring inference from two models along with their respective conformal prediction bands. Consistent with the prior work [11], we use the NVIDIA Cosmos Tokenizer (Continuous Videos) [38], but keep the weights frozen. Its pretraining eliminated the need to train a large VAE ourselves. The model was used on video feeds to predict future frames. For a video feed, tokenized latents are concatenated along channels, yielding shape $[B, H/16, W/16]$. A latent world model then predicts z_{t+1} from z_t. Before being fed into the Cosmos encoder, the images are compressed from 1200×700 to 512×288, to reduce memory usage and speed up training. The flow is illustrated in Figure 3.

Training is managed through the PyTorch Lightning Trainer API. Table V summarizes the training parameters. We train the world model with a composite loss that balances frame reconstruction, temporal consistency, and anomaly–sensitive regularization:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rec}} + \big(\mathcal{L}_{\text{rec}} - \mathcal{L}_{\text{cross}}\big) + 0.5\,\mathcal{L}_{\text{hyb}}. \tag{7}$$

Here, $\mathcal{L}_{\text{rec}}$ is a weighted pixel loss (MSE+SSIM) between predicted and ground–truth next frames, $\mathcal{L}_{\text{cross}}$ enforces temporal consistency by encouraging predictions to be closer to the next frame than the current frame, and $\mathcal{L}_{\text{hyb}}$ combines latent prediction error, perceptual similarity (LPIPS), and a weak center–region prior. We use a 70/30 training/validation split. Because consecutive frames often contain minimal differences, we reduce redundancy by applying a skip parameter to lower computation time and overhead. We store per-frame latents $\{z_t, z_{t+1}\}$ to be used with some CP metrics. Training each model took about 34 hours with early stopping.

F. Conformal Prediction Threshold Calibration

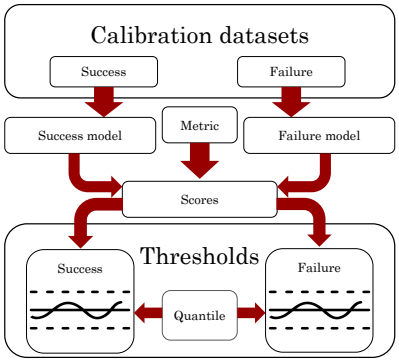


Fig. 5: Threshold Calibration Phase

We adopt CP to calibrate thresholds on anomaly scores, consistent with recent work [10], [11]. Our method enables classification of trajectories into success, known failure, or OOD. Each model is calibrated with its own calibrated band: one based on success data and one based on failure data. However, as acknowledged by previous work [11], distribution shifts can arise from environmental conditions, operator behavior, or hardware degradation. Then, bands calibrated on earlier data may no longer reflect deployment conditions, leading to misestimated detection rates without adaptive recalibration.

1) Calibration: For each trained model, we create two conformal prediction bands from separate calibration sets of success and failure data. We use a maximum score per trajectory instead of timestep-level scoring, so that decisions are not affected by temporal correlations in a trajectory. A visual representation of this process is shown in Figure 5. Consistent with previous CP anomaly detection work [11], since we cannot assume CP validity guarantees without perfect calibration and test data exchangeability, we set

TABLE I: CP scoring methods, grouped by category, from CP literature [36] and prior CP thresholding work [11]. For all metrics in this table, values higher than the calibrated threshold indicate a potential out-of-distribution sample.

Method	**Type**	**Description**
Reconstruction error	Residual-based	Pixel-level error between input and reconstruction.
Latent Prediction error	Residual-based	Deviation between latent representations of predicted and actual future frames.
Latent standard deviation	Residual-based	Temporal standard deviation of latent embeddings.
Mahalanobis distance	Distance-based	Squared distance of latent embedding from the estimated Gaussian distribution.
Latent distance (L2)	Distance-based	Euclidean distance between latent embedding and calibration mean.
Latent cosine distance	Distance-based	Complement of cosine similarity (directional alignment of latent vectors).
Training loss	Miscellaneous	Model's per-sample training loss function.

thresholds at a chosen quantile level $(1 - \alpha)$. Because thresholds are set below the maximum calibration trajectory score, some false negatives are expected, but this trade-off minimizes false positives, which is more critical for safe deployment. To assess metric suitability, we visually compare success and failure score distributions, where substantial overlap indicates poor separation.

