Identifying Precursors to Failures in Robotic Lift-and-Place Tasks

Zeyu Shangguan, Rajas Chitale, Rutvik Patel, Satyandra K. Gupta, Daniel Seita
University of Southern California
{zshanggu, rachital, rutvikra, guptask, seita}@usc.edu

***Abstract*—Failure prediction for robotic manipulation in industrial applications has experienced substantial advancements in terms of efficiency and reliability driven by the latest innovations in machine learning. Most of the existing works focus on reactive failure prediction. As an offline analyzing technique, it is not suitable for real-time failure prevention. Therefore, proactive failure prediction becomes a valuable approach to meeting the requirements of online deployment. Although recent research on proactive failure prediction has made progress, it still suffers from limitations such as object specificity, fixed action spaces, dynamic action, and high system complexity. In this paper, we study lift-and-place tasks, and propose a more effective approach to identifying failure by analyzing the relative motion of target objects in the scene. Consequently, we propose a novel method that proactively predict the failure by embedding the relative motions in the scene. We verify our proposed method on both simulation and real-world data. The experimental results indicate that our proposed method not only demonstrates improved generalization but also provides a more precise response to the precursors of failure.**

I. Introduction

Failures in robotic manipulation can result in serious safety hazards, posing physical harm to both humans and equipment [12, 6]. Therefore, it is imperative that robotic systems autonomously identify and assess high-risk situations during manipulation. We study one of the most fundamental and widely encountered tasks in industry: lift-and-place. A significant challenge in lift-and-place tasks commonly involves the target object dropping from the robot's carrier during manipulation, especially when the robot is executing dynamic action. Such failures can disrupt workflow, damage materials, and compromise functionality. Addressing these failure scenarios is therefore critical for enhancing the robustness and overall efficiency of robot manipulation.

During execution, failure prediction tasks typically fall into two categories: **reactive** failure prediction and **proactive** failure prediction [8, 1, 20]. Reactive failure prediction refers to offline analysis that categorizes manipulation failures after they occur, typically for the purpose of explanation or retrospective correction. Common methods for addressing reactive failure prediction include learning a failure classifier [17] and using large language models to explain failures [3, 9]. In contrast, proactive failure prediction relies on predicting future risks to enable real-time corrections. Approaches to proactive failure prediction often involve out-of-distribution detection [19, 16] and analyzing object geometric relationships [20, 5]. Compared to reactive failure analysis, which

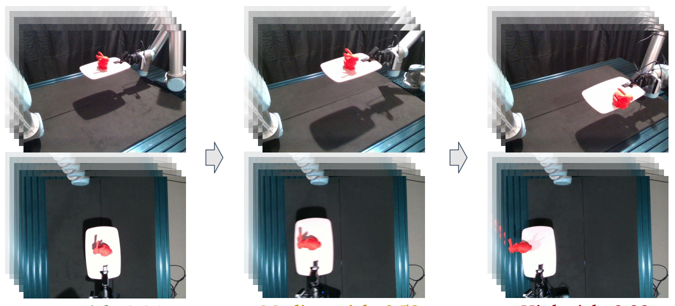

Fig. 1. Our method predicts a continuous risk value to represent the likelihood of a real-time failure in robotic lift-and-place tasks. The key insight is to analyze the relative motion between the target object (bunny) and the carrier (tray) from visual information. In the above example, when the relative motion is small, the predicted risk value is low (0.04). As the relative motion increases, the predicted risk value also goes up (0.52), suggesting that a failure may soon occur. When the motion becomes even larger, the predicted risk value rises to 0.92, resulting in inevitable failure. At test time, a robot can reduce its speed once our model predicts a risk value above a threshold.

analyzes the entire manipulation process, proactive failure prediction relies on the observations available *before* a failure occurs. This makes proactive prediction significantly more challenging and relatively less explored. In this paper, we focus on methods for proactive failure prediction.

In addition, a key challenge in proactive failure prediction in lift-and-place tasks is defining a generalized feature representation of failures that enables the model to perform well on unseen target objects. To address this, a prior study proposes to represent failures by analyzing geometric constraints in a scene [20], where a failure is likely to occur when these constraints are violated. However, their method requires sequential inference from multi-level generation models, which increases the complexity of the pipeline. We hypothesize that a critical factor in understanding failure cases lies in accurately analyzing the relative motion between target objects and carriers. To this end, we propose a two-step strategy. The first step leverages visual observations to extract motion values. The second step processes these values using a deep neural network, which outputs a continuous "risk" value representing the likelihood of failure. See Fig. 1 for an example.

Additionally, we implement a novel data collection pipeline capable of labeling risk values during lift-and-place tasks. It records key data points around the moment when failure happens throughout the manipulation process. The pipeline collects both the observations of the environment and the actions performed by the robot at each time step, providing a detailed chronological record of the manipulation sequence. At every time step, the pipeline calculates and assigns a risk value that serves as a quantitative indicator of the likelihood of manipulation failure, helping to pinpoint moments when the system operates under conditions of heightened uncertainty or instability. By enabling proactive risk assessment, our method achieves high success rates in early detection of high-risk scenarios, allowing for timely interventions.

