Supplementary Material: PREFAIL: Identifying Precursors to Failures in Robotic Lift-and-Place Tasks to Improve Task Execution Performance

Anonymous Authors

I. Data Post-processing

Our data post-processing involves a **hard cut-off**. We remove time steps after t_{fail}. 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 post-failure states from the initial low-risk stage due to similarly negligible relative motion. Furthermore, post-failure data is not informative for proactive failure prediction and is therefore excluded from training.

II. Reliability of 2D ICP

We use 2D ICP to calculate the relative motions M between the object and the carrier based on their contours $\{V_o^i, V_c^i\}_{\text{top, side}}$ in both camera views at each time step i. To verify the correctness and reliability of the calculated rotation and translation values, we visualize the contours and compare the calculated contour at $i+1$ time step with the ground truth, see Fig. 1. The blue contours indicates $\{V_o^{i-1}, V_c^{i-1}\}_{\text{top, side}}$, while the green contours indicates $\{V_o^i, V_c^i\}_{\text{top, side}}$, and the cyan contours are $\{V_o^i, V_c^i\}^*_{\text{top, side}}$ transformed by the calculated 2D ICP (rotation and translation).

While we calculate the relative motion between the current step and the initial step, the motion values are accumulated across all intermediate adjacent steps. The accuracy of 2D ICP depends heavily on the precision of the extracted contours. In simulation, contours are obtained directly from object projections, whereas in the real world they are derived from SAM2-based segmentation. As a result, the relative motion computed from simulated data is naturally more precise than that from real-world data.

III. Past Risk Data Augmentation

In our network design, given a sampling window of size m, the robot state input branch includes the past risk values from the previous m steps. During training, the model uses these ground-truth values for forward prediction, whereas during inference it must rely on the predicted risk values from the previous step. To reduce the discrepancy between training and inference, where ground-truth past risks are unavailable, we inject random noise into the ground-truth risk values during training so the model becomes robust to imperfect previous predictions, similar in spirit to techniques used to mitigate exposure bias in teacher forcing.

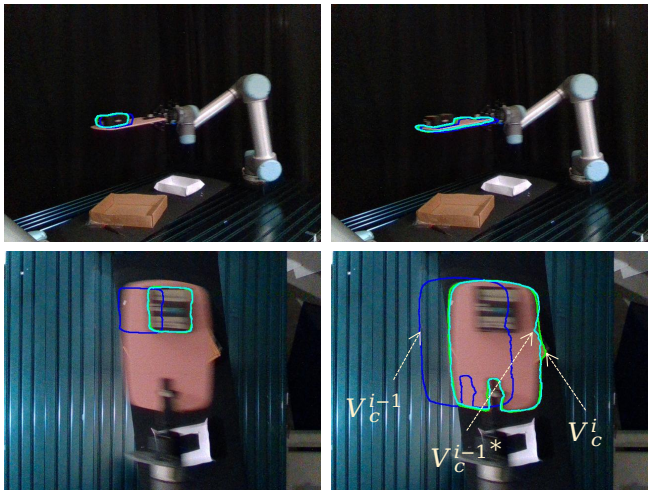

Fig. 1: Reliability of 2D ICP. The contours from the last step are shown in blue, the contours from the current step are shown in green, and the contours obtained by applying rotation and translation relative to the last step are shown in cyan.

IV. Dataset

Our collected dataset is stored in '.npy' files. For training, we split the data into training, validation, and test sets using an 8:1:1 ratio. During this process, we control that failure and non-failure roll-outs are evenly distributed across the three splits.

V. Simulation

In simulation, we use a single UR5 robot with a flat panel attached to the end-effector as the carrier. For each roll-out, we randomize multiple aspects of the scene. For the panel, we randomize its color, surface friction, target tilt angle, rotation speed, and whether the panel reverses direction or holds upon reaching the target angle. For the object, we randomize its color (set as the inverse of the panel color to ensure visual contrast), size, mass, inertia, surface friction, and its initial position on the carrier; the object's initial orientation is sampled from a predefined configuration. For the robot, we randomize the execution trajectory by both shuffling the order of waypoints and applying per-joint angle perturbations, as well as independently randomizing the movement speed and the pause duration between waypoints. For the background, we randomize the ground texture (selected from a pool of materials), texture repeat scale, surface reflectance, and apply small RGB noise to the floor color. Finally, we introduce independent perturbations to the 3D positions of both the top-view and front-view cameras.

