*Supplementary Material for WACV 2024 Paper ID 320 (Algorithms Track)*

# IndustReal: A Dataset for Procedure Step Recognition Handling Execution Errors in Egocentric Videos in an Industrial-Like Setting

## 1. Introduction

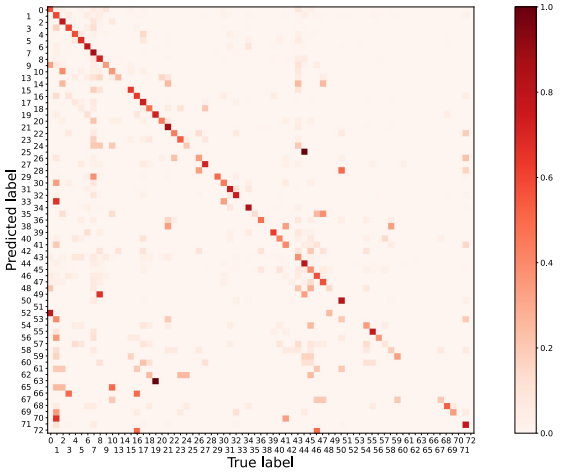
This supplementary material accompanies the main paper titled *IndustReal: A Dataset for Procedure Step Recognition Handling Execution Errors in Egocentric Videos in an Industrial-Like Setting*. This document offers readers a comprehensive and more in-depth understanding of the conducted research.

Section 2 provides additional information on the action recognition (AR), such as the distribution of class instances, implementation details, and further quantitative results. Section 3 provides implementation details on assembly state detection (ASD) and outlines additional qualitative as well as quantitative results. Lastly, Section 4 provides further clarification on the motivation for the proposed procedure order similarity (POS) evaluation metric, implementation details and pseudo code describing the procedure step recognition (PSR) baselines, and a qualitative example.

## 2. Action recognition

### 2.1. Annotations

Annotators were instructed to mark the action start when the participants initiate the action, rather than when participants touch the relevant object(s). Figure 2 demonstrates the action class instances and long-tail distribution for action recognition annotations in the IndustReal dataset. As can be seen, the *check_instruction* action class is the most common, followed by *align_objects*. The long-tail distribution demonstrates that 23 action classes (out of 73) constitute 80% of the dataset. All actions were annotated using the ELAN tool [2]. Two annotators annotated the entire dataset, and each annotator reviewed the annotations of the other.




0: take_short_brace
1: align_objects
2: take_pin_short
3: plug_short_pin
4: take_tooth_washer
5: take_nut
6: tighten_nut
7: check_instruction
8: take_partial_model
9: take_long_brace
10: take_screw_pin
11: take_instruction
12: put_instruction
13: take_pin_long
14: put_pin_long
15: take_wing_beam
16: plug_screw_pin
17: take_round_washer
18: take_acorn_nut
19: tighten_acorn_nut
20: take_pin_middle
21: take_wheel
22: plug_pin_long
23: take_wing
25: plug_partial_model
26: plug_pin_middle
27: take_pulley
28: plug_wheel
29: browse_instruction
30: fit_short_brace
31: fit_tooth_washer
32: fit_round_washer
33: fit_long_brace
34: fit_nut
35: put_screw_pin
36: put_wheel
37: check_wheel
38: pull_wheel
39: loosen_nut
40: put_nut
41: pull_objects
42: put_pin_middle
43: take_objects
44: put_partial_model
45: put_objects
46: pull_pin_short
47: put_pin_short
48: put_long_brace
50: fit_wheel
51: check_partial_model
52: put_short_brace
53: fit_objects
54: put_round_washer
55: fit_pulley
56: fit_wing_beam
57: put_tooth_washer
58: pull_pin_middle
59: put_wing_beam
60: put_pulley
61: pull_screw_pin
62: put_acorn_nut
63: loosen_acorn_nut
64: fit_partial_model
65: take_small_screw_pin
66: plug_small_screw_pin
67: put_small_screw_pin
68: fit_acorn_nut
69: fit_wing
70: pull_pin_long
71: plug_objects
72: pull_small_screw_pin
73: tighten_tooth_washer

Figure 1. Normalized confusion matrix for action recognition (MViTv2 [9] pretrained on Kinetics [7]).