Our main contributions include:

- A novel scene representation using the relative motion of objects, for generalizable proactive failure prediction.
- Experiments demonstrating that our proposed method adapts to unseen objects and demonstrates better performance compared to a baseline method.

II. RELATED WORKS

A. Failure Cases During Manipulation

In general, failure cases in robotic manipulation can be categorized into two categories: **task planning failures** and **task execution failures** [2]. Task planning failure refers to the case where the model cannot generate a valid plan due to an unsatisfactory scene setup, such as missing target objects, which leads to failed motion planning. In contrast, task execution failure refers to an accident that occurs during robot execution due to improper planning. We study the task execution failures.

There are two distinct *types* of task execution failures: **primary object failure** and **secondary object failure**, with each encompassing two *phases*: **the manipulation phase** and **the post-manipulation phase**. Primary object failure directly affects the object in contact with the robot, while secondary failure involves objects indirectly influenced. Manipulation-phase failures occur during interaction, while post-manipulation failures happen afterward. According to prior research [6], understanding the dynamics of these phases is crucial for designing robust systems. In this paper, we concentrate on secondary object failures that occur during the manipulation phase of the lift-and-place task, where the failure happens happens at the contact between the object and the carrier.

B. Failure Identification

Current research on failure identification in robotic manipulation can be broadly categorized into two approaches: **reactive failure analysis** (post-hoc, offline) and **proactive failure prediction** (ex-ante, online).

For reactive analysis methods, Arda et al. [6] frame failure identification as a classification problem that uses multimodal scene features as input and outputs predefined failure categories. Yuliang et al. [11] also propose a failure classifier that enhances robustness to camera faults such as noise and blur. REFLECT [9] leverages a vision-language model to assist in failure analysis by generating textual explanations of failure causes. AHA [3] uses a large language model to identify specific failure modes and provide textual explanations. Jeon et al. [7] design a model that combines multimodal scene features with recovery procedures, using an LLM to generate tailored recovery codes and link failure analysis with corrective planning. Pradip et al. [13] propose to extend the explanation of robot failure in a multi-modal way. Despite progress, their post-hoc nature limits use in industrial settings, where proactive prediction and prevention are crucial.

For proactive failure prediction, Enshen et al. [20] propose using geometric constraints for real-time failure prediction, but their multi-level generation models increase system complexity. Chen et al. [19] formulate failure detection as an out-of-distribution problem to address the limitations of data collection, but might produce false positives in OOD cases. RoboFail [15] generates action probability distributions to assess risk without explicit failure labels, but its fixed action space limits flexibility. Ping et al. [5] use machine learning to predict maximum object shift during manipulation, assigning risk scores to scenarios. However, this method is limited to the wafer transferring scenario. RoboFail [15] and Ping et al. [5] are the most relevant related works, where the precursor of failure is represented as a scalar value. However, we focus on a more generative and dynamic scenario.

Our research focuses on proactive failure prediction problem. Although previous works have significantly advanced proactive failure prediction, it still suffers from limitations such as dependence on specific objects, limited action spaces, and high system complexity.

III. PROBLEM FORMULATION

We define proactive failure prediction for the lift-and-place task as a regression problem, which takes a sequence of visual observations as raw input, noted as $O_{raw} = \{O^0_{raw}, O^1_{raw}, \ldots, O^n_{raw}\}$, and outputs a risk value R_t. We assume that failure occurs only once per trajectory, as any subsequent failures do not affect the overall safety of the situation. The observation $O^t_{raw} = \{I, EE_{pos}\}$ contains image frames from multiple camera sources at the current time step, and the position of the end-effector (EE_{pos}).

IV. METHODS

Our method is two-fold, as indicated in Fig. 2. The first step is to analyze the relative motion between the target object and the carrier. The kinematic state $M = \{\theta, T_x, T_y\}$ includes relative rotation and translation values. Then at the second step, we will use the collected kinematic states to predict failure proactively.

A. Data Labeling: Historical Backtracking

In this paper, we design a data collection pipeline that labels each time step with a risk value, $R_t \in [0, 1]$, during manipulation. We propose a historical backtracking method for data collection in simulation. The idea is to identify two key moments within a trajectory: the last possible moment to trigger an emergency stop (e-stop) t_{stop}, and the moment when the failure actually occurs t_{fail}. We note the start of the manipulation as t_0.