We test multiple CP residual and distance-based scoring metrics, as described in Table I from the CP literature [36]. We cannot use action-dependent metrics [10] due to teleoperation. We also compare against training loss since it indicates how well the model recognizes in-distribution data, but does not amplify the separation between that and OOD data. Since the models predict the next frame, we assign scores to frame pairs and summarize each trajectory by the maximum frame-pair score. For efficiency in real-time deployment, we adopt constant-value bands, similar to the prior work in failure detection with world models [11], rather than time-varying bands [10], and trim videos to equal length so no sample dominates calibration.

2) Evaluation: At test time, we score new videos drawn from success, failure, or OOD test sets and classify them using the decision function from Equation (1). If the distributions are separate, a success will only fall within the success band and outside the failure band, a failure will only fall within the failure band and outside the success band, and an OOD sample will fall outside of both bands. Calibrating each band and classifying one test trajectory data set takes $\sim$1.5 hours. We demonstrate our framework on both offline video streams taken from the robot as well as on hardware.

IV. Experimental Results

We use classification accuracy and detection time as our success metrics, like Xu et al. [10], and evaluate our framework on various conformal prediction metrics and thresholding quantiles. We plot histograms to assess each metric by the separation between distributions.

A. Detection Accuracy

Among the results shown in Table II, Mahalanobis distance achieved the highest OOD detection accuracy (100% across thresholds) but was prone to overfitting to the calibration set. In high-dimensional settings, covariance inversion can be unstable, which can lead to misclassifying valid but visually different successes as OOD. Regularization (shrinkage, PCA) reduced this instability but collapsed distinctions between nominal and OOD. Latent prediction error proved more reliable, achieving over 90% accuracy across classes at the 90% quantile. Unlike Mahalanobis, it does not rely on covariance inversion, instead measuring discrepancies between predicted and actual latent states. By contrast, L2 distance and latent standard deviation underperformed. L2 distance ignores covariance and temporal context, while standard deviation captures variability but not dynamic progression with time. Reconstruction error and latent cosine difference performed the worst. Since Cosmos was originally trained for reconstruction, it left little anomaly signal even for a failure video. Performance improved when inputs were more compressed (Table IV). For the latent cosine difference, the high-dimensional latent space lacked a consistent dominant direction, so the average vector was poorly defined. Training loss classified the nominal data well, as expected. Since it is not designed to amplify anomalies, this metric resulted in

TABLE II: Detection accuracy (%) of all three classes: success, known failure, and OOD, for three different quantiles. Three models were trained for both success and failure. Detection rates were consistent, except the reconstruction error for detecting failures improved by one trajectory.

Metric	OOD (Succ)	OOD (Fail)	OOD (Total)	Fail (Total)	Succ (Total)
Reconstruct. error	52.83	49.06	49.06	48.65	40.00
Latent pred. error	94.34	90.57	**90.57**	**91.89**	**91.11**
Latent std dev.	64.15	52.83	52.83	59.46	33.33
Mahalanobis	100.00	100.00	100.00	90.00	90.00
Latent dist. (L2)	75.47	54.72	54.72	81.08	40.00
Latent cosine dist.	45.28	43.40	39.62	67.57	42.22
Training Loss	90.57	84.91	84.91	89.19	80.00