### 2.2. Implementation details

The AR baselines (SlowFast [5] and MViTv2 [9]) are trained using the SlowFast library [5], and some baselines are pre-trained on MECCANO [10]. For all baselines, the configuration yaml files are provided on the dataset repository, such that they can directly be used to reproduce the reported results. The baselines are trained on one Nvidia Tesla v100 GPU.

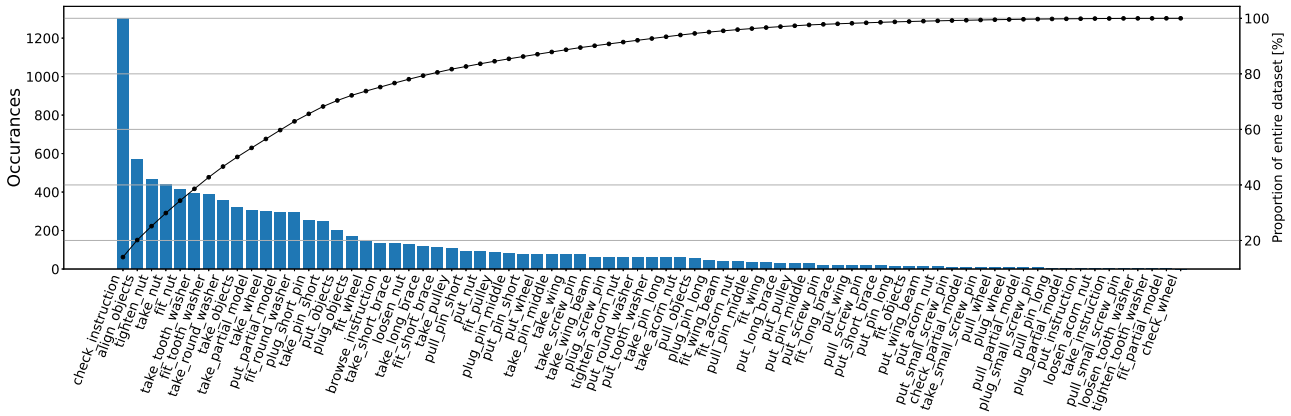



Figure 2. Visualization of the distribution of action recognition labels in the IndustReal dataset.

| Model | Modality | Top-1 acc. [%] | Top-5 acc. [%] |
|---|---|---|---|
| SlowFast [5] | RGB | **60.39** | **85.21** |
| SlowFast [5] | Depth | 43.20 | 73.98 |
| SlowFast [5] | Visible light | 53.75 | 81.48 |
| SlowFast [5] | Stereo images | 57.72 | 83.03 |
| MViTv2 [9] | RGB | **65.25** | **87.93** |
| MViTv2 [9] | Depth | 49.08 | 76.51 |
| MViTv2 [9] | Visible light | 58.59 | 83.50 |
| MViTv2 [9] | Stereo images | 58.86 | 83.55 |

Table 1. AR performance benchmark on IndustReal per modality. All models are pre-trained on Kinetics [7].

### 2.3. Quantitative and qualitative analysis

Figure 1 shows the normalized confusion matrix for action recognition on the IndustReal test set. Specifically, it shows the performance of the MViTv2 [9] transformer, pre-trained on the Kinetics dataset [7]. It is observed that although the performance is generally adequate, the model confuses actions that are visually similar, such as *take* and *put* the *short_brace*, and *tighten* and *loosen* the *acorn_nut*.

Furthermore, the performance of both the MVit and SlowFast architectures on the RGB, depth, visible light, and stereo image modalities are outlined in 1. For both architectures, RGB outperforms the other modalities, likely due to the extensive pre-training on the (RGB) Kinetics [7] dataset. Due to hardware limitations on the current HL2 operating system, it is not possible to access the short-throw depth and RGB camera simultaneously, hence the short-throw depth data are not provided. However, the stereo images can be used to generate high-resolution depth maps, if required.