In our lift-and-place case, the t_{fail} is the moment when the target object starts dropping from the carrier. The t_{stop} is

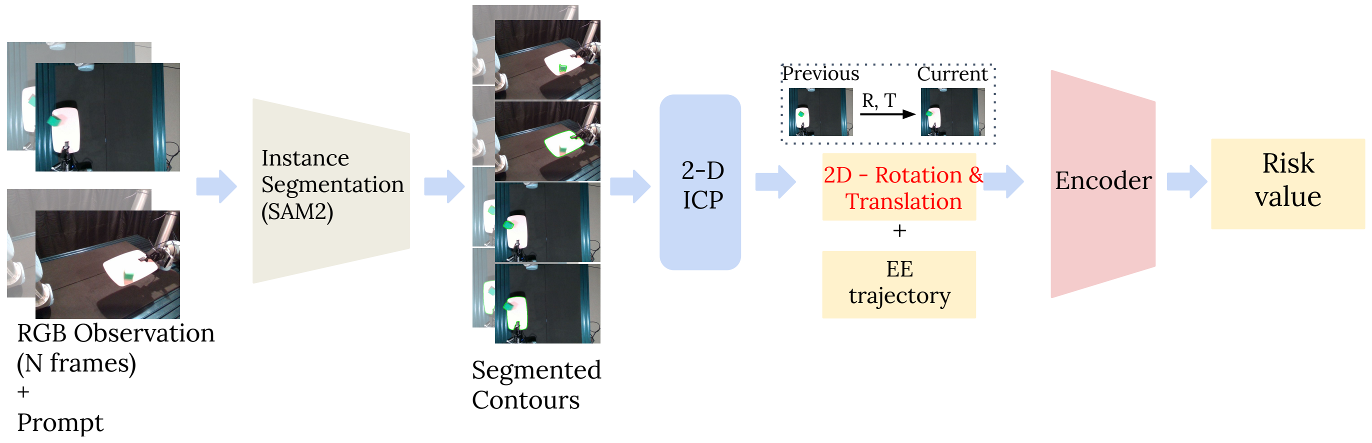

Fig. 2. Our proposed method consists of two stages. First, it processes a series of images captured from multiple camera views to extract the segmented contours of the target object and the carrier. Then, a 2-D ICP algorithm computes the motion values between consecutive time steps based on these contours. In the second stage, the calculated motion values, together with the recorded current end-effector trajectory, are input to a feature encoder, which outputs a continuous risk value indicating the likelihood of failure.

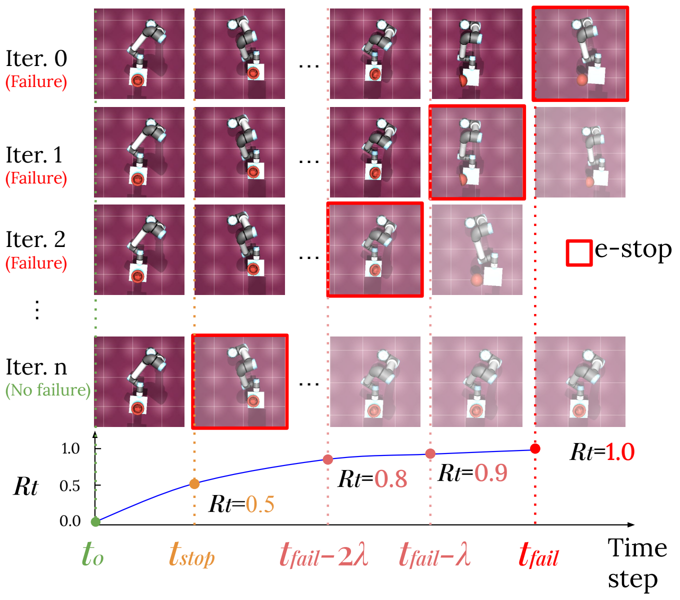

Fig. 3. Historical backtracking data collection pipeline. The above part demonstrates the process of searching backward from the actual failure moment t_{fail} in steps of λ. At each moment $t_{fail} - i \cdot \lambda$ (where $i = 0, 1, 2, \ldots$), we apply an e-stop and observe whether failure still occurs in the following time steps (indicated by the translucent color). The bottom part is the labeled ground truth curve of the risk value, $R_t = 0.5$ at the moment t_{stop}, $R_t = 1.0$ at the moment t_{fail}.

determined by our proposed historical backtracking method. The key idea is to identify the earliest safe intervention point by iteratively searching backward from the actual failure moment t_{fail} in steps of λ, starting at $t_{fail} - \lambda$, then $t_{fail} - 2\lambda$, followed by $t_{fail} - 3\lambda$, and so forth. At each earlier time step $t_{fail} - i * \lambda (i = 0 = 1, 2, \ldots)$, an emergency stop (e-stop) is applied to determine if the failure can be avoided, since the object will still move due to its inertia and contact friction. If not, this process continues.