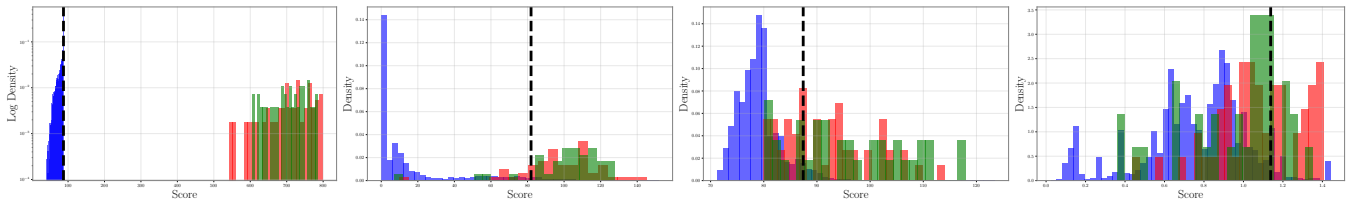
(a) 90% threshold

Metric	OOD (Succ)	OOD (Fail)	OOD (Total)	Fail (Total)	Succ (Total)
Reconstruct. error	35.85	45.28	35.85	18.92	37.78
Latent pred. error	86.79	77.36	**77.36**	**89.19**	**75.56**
Latent std dev.	56.60	30.19	30.19	43.24	6.67
Mahalanobis	100.00	100.00	100.00	95.00	95.00
Latent dist. (L2)	64.15	37.74	37.74	72.97	17.78
Latent cosine dist.	39.62	35.85	35.85	51.35	28.89
Training Loss	75.47	54.72	54.72	75.68	51.11

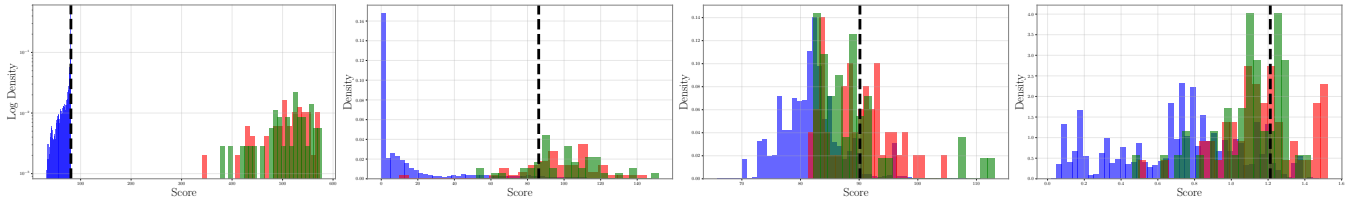
(b) 95% threshold

Metric	OOD (Succ)	OOD (Fail)	OOD (Total)	Fail (Total)	Succ (Total)
Reconstruct. error	18.87	16.98	16.98	16.98	0.00
Latent pred. error	0.00	7.55	0.00	0.00	8.89
Latent std dev.	0.00	2.70	0.00	2.70	0.00
Mahalanobis	100.00	100.00	100.00	100.00	100.00
Latent dist. (L2)	0.00	0.00	0.00	0.00	4.44
Latent cosine dist.	0.00	0.00	0.00	0.00	6.67
Training Loss	0.00	9.43	0.00	0	7.55

(c) 100% threshold



(a) Mahalanobis Success (b) Latent Pred. Error Success (c) Latent L2 Distance Success (d) Reconstruction Success



(e) Mahalanobis Failure (f) Latent Pred. Error Failure (g) Latent L2 Distance Failure (h) Reconstruction Failure

Fig. 6: Nominal vs. OOD distributions scored by success and failure models on select metrics, with the 95% quantile threshold shown with the black dashed line. Legend: (Nominal, OOD, Succ/Fail - Opposite Nominal)

weaker separation, notably at the 95% threshold.

The histograms of score distributions in Figure 6 support the above results. Mahalanobis showed clean separation, while most showed overlap. This observation suggests the importance of quantile threshold choice. Because most metric distributions overlap, shifting the threshold inevitably misclassifies some portion of nominal or OOD data. Lower quantiles increased OOD accuracy but reduced nominal accuracy, while higher quantiles had the opposite effect. In safety-critical domains, false positives are preferable to false negatives. In our case, the 100% quantile was overly conservative, while 90% achieved the best balance, with latent prediction error classifying ∼91% correctly across all three classes.