## 3. Assembly state detection

### 3.1. Implementation details

All ASD baselines make use of the YOLOv8-m [6] backbone. The baselines were trained with a learning rate of 5e-4 using the Adam optimizer, warm-up of 0.5 epoch, patience of 5 epochs, and early stopping enabled. Data augmentation is limited to HSV (with default fractions) and image scaling with a +/- gain of 0.2 for all models. For the model trained solely on synthetic data, random occlusions and image mix-up are used as additional data augmentation techniques. The random occlusions are generated using a single rectangle with random color and size, that covers at least 50% of the bounding box and forces the class label to be background. These occlusions are generated on 33% of the training images. Image mix-up is performed by merging a synthetic training image with a random image taken from VOC2012 [4], weighing the VOC image with a random factor between 0 and 0.2.

All baselines are trained on a single Nvidia Tesla v100.

### 3.2. Quantitative and qualitative analysis

Two samples from the synthetic dataset constructed with Unity Perception [1], used to complement the real-world IndustReal annotations, are demonstrated in Fig. 3. Furthermore, the precision-recall curve of B3 on the test set is shown in Fig. 4. Figure 5 highlights the prediction of the B3 ASD baseline on the IndustReal test set. The figure highlights correct and incorrect predictions for challenging samples.

## 4. Procedure step recognition

### 4.1. Procedure order similarity metric

The main paper proposes to measure the quality of a predicted sequence order $\hat{y}$ (e.g., 'ACB') by comparing it to


(a) Sample one. (b) Sample two.

Figure 3. Samples from the synthetic dataset, generated using the publicly available 3D models of all parts and Unity Perception [1].

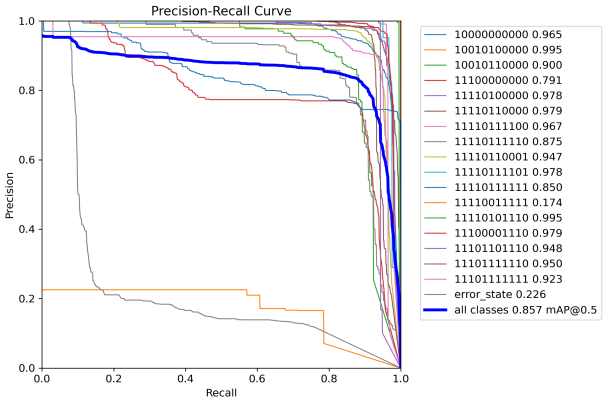



Figure 4. Precision-recall curve on ASD for YOLOv8-m [6], trained on a combination of real and synthetic data, evaluated on the IndustReal test set.

a similarity measure with respect to the ground-truth sequence $y$ (e.g., 'ABC') using string similarity. Note that this is a simplified denotation of $\hat{y}$, as it does not include the prediction time $\hat{t}_{\sigma(i)}$ and confidence $c_{\sigma(i)}$. Approaches on temporal action segmentation use the Levenshtein (Lev) distance [8], normalized over the length of the ground-truth sequence . We propose to make two changes to this metric, which are outlined in more detail here than in the section in the main paper.

The edit distance is defined as the minimum number of edits required to go from a predicted to a ground-truth sequence. For the Lev distance, there are three possible edits to a sequence: insertions, deletions, and substitutions. Insertions and deletions refer to inserting or removing an element of the sequence, e.g. "ABC_" to "ABCD" requires the insertion of a "D". Substitutions consist of replacing an existing sequence prediction for another, e.g. "ABCE" to "ABCD" requires a single substitution of "E" to "D". The first change to the edit distance proposed in [8] is to weight substitutions with a factor of 2, rather than weighting them equally as insertions and deletions, which are weighted by a factor of 1. This essentially eliminates substitutions, as a deletion followed by an insertion is of equal edit distance. The exclusion of substitutions is proposed to prevent the metric from favouring models with many false positives. Models that falsely predict a step when it is not observed, would be penalized less than models that do not predict those false steps.