For example, in Fig. 3, the left section shows a curve collected from the simulator (we use MuJoCo [18]). The curve in the right section represents the robot trajectory. From top to bottom is the process of our historical backtracking exploration method. For Iteration 0 in Fig. 3, we apply an emergency stop (e-stop) when we observe a failure. Therefore, from that moment onward, the risk value is always $R_t = 1.0$. Then for Iteration 1, we replay the trajectory but apply the e-stop at $t_{fail} - \lambda$. If the failure still occurs, then this is a high-risk situation. Then, for Iteration 2 and all subsequent ones, we continuously advance the e-stop by λ time. Eventually, after n rounds, we find a moment $t_{stop} = t_{fail} - n\lambda$ at which applying e-stop prevents failure, and we define the risk value as $R_t = 0.5$. Based on this, we assign all time steps before t_{stop} to represent low-risk situations where failure can be avoided, while the time steps after t_{stop} represent high-risk situations with unavoidable failures. Then, we apply Hermite splines [4] to interpolate the risk values at other time points.

In the real-world, it is infeasible to continually reset and perform historical backtracking. Therefore, we develop a GUI to manually label real-world data. The times t_0 and t_{fail} are labeled as the start of the manipulation and the actual failure moment, respectively. The time t_{stop} is labeled manually when a human observes substantial relative motion between the target object and the carrier.

Our data post-processing involves a **hard cut-off** and **uneven sampling**, as seen in Fig. 4. For the hard cut-off, we remove time steps following each failure. This is because the relative motion is minimal or absent both when the robot is moving steadily and after a failure, when the object remains stationary on the ground and the robot stays fixed. In such cases, the model may struggle to distinguish between these scenarios solely on such homogeneous relative motion. Furthermore, post-failure data is not informative for proactive failure prediction and is therefore excluded from training. Our raw data is also naturally imbalanced, as failures are rare and occur only briefly during manipulation, meaning most training data represents low-risk cases. To mitigate this imbalance, we sample fewer low-risk instances. In each manipulation trajectory, we downsample the data before t_{stop} using a stride

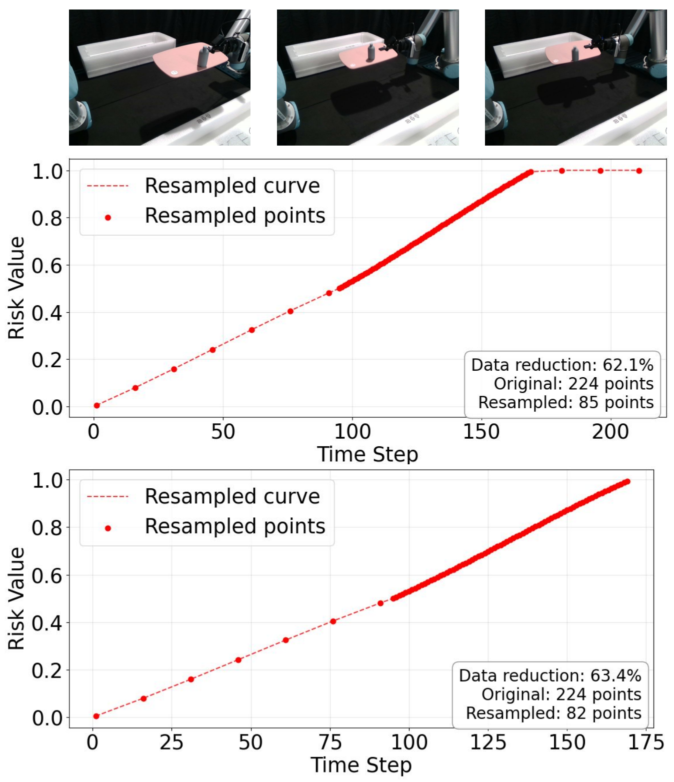

Fig. 4. Visualization of the data post-processing for a real-world data. The top three figures of the manipulation scene illustrate the moments t_0, t_{stop}, and t_{fail}, from left to right. We resample the data points to save more data points with high-risk label while reducing the proportion of the low-risk label. The original data is indicated by the red dotted line, while the resampled data points are marked by red circles. In addition, we apply a hard cut-off to remove data points after the failure occurs (around the 170-th time step). Through resampling, we reduce the number of data points by 62.1%. The hard cut-off further reduces the data by an additional 1.3%.

	τ^{gt}_{fail}	$\tau^{gt}_{no-fail}$
τ^{pred}_{fail} & $(t^{pred}_{stop} \leq t_{stop})$	TP	FP
τ^{pred}_{fail} & $(t^{pred}_{stop} > t_{stop})$	FP	
$\tau^{pred}_{no-fail}$	FN	TN

TABLE I
DEFINITION OF TRUE POSITIVE, TRUE NEGATIVE, FALSE POSITIVE, AND FALSE NEGATIVE.

of 2. In addition, our data post-processing method significantly reduces the data volume. For example, as shown in Fig. 4, resampling alone reduces the data by 62.1%, and the reduction increases to 63.4% when hard cut-off is additionally applied.