Finally, the success model consistently encompassed the detections made by the failure model, i.e., it classified all the same OOD cases. This pattern likely occurred since the failure model was exposed to diverse failure types, while the success model was trained on a single, more consistent case, yielding better separation.

B. Detection Time

Another success metric that we considered was the difference between the true failure detection time, as observed by a human, as opposed to the time at which the framework made a classification. The results can be seen in Table III.

TABLE III: Difference in detection time ($\hat{k}$), for each class given the model used. Results are evaluated at the 95% quantile and presented as mean detection time in seconds $\pm$ standard error.

Metric	OOD (Succ)	OOD (Fail)	Success (Fail)	Failure (Succ)
Reconstruct. error	-2.059 ± 0.583	-2.133 ± 0.611	-2.570 ± 0.692	-1.648 ± 0.887
Latent pred. error	-1.844 ± 0.406	-2.517 ± 0.490	-3.526 ± 0.452	-3.440 ± 0.533
Latent std dev.	-1.313 ± 0.554	-2.247 ± 0.838	-3.949 ± 0.891	-2.728 ± 0.874
Mahalanobis	-5.974 ± 0.438	-5.974 ± 0.438	-9.005 ± 0.191	-8.637 ± 0.618
Latent dist. (L2)	-1.686 ± 0.565	-2.669 ± 0.796	-3.737 ± 0.679	-2.217 ± 0.807
Latent cosine dist.	-3.536 ± 0.697	-4.052 ± 0.848	-1.802 ± 0.855	-1.968 ± 0.790
Training Loss	1.949 ± 1.163	2.090 ± 1.054	0.663 ± 1.552	0.934 ± 1.115

Nearly all metrics were able to predict the class before they became apparent to a human observer. This pattern indicates that the metrics are sensitive to underlying visual or latent patterns that precede observable failure modes. The most successful metric for accuracy, latent prediction error, made classifications on average between 1 and 3 seconds earlier than a human observer and almost consistently had the lowest standard error. An example run over a 10-second period can be seen in Figure 7. Mahalanobis was the earliest, but since it is prone to overfitting, it reduces interpretability and increases the likelihood of false positives. The training loss was the only metric that detected failures later than the human observer, since the loss function amplifies nominal patterns and lacks any contrastive signal for anomalies.

C. Hardware Results

We validate our framework on a hardware platform, seen in Figure 7, demonstrating its ability to operate in real time. We deployed a Boston Dynamics Spot, equipped with LiDAR, cameras for navigation [39], and a Pan-Tilt-Zoom (PTZ) camera for detecting gauges, which moves independently of the robot's pose. Due to shared-use restrictions, we were unable to host the models directly on board the robot. Instead, inference was hosted externally. However, the full pipeline only requires at most ∼1.5 GB of storage, dominated by the pretrained Cosmos weights (∼1.3 GB), with our class-specific models (∼60 MB) and auxiliary CP artifacts (thresholds, precomputed values, $<$ 5 MB), so we suspect it can be deployed onboard. We experimented with several gauges placed in an office.

At runtime, the PTZ camera captures two frames and processes them through the framework (Figure 1). In the case of a success, the robot remains idle until commanded to proceed to the next gauge. For a failure, the system records the affected gauge, which will be revisited after all gauges have been surveyed. For OOD inputs due to a distant, blurry reading, the robot initiates a zoom operation and reprocesses

(a) Succ. Detection with Succ. Model

(b) OOD Detection with Succ. Model

Fig. 7: Hardware demonstration of framework for example success and OOD cases. Using latent prediction error, the scores were tracked with respect to the threshold over 10 seconds. The OOD case exceeds the threshold as expected.

the new frames through the pipeline. A single classification run requires approximately three minutes, primarily due to network latency from passing data to the externally hosted models. The actual scoring is comparably instantaneous when using a precomputed threshold. While this latency is acceptable in the current application, higher-risk tasks would benefit from hosting the model locally to enable immediate detection and intervention. Nevertheless, we designed our framework to employ lightweight models, observation-based metrics that do not require auxiliary networks, and constant-valued thresholds, successfully demonstrating its abilities on a real hardware platform.