Table 2. Behaviour of the procedure order similarity (POS) metric for various predictions to the ground truth sequence $y$ "ABCD", compared to using the Lev-based similarity proposed in [8]. wDamLev indicates the weighted DamLev as proposed in Section 4.1, prediction errors are are underlined.

| $\hat{y}$ | ABDC | ADCB | DBCA | _BCD |
|---|---|---|---|---|
| Edits (Lev) | 2 | 2 | 2 | 1 |
| Edits (DamLev) [3] | 1 | 2 | 2 | 1 |
| Edits (wDamLev) | 1 | 3 | 4 | 1 |
| Edit (Lev) [8] | 0.50 | 0.50 | 0.50 | 0.75 |
| POS (DamLev) | 0.75 | 0.50 | 0.50 | 0.75 |
| POS (wDamLev) | 0.75 | 0.25 | 0.00 | 0.75 |

The second proposed modification is to use the Damerau-Levenshtein (DamLev) [3] edit distance, rather than the Lev distance. The DamLev edit distance allows a fourth edit method, namely transpositions, which are swaps between two subsequent elements in the sequence, *e.g.* "ABDC" to "ABCD". Transpositions are included in our proposed procedure order similarity, as it is intuitive to penalize "ACB" less compared to "CAB". Transpositions, like insertions and deletions, are weighed with a factor of 1.

Similarly to Lea *et al.* [8], it is proposed to normalize the edit distance. The aforementioned work normalizes with respect to the length of either the ground truth or the prediction, depending on which is longer. We propose to always normalize with respect to the length of the ground truth, as this prevents models with many false positives from being normalized favourably. Combined with the elimination of substitutions, the proposed normalization no longer bounds the normalized distance between 0 and 1. For instance, a ground truth of (1, 2, 3) and a prediction of (4, 5, 6) would result in a normalized edit distance of 2 (3 deletions followed by 3 insertions). Therefore, we propose to clip all normalized edit distances which are larger than 1. Since no prediction at all would yield a normalized edit distance of unity, distances larger than that are considered particularly poor predictions. Consequently, the difference between bad and worse predictions is uninteresting and not exploited in this metric. Finally, the clipped, normalized edit distance is subtracted from the unity value. This results in a similarity metric, rather than a distance metric. Thus, the procedure order similarity (POS) between $y$ and $\hat{y}$ is defined by Equation 4 in the main paper. A POS value of unity signifies a perfect match between the prediction and the ground truth ($y = \hat{y}$). The behaviour of the POS metric, as well as the


(a) Prediction of the correct class, despite motion blur and a part occluded by the assembly.

(b) Correct prediction (with low confidence) for final state, despite a clearly misaligned front wheel assembly.

(c) Incorrect prediction (with high confidence) for the completed model, while the washer and nut for the front wheel assy are not yet completed.

(d) Incorrect prediction (with high confidence) for completed model, as the rear assy is incorrectly assembled due to a missing rear pulley.

Figure 5. Assembly state detection predictions on the IndustReal test set, using YOLOv8-m [6] (B3 in main paper).

Table 3. Demonstration of the behaviour of PSR metrics on some example predictions. A perfect POS, $F_1$ score, or delay individually does not give sufficient information regarding overall performance. The combination of the metrics provides the best insight into model quality.

| | Action \| observation time | POS | $F_1$ score | Delay $\tau$ |
|---|---|---|---|---|
| Ground truth | $a_0$ \| 5 s, $a_1$ \| 10 s, $a_2$ \| 15 s, $a_3$ \| 20 s | – | – | – |
| Prediction 1 | $a_0$ \| 5 s, $a_1$ \| 10 s, $a_2$ \| 15 s, $a_3$ \| 20 s | 1.00 | 1.00 | 0.0 s |
| Prediction 2 | $a_0$ \| 5 s, $a_1$ \| 10 s, $a_3$ \| 20 s, $a_2$ \| 25 s | 0.75 | 1.00 | 2.5 s |
| Prediction 3 | $a_0$ \| 5 s, $a_1$ \| 10 s, $a_3$ \| 20 s | 0.75 | 0.86 | 5.0 s |
| Prediction 4 | $a_3$ \| 20 s, $a_2$ \| 25 s, $a_1$ \| 30 s, $a_0$ \| 35 s | 0.00 | 1.00 | 5.0 s |
| Prediction 5 | $a_0$ \| 5 s, $a_1$ \| 5 s, $a_2$ \| 10 s, $a_3$ \| 15 s | 1.00 | 0.40 | 0.0 s |

DamLev edit distance, is illustrated in Tab. 2.