B. Network Design

As shown in Fig. 2, the idea of our network is to first analyze the object motion from the RGB images, and then predict the risk value. Intuitively, large relative motion indicates high risk. However, this motion may be unobservable from certain viewpoints, such as if the object moves along the camera's line of sight. To address this, we use two cameras at different locations. In addition, robot manipulation is a dynamic process, so we analyze failure based on a sequence of RGB images from each camera captured within a time window.

We will first process the raw input to get the motion values of the target object (M_{obj}) and the carrier (M_{carr}) from multiple camera views between each two time step. Both M_{obj} and M_{carr} will include a rotation value and a translation value (T). And then we get $O_{proc} = \{O^0_{proc}, O^1_{proc}, \ldots, O^{n-1}_{proc}\}$, where $O^t_{proc} = \{M_{obj}, M_{carr}, EE_{pos}\}$. Then we will process these processed inputs with a sampling window of length m, thus, the observation window would be $O_{win} = \{O^i_{win}, O^{i+1}_{win}, \ldots, O^{i-1+m}_{win}\}$, where each O^t_{win} corresponds to a ground truth risk value $R^{gt}_t \in [0, 1]$.

One direct approach to analyzing relative motion is to model the dynamics of object interactions. However, this requires extensive scene modeling and estimation of physical properties of both the object and the robot, which can be unreliable—especially for properties like friction and surface texture. Therefore, we propose analyzing relative motion by estimating rotation and translation based on contour movement. We first get the geometric contours of the target object and the carrier in both camera views ($\{c^{obj,cam1}_i, c^{obj,cam2}_i, c^{carr,cam1}_i, c^{carr,cam2}_i\}$) at each time step i. In the real world, we do this by using a pre-trained SAM2 model [14] to perform this process. In simulation, we directly obtain the object shape projection as its contour. And then we adopt the classic 2D-iterative closest point (ICP) method to calculate the motion values M_{obj}, M_{carr}. As described in Sec. III, M includes both the rotation angle θ and translation values $T(x, y)$. Therefore,

$$
\begin{aligned}
M^{cam1}_{obj} &= \{\theta^{obj,cam1}_i, T(x)^{obj,cam1}_i, T(y)^{obj,cam1}_i\}, \\
M^{cam2}_{obj} &= \{\theta^{obj,cam2}_i, T(x)^{obj,cam2}_i, T(y)^{obj,cam2}_i\}, \\
M^{cam1}_{carr} &= \{\theta^{carr,cam1}_i, T(x)^{carr,cam1}_i, T(y)^{carr,cam1}_i\}, \\
M^{cam2}_{carr} &= \{\theta^{carr,cam2}_i, T(x)^{carr,cam2}_i, T(y)^{carr,cam2}_i\}.
\end{aligned}
$$

Consequently, for each time step, we get a motion vector $\vec{M}$ consisting of 15 values as the input to the encoder (we omit the vector arrow in the remainder of the paper for ease of reading). It includes data from two cameras, two contours per camera, three motion values per contour, and three end-effector position values.

For the encoder, we adopt a simple deep neural network such as ResNet-18 and ResNet-50, to extract the feature embedding of the motion vectors $\vec{M}$ and output a continuous risk value $R_t \in [0, 1]$. The training loss is defined by the MSE loss over n samples:

$$
\mathcal{L}_{MSE} = \frac{1}{n} \sum_{t=1}^{n} (R^{pred}_t - R^{gt}_t)^2. \tag{1}
$$

C. Assessment Metrics

We evaluate using four metrics: MSE, precision, recall and execution rate. The MSE value is directly from Eq. 1. We calculate precision and recall by defining 5 cases as described in Tab. I, where we note the trajectories with ground truth label that indicate failure occurred as τ^{gt}_{fail}, and $\tau^{gt}_{no-fail}$ if no

Setting	Encoder Arch	Input	Evaluation	MSE ↓	Precision ↑	Recall ↑	R_{exe} ↑
Sim-only	ResNet-18	Raw image	Sim&Real	0.065	0.448	1.000	0.200
		Motion value	Sim&Real	0.077	0.364	0.981	0.487
		Raw image	Real-only	0.271	0.600	1.000	0.000
		Motion value	Real-only	0.175	0.000	0.000	0.000
	ResNet-50	Raw image	Sim&Real	0.065	0.347	1.000	0.275
		Motion value	Sim&Real	0.080	0.449	0.985	0.368
		Raw image	Real-only	0.279	0.600	1.000	0.000
		Motion value	Real-only	0.235	0.000	0.000	0.000
Sim&Real	ResNet-18	Raw image	Sim&Real	0.063	0.451	0.995	0.218
		Motion value	Sim&Real	0.077	0.351	0.981	0.504
		Raw image	Real-only	0.172	0.500	0.667	0.270
		Motion value	Real-only	0.154	0.000	0.000	0.000
	ResNet-50	Raw image	Sim&Real	0.072	0.466	0.995	0.210
		Motion value	Sim&Real	0.079	0.451	1.000	0.377
		Raw image	Real-only	0.174	0.600	1.000	0.000
		Motion value	Real-only	0.138	0.600	1.000	0.000
Real-only	ResNet-18	Raw image	Sim&Real	0.099	0.546	1.000	0.000
		Motion value	Sim&Real	0.266	0.280	0.318	0.183
		Raw image	Real-only	0.147	0.600	1.000	0.000
		Motion value	Real-only	0.134	0.600	1.000	0.000
	ResNet-50	Raw image	Sim&Real	0.098	0.546	1.000	0.000
		Motion value	Sim&Real	0.437	0.750	0.012	0.375
		Raw image	Real-only	0.147	0.600	1.000	0.000
		Motion value	Real-only	0.046	0.750	1.000	0.375