V. Conclusion

We presented a framework for anomaly detection and failure classification in the context of industrial gauge inspection using a world model backbone and conformal prediction thresholding. Our experiments demonstrated that we can separate between success, failure, and OOD classes at over 90% accuracy, earlier than a human observer, highlighting the potential of the method for robust runtime monitoring for real-time inspection tasks. Beyond gauge inspection, such a framework can have a broader impact in enabling safer, more reliable deployment of autonomous systems in safety-critical industrial domains and identifying gaps in training data to guide targeted collection and retraining.

Our work has several opportunities for future improvement. We can address the distribution shifts between the calibration and test data with martingales [40]. Our current CP bands are static, and developing adaptive, time-varying bands could improve responsiveness to temporal drift. We only explored manual dimensionality reduction, so leveraging compression modules like singular value decomposition may enhance efficiency and accuracy. Incorporating semantic failure detection and more history could enable recognition of higher-level failures that reflect human-like understanding. Finally, training a single model with a classification head, rather than storing multiple models, can further decrease the computational overhead required to run the detector online.

References

[1] Boston Dynamics, "3 ways mobile robots improve industrial inspections," 2024. [Online]. Available: https://bostondynamics.com/blog/3-ways-mobile-robots-improve-industrial-inspections/

[2] A. Shukla and H. Karki, "Application of robotics in onshore oil and gas industry—A review Part I," *Robotics and Autonomous Systems*, vol. 75, pp. 490–507, 2016.

[3] C. Gehring, P. Fankhauser, L. Isler, R. Diethelm, S. Bachmann, M. Potz, L. Gerstenberg, and M. Hutter, "ANYmal in the Field: Solving industrial inspection of an offshore HVDC platform with a quadrupedal robot," in *Field and Service Robotics*, G. Ishigami and K. Yoshida, Eds., 2021, pp. 247–260.

[4] M. F. Ginting, S.-K. Kim, D. D. Fan, M. Palieri, M. J. Kochenderfer, and A. akbar Agha-mohammadi, "SEEK: Semantic reasoning for object goal navigation in real world inspection tasks," in *Robotics: Science and Systems*, 2024.

[5] Field AI, "Technology | Field AI," 2025. [Online]. Available: https://www.fieldai.com/technology

[6] E. Robotics, "Automate inspection with advanced robots," 2025. [Online]. Available: https://www.energy-robotics.com/inspection-robots

[7] ANYbotics, "Automate inspection across industries," 2025. [Online]. Available: https://www.anybotics.com/industries/

[8] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault tolerant techniques —part I: Fault diagnosis with model-based and signal-based approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[9] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[10] C. Xu, T. K. Nguyen, E. Dixon, C. Rodriguez, R. Lee, P. Miller, P. Shah, R. Ambrus, H. Nishimura, and M. Itkina, "Can we detect failures without failure data?: Uncertainty-aware runtime failure detection for imitation learning policies," in *Robotics: Science and Systems*, 2025.

[11] I. R. Ward, M. Ho, H. Liu, A. Feldman, J. Vincent, L. Kruse, S. Cheong, D. Eddy, M. J. Kochenderfer, and M. Schwager, "Foundational world models accurately detect bimanual manipulator failures," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

[12] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. Springer, 2009.

[13] J. Nitsch, M. Itkina, R. Senanayake, J. Nieto, M. Schmidt, R. Siegwart, M. J. Kochenderfer, and C. Cadena, "Out-of-Distribution detection for automotive perception," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2021, pp. 2938–2943.

[14] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[15] T. Defard, A. Setkov, A. Loesch, and R. Audigier, "PaDiM: A patch distribution modeling framework for anomaly detection and localization," in *Pattern Recognition. ICPR International Workshops and Challenges*, A. Del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante, and R. Vezzani, Eds., vol. 12664, 2021, pp. 475–489.