Modules	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Hermite (used)	**0.995**	**0.997**	0.995	**0.923**
Bézier	0.973	0.988	0.995	0.887
Linear	0.958	0.980	**1.0000**	0.825
Step	0.314	0.314	**1.0000**	0.1373

TABLE I: Comparison of different interpolation techniques.

VI. Splines

For the ground truth risk value in each failure roll-out. After defining the t_0, t_{stop}, and t_{fail}, we use a Hermite spline [1] to interpolate the risk values for the remaining time steps. We also evaluate alternative interpolation methods, including Bézier interpolation [2], linear interpolation and step interpolation. The comparison results are summarized in Tab. I. With continuous interpolated values such as splines, the overall performance is significantly better than with step interpolation. This is consistent with our discussion in Sec. V-G.

VII. Complete Performance Report for Real-World Cross Validation

Since we perform 10-fold cross-validation for the real-world experiments, we report the results from all folds (Tables II–XI) alongside the average performance across folds (Table XII) to provide a comprehensive evaluation of the model's performance.

Methods / Variants		Encoder Arch	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Fail-Detect [3]	(CFM [4], FM)	ResNet18	0.481	0.481	**1.000**	0.377
	(CFM [4], DP)		0.481	0.481	**1.000**	0.359
	(NatPN [5], FM)		0.481	0.481	**1.000**	0.311
	(NatPN [5], DP)		0.481	0.481	**1.000**	0.311
	(PCA-kmeans [6], FM)		0.481	0.481	**1.000**	0.172
	(PCA-kmeans [6], DP)		0.444	0.444	**1.000**	0.269
	(RND [7], FM)		0.481	0.481	**1.000**	0.329
	(RND [7], DP)		0.481	0.481	**1.000**	0.367
	(logpZO [3], FM)		0.481	0.481	**1.000**	0.231
	(logpZO [3], DP)		0.481	0.481	**1.000**	0.227
PREFAIL (ours)		ResNet18	**0.867**	**0.926**	**1.000**	**0.943**
PREFAIL (ours)		ResNet50	0.800	0.889	**1.000**	0.927
PREFAIL (ours)		ResNet152	0.857	0.926	**1.000**	0.969

TABLE II: Performance on real-world data cross validation split 1.

Methods / Variants		Encoder Arch	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Fail-Detect [3]	(CFM [4], FM)	ResNet18	0.519	0.519	**1.000**	0.270
	(CFM [4], DP)		0.519	0.519	**1.000**	0.205
	(NatPN [5], FM)		0.519	0.519	**1.000**	0.211
	(NatPN [5], DP)		0.519	0.519	**1.000**	0.211
	(PCA-kmeans [6], FM)		0.519	0.519	**1.000**	0.106
	(PCA-kmeans [6], DP)		0.519	0.519	**1.000**	0.110
	(RND [7], FM)		0.519	0.519	**1.000**	0.125
	(RND [7], DP)		0.519	0.519	**1.000**	0.143
	(logpZO [3], FM)		0.519	0.519	**1.000**	0.100
	(logpZO [3], DP)		0.519	0.519	**1.000**	0.100
PREFAIL (ours)		ResNet18	**1.000**	**1.000**	**1.000**	**0.943**
PREFAIL (ours)		ResNet50	**1.000**	**1.000**	**1.000**	0.904
PREFAIL (ours)		ResNet152	0.933	0.963	**1.000**	0.912

TABLE III: Performance on real-world data cross validation split 2.