TABLE II
EXPERIMENT RESULTS BASED ON THREE TYPES OF TRAINING SETS. RAW IMAGE REFERS TO OUR BASELINE METHOD.

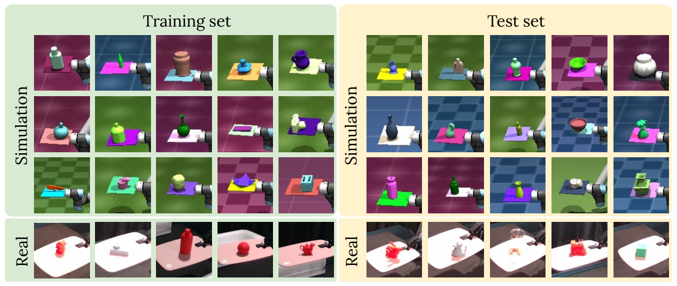

Fig. 5. Visualization of the dataset. The objects used in the training set have no overlap with those in the test set.

failure occurred. Similarly, for the predicted results, we note the trajectory as τ^{pred}_{fail} if failure is detected and $\tau^{pred}_{no-fail}$ if no failure is detected. For τ^{gt}_{fail}, we already define the key moment to identify failure as t_{stop}, while for τ^{pred}_{fail}, we define this moment as t^{pred}_{stop}. Examples of these 5 cases are in Sec. V-B.

In addition, to avoid the model simply predicting a high risk value at the start of manipulation in order to always avoid failure, we quantify this behavior to evaluate overly conservative predictions. We assess the model by introducing the execution rate R_{exe}, which represents the percentage of time steps completed at the moment when the failure is correctly predicted, as seen in Eq. 2:

$$R_{exe} = \frac{t^{pred}_{stop}}{t_{stop}}, \tag{2}$$

where the optimal R_{exe} is 1.0, meaning that the model predicts the failure at the last possible moment to apply an emergency compensation.

V. EXPERIMENTS

We use one UR-5 robot arm for both MuJoCo simulation [18] and real-world experiments. The carrier of the UR-5 consists of a Robotiq parallel-jaw gripper which firmly holds a flat tray, and an object lies on the tray. We install two RealSense cameras, one from a top-down view and the other from a side view. We use two GeForce 4090 GPUs for model training.

We collect 5,349 manipulation trajectories from simulation, and 52 trajectories from the real-world. Each trajectory demonstrates the robot performing a lift-and-place task, where it holds a tray, transports an object placed on top, and moves back and forth within the workspace. The tray is assigned a random tilt angle offset from the horizontal pose, and the joint speeds are perturbed with random acceleration offsets to reach the **maximum joint speed** during manipulation, mimicking an industrial scenario. We have 5,349 objects for simulation and 10 objects for the real-world. The objects for simulation come from Lum et al. [10], which contains 5,751 distinct object meshes. The real-world objects are 3D printed. The collected data is split into training, validation, and test sets, using an 8:1:1 ratio. The objects in the test set are distinct from those in training, see Fig. 5 for some examples.