[16] Z. Xiao, Q. Yan, and Y. Amit, "Likelihood regret: An out-of-distribution detection score for variational auto-encoder," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 20 685–20 696.

[17] Z. Wang, B. Dai, D. Wipf, and J. Zhu, "Further analysis of outlier detection with deep generative models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 8982–8992.

[18] P. Izmailov, P. Kirichenko, M. Finzi, and A. G. Wilson, "Semi-

supervised learning with normalizing flows," in *International Conference on Machine Learning (ICML)*, 2020, pp. 4615–4630.
[19] I. R. Ward, M. Paral, K. Riordan, and M. J. Kochenderfer, "Improving the resilience of quadrotors in underground environments by combining learning-based and safety controllers," in *Conference on Control, Decision and Information Technologies*, 2025.
[20] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" in *International Conference on Robot Learning*, 2019.
[21] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, 2017.
[22] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International Conference on Machine Learning (ICML)*, 2015, pp. 1613–1622.
[23] M. A. A.Alobaidy, D. J. M. Abdul-Jabbar, and S. Z. Al-khayyt, "Faults diagnosis in robot systems: A review," *Al-Rafidain Engineering Journal (AREJ)*, vol. 25, no. 2, pp. 164–175, 2020.
[24] S. Daftry, S. Zeng, J. A. Bagnell, and M. Hebert, "Introspective perception: Learning to predict failures in vision systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1743–1750.
[25] B. Lussier, M. Gallien, J. Guiochet, F. Ingrand, M.-O. Killijian, and D. Powell, "Fault tolerant planning for critical robots," in *International Conference on Dependable Systems and Networks*, 2007, pp. 144–153.
[26] N. Basha, M. Nounou, and H. Nounou, "Multivariate fault detection and classification using interval principal component analysis," *Journal of Computational Science*, vol. 27, pp. 1–9, 2018.
[27] P. Hofmann and Z. Tashman, "Hidden Markov models and their application for predicting failure events," in *International Conference on Computational Science*, Cham, 2020, pp. 464–477.
[28] D. Maincer, Y. Benmahamed, M. Mansour, M. Alharthi, and S. S, "Fault diagnosis in robot manipulators SVM and KNN," *Intelligent Automation & Soft Computing*, vol. 35, no. 2, pp. 1957–1969, 2022.
[29] M. A. Costa, B. Wullt, M. Norrlöf, and S. Gunnarsson, "Failure detection in robotic arms using statistical modeling, machine learning and hybrid gradient boosting," *Measurement*, vol. 146, pp. 425–436, 2019.
[30] A. Elhafsi, R. Sinha, C. Agia, E. Schmerling, I. A. Nesnas, and M. Pavone, "Semantic anomaly detection with large language models," *Autonomous Robots*, vol. 47, no. 8, pp. 1035–1055, 2023.
[31] R. Sinha, A. Elhafsi, C. Agia, M. Foutter, E. Schmerling, and M. Pavone, "Real-time anomaly detection and reactive planning with large language models," in *Robotics: Science and Systems*, 2024.
[32] P. Oza and V. M. Patel, "C2ae: Class conditioned auto-encoder for open-set recognition," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[33] C. Agia, R. Sinha, J. Yang, Z.-a. Cao, R. Antonova, M. Pavone, and J. Bohg, "Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress," in *Conference on Robot Learning (CoRL)*, 2024.
[34] S. X. Ding, "Residual generation with enhanced robustness against unknown inputs," in *Model-based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools*. Springer, 2008, pp. 161–246.
[35] K. Nakamura, L. Peters, and A. Bajcsy, "Generalizing safety beyond collision-avoidance via latent-space reachability analysis," in *Robotics: Science and Systems*, 2025.
[36] A. N. Angelopoulos and S. Bates, "Conformal prediction: A gentle introduction," *Foundations and Trends in Machine Learning*, vol. 16, no. 4, pp. 494–591, 2023.
[37] X. Zhou, B. Chen, Y. Gui, and L. Cheng, "Conformal prediction: A data perspective," *ACM Computing Surveys*, p. 3736575, May 2025.
[38] N. Agarwal *et al.*, "Cosmos world foundation model platform for physical ai," *arXiv preprint arXiv:2501.03575*, 2025.
[39] A. Bouman, M. F. Ginting, N. Alatur, M. Palieri, D. D. Fan, T. Touma, T. Pailevanian, S.-K. Kim, K. Otsu, J. Burdick, and A.-a. Agha-Mohammadi, "Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2518–2525.
[40] R. Luo, R. Sinha, Y. Sun, A. Hindy, S. Zhao, S. Savarese, E. Schmerling, and M. Pavone, "Online distribution shift detection via recency prediction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 16 251–16 263.