Methods / Variants		Encoder Arch	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Fail-Detect [3]	(CFM [4], FM)	ResNet18	0.519	0.519	**1.000**	0.184
	(CFM [4], DP)		0.519	0.519	**1.000**	0.346
	(NatPN [5], FM)		0.481	0.481	**1.000**	0.159
	(NatPN [5], DP)		0.481	0.481	**1.000**	0.159
	(PCA-kmeans [6], FM)		0.519	0.519	**1.000**	0.283
	(PCA-kmeans [6], DP)		0.519	0.519	**1.000**	0.097
	(RND [7], FM)		0.519	0.519	**1.000**	0.366
	(RND [7], DP)		0.519	0.519	**1.000**	0.248
	(logpZO [3], FM)		0.519	0.519	**1.000**	0.196
	(logpZO [3], DP)		0.519	0.519	**1.000**	0.241
PREFAIL (ours)		ResNet18	**0.929**	**0.963**	**1.000**	**0.891**
PREFAIL (ours)		ResNet50	**0.929**	**0.963**	**1.000**	0.834
PREFAIL (ours)		ResNet152	0.857	0.926	**1.000**	0.896

TABLE IV: Performance on real-world data cross validation split 3.

Methods / Variants		Encoder Arch	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Fail-Detect [3]	(CFM [4], FM)	ResNet18	0.481	0.481	**1.000**	0.276
	(CFM [4], DP)		0.481	0.481	**1.000**	0.324
	(NatPN [5], FM)		0.519	0.519	**1.000**	0.187
	(NatPN [5], DP)		0.519	0.519	**1.000**	0.187
	(PCA-kmeans [6], FM)		0.519	0.519	**1.000**	0.109
	(PCA-kmeans [6], DP)		0.519	0.519	**1.000**	0.119
	(RND [7], FM)		0.481	0.481	**1.000**	0.225
	(RND [7], DP)		0.519	0.519	**1.000**	0.251
	(logpZO [3], FM)		0.519	0.519	**1.000**	0.131
	(logpZO [3], DP)		0.519	0.519	**1.000**	0.096
PREFAIL (ours)		ResNet18	**0.929**	0.926	0.929	**0.929**
PREFAIL (ours)		ResNet50	0.867	**0.926**	**1.000**	0.908
PREFAIL (ours)		ResNet152	**0.929**	**0.926**	0.929	0.910

TABLE V: Performance on real-world data cross validation split 4.

Methods / Variants		Encoder Arch	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Fail-Detect [3]	(CFM [4], FM)	ResNet18	0.407	0.407	**1.000**	0.367
	(CFM [4], DP)		0.481	0.481	**1.000**	0.411
	(NatPN [5], FM)		0.407	0.407	**1.000**	0.290
	(NatPN [5], DP)		0.407	0.407	**1.000**	0.290
	(PCA-kmeans [6], FM)		0.444	0.444	**1.000**	0.249
	(PCA-kmeans [6], DP)		0.444	0.444	**1.000**	0.312
	(RND [7], FM)		0.519	0.519	**1.000**	0.394
	(RND [7], DP)		0.519	0.519	**1.000**	0.348
	(logpZO [3], FM)		0.519	0.519	**1.000**	0.185
	(logpZO [3], DP)		0.481	0.481	**1.000**	0.332
PREFAIL (ours)		ResNet18	**1.000**	**1.000**	**1.000**	**0.936**
PREFAIL (ours)		ResNet50	0.929	0.963	**1.000**	0.932
PREFAIL (ours)		ResNet152	0.929	0.963	**1.000**	0.967

TABLE VI: Performance on real-world data cross validation split 5.

Methods / Variants		Encoder Arch	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Fail-Detect [3]	(CFM [4], FM)	ResNet18	0.481	0.481	**1.000**	0.337
	(CFM [4], DP)		0.519	0.519	**1.000**	0.296
	(NatPN [5], FM)		0.519	0.519	**1.000**	0.403
	(NatPN [5], DP)		0.519	0.519	**1.000**	0.403
	(PCA-kmeans [6], FM)		0.481	0.481	**1.000**	0.345
	(PCA-kmeans [6], DP)		0.481	0.481	**1.000**	0.413
	(RND [7], FM)		0.519	0.519	**1.000**	0.300
	(RND [7], DP)		0.519	0.519	**1.000**	0.367
	(logpZO [3], FM)		0.519	0.519	**1.000**	0.273
	(logpZO [3], DP)		0.519	0.519	**1.000**	0.274
PREFAIL (ours)		ResNet18	**0.929**	**0.963**	**1.000**	**0.932**
PREFAIL (ours)		ResNet50	0.846	0.889	0.917	**0.938**
PREFAIL (ours)		ResNet152	0.923	0.926	0.923	0.936

TABLE VII: Performance on real-world data cross validation split 6.