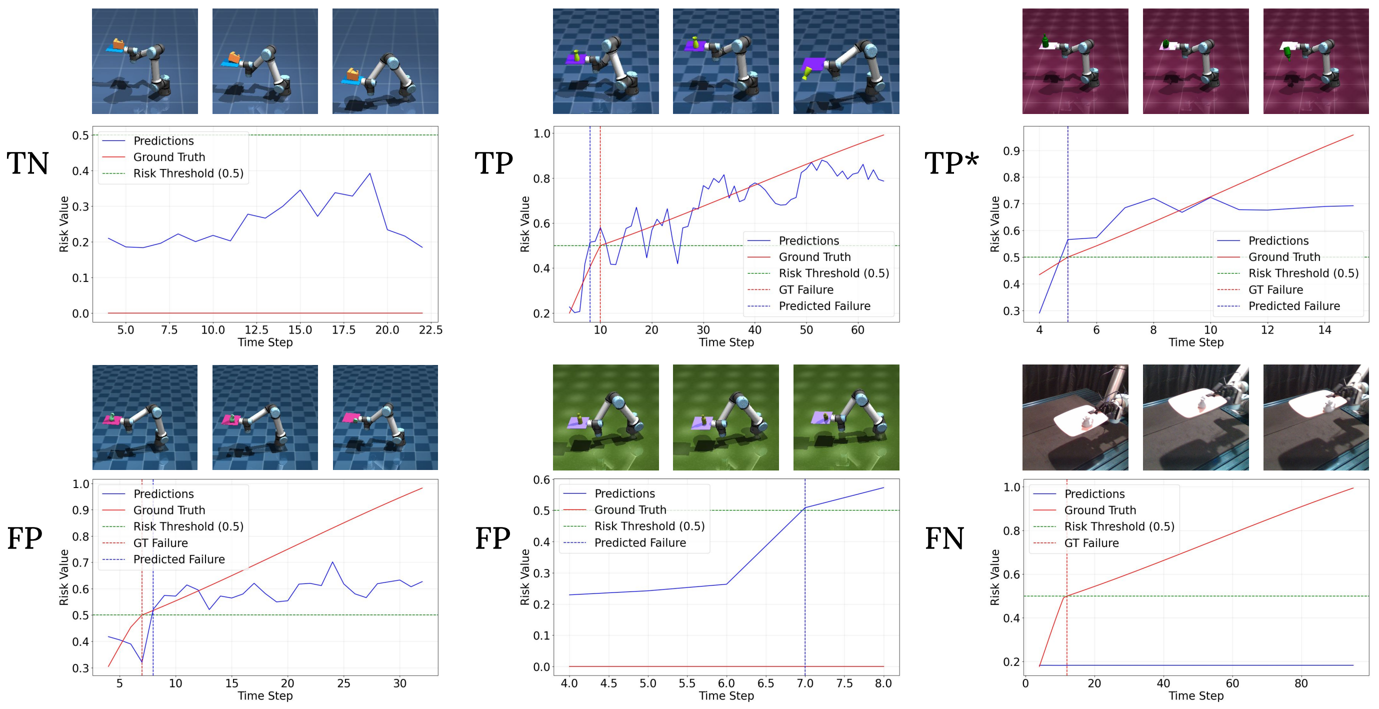

Fig. 6. Visualization of the predicted risk value curve vs the ground truth risk value curve during evaluation. The solid red line refers to the ground truth curve, while the solid blue line demonstrates the predicted curve. The dotted red line refers to the moment of t_{stop}, while the dotted blue line is t_{stop}^{pred}. The threshold of risk value ($R_t = 0.5$) is indicated by dotted green line. TP* is a perfect case that demonstrates our $t_{stop}^{pred} = t_{stop}$.

A. Model Evaluation

To verify the effectiveness of our proposed motion value representation, we implement a vision-based representation method as a baseline. This shares the same feature encoder structure as our proposed method, but takes raw images as input instead of motion values. The encoder directly predicts the regression value of R_t from the input images.

We train the model and report its performance using three training sets: pure simulation data, pure real-world data, and a mix of both. For each training setting, we train the model with different encoder sizes. For each encoder architecture, we train the model with both raw images and motion values. We evaluate each model on two types of test sets: pure real-world data and Sim&Real data.

Our experimental results are listed in Tab. II. As described in Sec. IV-C, the four metrics evaluate from multiple perspectives. MSE evaluates the overall performance of the model in terms of the prediction accuracy at each time step. However, low MSE does not mean the model always correctly predicts failures. Precision and recall values indicate the correctness of proactive failure prediction. While a high precision value demonstrates a successful failure prediction before it actually happens, it is possible that the model is overly conservative. For example, it would be undesirable if a model always predicts failure at the beginning of manipulation.

From the experimental results, we observe that as model size increases, the MSE values tend to increase. For example, the MSE of the ResNet-50 model trained on Sim&Real data with raw image inputs and tested on pure real-world data, increases from 0.172 to 0.174, compared to the model trained with ResNet-18. However, the precision improves from 0.500 to 0.600 and the recall greatly improves from 0.667 to 1.000.

Compared to the baseline method, where the raw image is the only input, while our proposed method tends to have higher MSE and lower precision and recall, the R_{exe} is higher. This means our method is effective in keeping the model from acting overly conservative. This is an important advancement, since a model with low R_{exe} has weak practical significance.

In terms of generalization, there is a visual sim2real gap between simulated and real-world images. Our proposed motion-value-based method is less sensitive to the visual domain gap, as it does not directly rely on raw image features. However, there still exists a sim2real gap when we obtain object contours in the real-world, since we use an open-source segmentation model, while in simulation we can directly use ground-truth geometry projection. When evaluating the Real-only test set, the MSE of the baseline is always larger than our method, and the R_{exe} of the baseline method is lower in most cases. These indicate that our method has stronger visual generalization.