A. Acknowledgements

Generative AI tools and technologies were used in this work solely for validating ideas in the brainstorming stage (ChatGPT), tab-autocompleting while extending prewritten code to new metrics, with all methods and analyses otherwise unchanged (Cursor using Claude), and for figure placement and table creation in LaTeX (ChatGPT).

APPENDIX

A. Low Resolution Results

We ran additional experiments to observe if higher image compression before the world model would affect results. Table IV depicts the results from 50% higher compression.

TABLE IV: Detection accuracy (%) across thresholds at 50% higher image compression (1200×700 to 256×144).

Metric	OOD (Succ)	OOD (Fail)	OOD (Total)	Fail (Total)	Succ (Total)
Reconstruct. error	60.38	49.06	45.28	51.35	42.22
Latent pred. error	96.23	94.34	94.34	91.89	91.11
Latent std dev.	64.15	52.83	52.83	59.46	17.78
Mahalanobis	100.00	100.00	100.00	100.00	90.00
Latent dist. (L2)	75.47	67.92	66.04	81.08	57.78
Latent cosine dist.	52.83	24.53	24.53	64.86	20.00

(a) 90% threshold

Metric	OOD (Succ)	OOD (Fail)	OOD (Total)	Fail (Total)	Succ (Total)
Reconstruct. error	39.62	41.51	33.96	32.43	35.56
Latent pred. error	88.68	77.36	77.36	89.19	80.00
Latent std dev.	49.06	30.19	30.19	43.24	17.78
Mahalanobis	100.00	100.00	100.00	100.00	95.00
Latent dist. (L2)	60.38	32.08	32.08	72.97	13.33
Latent cosine dist.	35.85	20.75	20.75	37.25	13.33

(b) 95% threshold

Metric	OOD (Succ)	OOD (Fail)	OOD (Total)	Fail (Total)	Succ (Total)
Reconstruct. error	7.55	22.64	7.55	2.70	6.67
Latent pred. error	0.00	7.55	0.00	0.00	8.89
Latent std dev.	0.00	0.00	0.00	2.70	0.00
Mahalanobis	100.00	100.00	100.00	100.00	100.00
Latent dist. (L2)	0.00	0.00	0.00	0.00	6.67
Latent cosine dist.	0.00	0.00	0.00	2.70	0.00

(c) 100% threshold

B. Reproducibility

The code and subset of the data required to reproduce these results are available at `https://autoinspection-classification.github.io/`. Please see below for the GPU architecture and training configurations for the models.

TABLE V: Model Training Configurations

Optimizer	AdamW with weight decay (10^{-4})
Batch size	Default 32 (configurable)
Learning rate	Defined in hyperparameters
Gradient clipping	Value of 1.0
Early stopping	Patience 15 epochs, $\Delta = 10^{-5}$
Check-pointing	Minimum validation loss
Epochs	Configurable (default: 10)
Device selection	GPU with CUDA, fallback to CPU
Data workers	Training: 3, Validation: 1 (default)

All simulations were executed on an g4dn.xlarge Ubuntu EC2 Amazon Web Services Instance with 100 GiB of storage, 16 GiB RAM, and 4 CPU cores.