As mentioned in the main paper, looking exclusively at the POS metric is not sufficient for two reasons. Firstly, it does not penalize false positive predictions, if the action is later indeed correctly completed. Secondly, POS does not take time delay into account. For instance, a model could guess the order correctly before any step is completed, and obtain a perfect POS score. For these reasons, two complementary metrics are selected: $F_1$ score and the average delay $\tau$. Table 3 demonstrates the behaviour of all three metrics and outlines why the combination of them is essential.

## 4.2. Implementation details

The three baselines towards PSR that are evaluated in the paper, are further outlined in this section. Specifically, Algorithm 1 describes B1, Algorithm 2 describes B2, and Algorithm 3 describes B3. B1 consists of the most straightforward approach, where each detected assembly state that differs from the previously observed state, triggers the completion of all steps required to arrive at the detected state. B2 accumulates the prediction confidences over time, therefore filtering the ASD predictions, resulting in improved POS and $F_1$ score at the cost of an increased delay $\tau$. B3 accumulates the confidences, like B2, and additionally restricts the state transitions to the ones that are expected in the procedure. Only the expected transitions are used to predict completed procedure steps.

**input :** List of ASD predictions $\hat{ASD}$ (in video $X_i$)
**output:** PSR predictions $\hat{y}$
$\hat{y} \leftarrow$ empty list;
$ASD_{curr} \leftarrow \hat{ASD}[0]$;
$T \leftarrow 0.5$; `/* Conf. threshold */`
**for** $f \in \{0 \dots \text{len}(\hat{ASD})\}$ {
  $state, conf \leftarrow getHighestPred(\hat{ASD}[f])$;
  **if** $conf \geq T$ **then**
    **if** $ASD_{curr} \neq state$ **then**
      append $state$ to $\hat{y}$;
      $ASD_{curr} \leftarrow state$;
    **end**
  **end**
}

**Algorithm 1:** PSR baseline 1

## 4.3. Qualitative analysis

Figure 6 outlines the ASD predictions on a single video in the IndustReal test set for the best scoring ASD approach, as well as the approach trained exclusively on synthetic data. These predictions are used by the PSR baselines and can therefore be used to qualitatively analyze the results from Table 4 in the main paper. It is observed that neither model is able to detect the error state around frame 1300, resulting in false positives for both PSR baselines. The model trained only on synthetic data wrongly classifies the second

**input :** List of ASD predictions $\hat{ASD}$ (in video $X_i$)
**output:** PSR predictions $\hat{y}$
$\hat{y} \leftarrow$ empty list;
$ASD_{curr} \leftarrow \hat{ASD}[0]$;
$confs \leftarrow$ array with zeros for each component;
$T \leftarrow 8.0$ ; /* Conf. threshold */
$decay \leftarrow 0.75$ ; /* Confidence decay */
**for** $f \in \{0 \dots \text{len}(\hat{ASD})\}$ {
  $state, conf \leftarrow getHighestPred(\hat{ASD}[f])$;
  **for** $i \in \{0 \dots \text{len}(state)\}$ {
    **if** $ASD_{curr,i} \neq state_i$ **then**
      $confs_i \leftarrow confs_i + conf$;
      **if** $confs_i \geq T$ **then**
        append $state_i$ to $\hat{y}$;
        $ASD_{curr,i} \leftarrow state_i$;
      **end**
    **else**
      $confs_i = confs_i \cdot decay$;
    **end**
  }
}