B. Visualization

In Fig. 6, we visualize representative prediction results and simultaneously illustrate the five cases used to calculate precision and recall. These results are based on the setting where the model is trained and evaluated on Sim&Real data, using ResNet-18 with motion values as input. In the TN case, our model correctly predicts low risk values throughout the manipulation. In the FP case, the model either incorrectly predicts a high risk value when no failure occurs, or it predicts t_{stop}^{pred} after the ground-truth t_{stop}. In the TP case, the model correctly predicts t_{stop}^{pred} before the ground-truth t_{stop}. In the FN case, the model fails to predict any high risk values, even though the ground truth indicates a failure. Additionally, the TP* case represents a perfect prediction, where $t_{stop}^{pred} = t_{stop}$ and $R_{exe} = 1.0$.

VI. Conclusion

In this paper, we study proactive failure prediction in the context of lift-and-place tasks. We propose extracting relative motion from images to represent visual changes in the scene and encoding these motion values to proactively predict failures. Our experiments suggest that the proposed method effectively generalizes to unseen objects during test-time deployment. We hope that our proposed method contributes to advancing proactive failure prediction.

References

[1] Christopher Agia, Rohan Sinha, Jingyun Yang, Zi-ang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking Failure Modes of Generative Policies: Runtime Monitoring of Consistency and Progress. In *Conference on Robot Learning (CoRL)*, 2024.

[2] Maximilian Diehl. *Explainable and Interpretable Methods for Handling Robot Task Failures*. PhD thesis, Chalmers Tekniska Hogskola (Sweden), 2025.

[3] Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*, 2024.

[4] KAYA Erdogan. Spline interpolation techniques. *Journal of Technical Science and Technologies*, pages 47–52, 2013.

[5] Ping Wun Huang and Kuan-Jung Chung. Task failure prediction for wafer-handling robotic arms by using various machine learning algorithms. *Measurement and Control*, 54(5-6):701–710, 2021.

[6] Arda Inceoglu, Eren Erdal Aksoy, and Sanem Sariel. Multimodal detection and classification of robot manipulation failures. *IEEE Robotics and Automation Letters*, 9 (2):1396–1403, 2024. doi: 10.1109/LRA.2023.3346270.

[7] Jeon Ho Kang, Neel Dhanaraj, Siddhant Wadaskar, and Satyandra K Gupta. Using large language models to generate and apply contingency handling procedures in collaborative assembly applications. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15585–15592. IEEE, 2024.

[8] Gregory LeMasurier, Alvika Gautam, Zhao Han, Jacob W Crandall, and Holly A Yanco. Reactive or proactive? how robots should explain failures. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, pages 413–422, 2024.

[9] Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. In *Conference on Robot Learning*, pages 3468–3484. PMLR, 2023.

[10] Tyler Ga Wei Lum, Albert H Li, Preston Culbertson, Krishnan Srinivasan, Aaron D Ames, Mac Schwager, and Jeannette Bohg. Get a grip: Multi-finger grasp evaluation at scale enables robust sim-to-real transfer. *arXiv preprint arXiv:2410.23701*, 2024.

[11] Yuliang Ma, Jingyi Liu, Ilshat Mamaev, and Andrey Morozov. Multimodal failure prediction for vision-based manipulation tasks with camera faults. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2951–2957. IEEE, 2024.

[12] Anirudha Majumdar and Marco Pavone. How should a robot assess risk? towards an axiomatic theory of risk in robotics. In *Robotics Research: The 18th International Symposium ISRR*, pages 75–84. Springer, 2020.

[13] Pradip Pramanick and Silvia Rossi. Multimodal coherent explanation generation of robot failures. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2487–2493. IEEE, 2024.

[14] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[15] Som Sagar and Ransalu Senanayake. Robofail: Analyzing failures in robot learning policies. *arXiv preprint arXiv:2412.02818*, 2024.

[16] Rohan Sinha, Amine Elhafsi, Christopher Agia, Matthew Foutter, Edward Schmerling, and Marco Pavone. Real-time anomaly detection and reactive planning with large language models. *arXiv preprint arXiv:2407.08735*, 2024.

[17] Santosh Thoduka, Nico Hochgeschwender, Juergen Gall, and Paul G Plöger. A multimodal handover failure detection dataset and baselines. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 17013–17019. IEEE, 2024.

[18] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[19] Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Tree Miller, Robert Lee, Paarth Shah, Rares Ambrus, Haruki Nishimura, and Masha Itkina. Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies. *CoRR*, 2025.

[20] Enshen Zhou, Qi Su, Cheng Chi, Zhizheng Zhang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and He Wang. Code-as-monitor: Constraint-aware visual programming for reactive and proactive robotic failure detection. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6919–6929, 2025.