Methods / Variants		Encoder Arch	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Fail-Detect [3]	(CFM [4], FM)	ResNet18	0.444	0.444	**1.000**	0.274
	(CFM [4], DP)		0.519	0.519	**1.000**	0.264
	(NatPN [5], FM)		0.519	0.519	**1.000**	0.262
	(NatPN [5], DP)		0.519	0.519	**1.000**	0.262
	(PCA-kmeans [6], FM)		0.444	0.444	**1.000**	0.132
	(PCA-kmeans [6], DP)		0.481	0.481	**1.000**	0.221
	(RND [7], FM)		0.481	0.481	**1.000**	0.303
	(RND [7], DP)		0.481	0.481	**1.000**	0.317
	(logpZO [3], FM)		0.444	0.444	**1.000**	0.252
	(logpZO [3], DP)		0.481	0.481	**1.000**	0.190
PREFAIL (ours)		ResNet18	**1.000**	**1.000**	**1.000**	**0.940**
PREFAIL (ours)		ResNet50	**1.000**	**1.000**	**1.000**	0.828
PREFAIL (ours)		ResNet152	0.933	0.963	**1.000**	0.880

TABLE VIII: Performance on real-world data cross validation split 7.

Methods / Variants		Encoder Arch	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Fail-Detect [3]	(CFM [4], FM)	ResNet18	0.519	0.519	**1.000**	0.291
	(CFM [4], DP)		0.519	0.519	**1.000**	0.274
	(NatPN [5], FM)		0.519	0.519	**1.000**	0.208
	(NatPN [5], DP)		0.519	0.519	**1.000**	0.208
	(PCA-kmeans [6], FM)		0.519	0.519	**1.000**	0.201
	(PCA-kmeans [6], DP)		0.519	0.519	**1.000**	0.184
	(RND [7], FM)		0.519	0.519	**1.000**	0.208
	(RND [7], DP)		0.519	0.519	**1.000**	0.288
	(logpZO [3], FM)		0.519	0.519	**1.000**	0.148
	(logpZO [3], DP)		0.519	0.519	**1.000**	0.182
PREFAIL (ours)		ResNet18	**0.933**	**0.963**	**1.000**	**0.958**
PREFAIL (ours)		ResNet50	**1.000**	**1.000**	**1.000**	0.946
PREFAIL (ours)		ResNet152	0.778	0.852	**1.000**	0.949

TABLE IX: Performance on real-world data cross validation split 8.

Methods / Variants		Encoder Arch	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Fail-Detect [3]	(CFM [4], FM)	ResNet18	0.444	0.444	**1.000**	0.329
	(CFM [4], DP)		0.481	0.481	**1.000**	0.201
	(NatPN [5], FM)		0.519	0.519	**1.000**	0.366
	(NatPN [5], DP)		0.519	0.519	**1.000**	0.366
	(PCA-kmeans [6], FM)		0.481	0.481	**1.000**	0.169
	(PCA-kmeans [6], DP)		0.444	0.444	**1.000**	0.174
	(RND [7], FM)		0.481	0.481	**1.000**	0.420
	(RND [7], DP)		0.519	0.519	**1.000**	0.377
	(logpZO [3], FM)		0.519	0.519	**1.000**	0.061
	(logpZO [3], DP)		0.444	0.444	**1.000**	0.151
PREFAIL (ours)		ResNet18	**0.933**	**0.963**	**1.000**	**0.915**
PREFAIL (ours)		ResNet50	**0.929**	**0.963**	**1.000**	**0.931**
PREFAIL (ours)		ResNet152	**0.933**	**0.963**	**1.000**	0.908

TABLE X: Performance on real-world data cross validation split 9.