**Algorithm 2:** PSR baseline 2

**input :** List of ASD predictions $\hat{ASD}$ (in video $X_i$), descriptive set of procedure information $\mathcal{P}$
**output:** PSR predictions $\hat{y}$
$\hat{y} \leftarrow$ empty list;
$ASD_{curr} \leftarrow \text{initialState}(\mathcal{P})$;
$confs \leftarrow$ array with zeros for each component;
$s_{exp} \leftarrow findExpectedStates(\mathcal{P})$;
$T \leftarrow 8.0$ ; /* Conf. threshold */
$decay \leftarrow 0.75$ ; /* Confidence decay */
**for** $f \in \{0 \dots \text{len}(\hat{ASD})\}$ {
  $state, conf \leftarrow getHighestPred(\hat{ASD}[f])$;
  **for** $i \in \{0 \dots \text{len}(state)\}$ {
    **if** $ASD_{curr,i} \neq state_i$ **then**
      $confs_i \leftarrow confs_i + conf$;
      **if** $confs_i \geq T$ & $ASD_{curr,i} \in s_{exp}$ **then**
        append $state_i$ to $\hat{y}$;
        $ASD_{curr,i} \leftarrow state_i$;
      **end**
    **else**
      $confs_i = confs_i \cdot decay$;
    **end**
  }
}

**Algorithm 3:** PSR baseline 3

and last state in the video, explaining the difference in performance between B3 and B3-S outlined in Table 4 in the main paper.

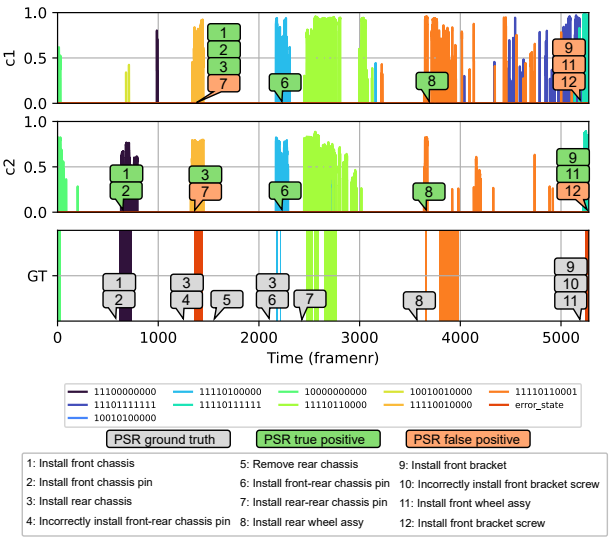



Figure 6. PSR predictions by B3-S (c1) and B3 (c2), together with the accompanied ASD classifications by YOLOv8-m [6], trained exclusively on synthetic data and on a combination of real and synthetic data, respectively. Predictions are shown for a single video in the IndustReal test set. False and true positive PSR predictions are highlighted.

## References


[1] Steve Borkman, Adam Crespi, Saurav Dhakad, Sujoy Ganguly, Jonathan Hogins, You-Cyuan Jhang, Mohsen Kamalzadeh, Bowen Li, Steven Leal, Pete Parisi, et al. Unity perception: Generate synthetic data for computer vision. *arXiv preprint arXiv:2107.04259*, 2021.

[2] Hennie Brugman, Albert Russel, and Xd Nijmegen. Annotating multi-media/multi-modal resources with elan. In *LREC*, pages 2065–2068, 2004.

[3] Fred J Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, Mar. 1964.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[5] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In *Proc. ICCV*, pages 6202–6211, Los Alamitos, 2019. IEEE.

[6] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, Jan. 2023.

[7] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[8] Colin Lea, Austin Reiter, René Vidal, and Gregory D Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. In *European conference on computer vision*, pages 36–52. Springer, 2016.

[9] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. MViTv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4804–4814, 2022.

[10] Francesco Ragusa, Antonino Furnari, Salvatore Livatino, and Giovanni Maria Farinella. The MECCANO dataset: Understanding human-object interactions from egocentric videos in an industrial-like domain. In *Proc. WACV*, pages 1569–1578, Los Alamitos, 2021. IEEE.