Methods / Variants		Encoder Arch	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Fail-Detect [3]	(CFM [4], FM)	ResNet18	0.481	0.481	**1.000**	0.267
	(CFM [4], DP)		0.481	0.481	**1.000**	0.345
	(NatPN [5], FM)		0.444	0.444	**1.000**	0.355
	(NatPN [5], DP)		0.444	0.444	**1.000**	0.355
	(PCA-kmeans [6], FM)		0.519	0.519	**1.000**	0.495
	(PCA-kmeans [6], DP)		0.519	0.519	**1.000**	0.305
	(RND [7], FM)		0.519	0.519	**1.000**	0.218
	(RND [7], DP)		0.481	0.481	**1.000**	0.314
	(logpZO [3], FM)		0.481	0.481	**1.000**	0.370
	(logpZO [3], DP)		0.481	0.481	**1.000**	0.396
PREFAIL (ours)		ResNet18	**0.923**	**0.926**	0.923	**0.937**
PREFAIL (ours)		ResNet50	0.857	0.889	0.923	**0.957**
PREFAIL (ours)		ResNet152	**0.923**	**0.926**	0.923	0.922

TABLE XI: Performance on real-world data cross validation split 10.

Methods / Variants		Encoder Arch	Precision ↑	Accuracy ↑	Recall ↑	η ↑
Fail-Detect [3]	(CFM [4], FM)	ResNet18	0.478	0.478	**1.000**	0.297
	(CFM [4], DP)		0.500	0.500	**1.000**	0.303
	(NatPN [5], FM)		0.493	0.493	**1.000**	0.275
	(NatPN [5], DP)		0.493	0.493	**1.000**	0.275
	(PCA-kmeans [6], FM)		0.493	0.493	**1.000**	0.226
	(PCA-kmeans [6], DP)		0.489	0.489	**1.000**	0.220
	(RND [7], FM)		0.504	0.504	**1.000**	0.289
	(RND [7], DP)		0.507	0.507	**1.000**	0.302
	(logpZO [3], FM)		0.504	0.504	**1.000**	0.195
	(logpZO [3], DP)		0.496	0.496	**1.000**	0.219
PREFAIL (ours)		ResNet18	**0.944**	**0.963**	0.985	**0.932**
PREFAIL (ours)		ResNet50	0.916	0.948	0.984	0.910
PREFAIL (ours)		ResNet152	0.900	0.933	0.977	0.925

TABLE XII: Average performance on real-world data across all cross validation splits.

REFERENCES

[1] W. Gautschi, *Numerical analysis*. Springer Science & Business Media, 2011.

[2] P. Bézier, "Numerical control-mathematics and applications," *Translated by AR Forrest*, 1972.

[3] C. Xu, T. K. Nguyen, E. Dixon, C. Rodriguez, P. Miller, R. Lee, P. Shah, R. A. 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 (RSS)*, 2025.

[4] L. Yang, Z. Zhang, Z. Zhang, X. Liu, M. Xu, W. Zhang, C. Meng, S. Ermon, and B. Cui, "Consistency flow matching: Defining straight flows with velocity consistency," *arXiv preprint arXiv:2407.02398*, 2024.

[5] B. Charpentier, O. Borchert, D. Zügner, S. Geisler, and S. Günnemann, "Natural posterior network: Deep bayesian predictive uncertainty for exponential family distributions," in *International Conference on Learning Representations (ICLR)*, 2022.

[6] H. Liu, Y. Zhang, V. Betala, E. Zhang, J. Liu, C. Ding, and Y. Zhu, "Multi-task interactive robot fleet learning with visual world models," in *Conference on Robot Learning (CoRL)*, 2025.

[7] N. He, S. Li, Z. Li, Y. Liu, and Y. He, "Rediffuser: reliable decision-making using a diffuser with confidence estimation," in *International Conference on Machine Learning (ICML)*, 2024.