Learning to Recognize Correctly Completed Procedure Steps in Egocentric Assembly Videos through Spatio-Temporal Modeling*

Tim J. Schoonbeek[a,1,*], Shao-Hsuan Hung[a,*], Dan Lehman[a], Hans Onvlee[b], Jacek Kustra[b], Peter H.N. de With[a], Fons van der Sommen[a]

[a]*Dept. of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, Netherlands*
[b]*ASML, De Run 6501, 5504 DR, Veldhoven, Netherlands*

Abstract

Procedure step recognition (PSR) aims to identify all correctly completed steps and their sequential order in videos of procedural tasks. The existing state-of-the-art models rely solely on detecting assembly object states in individual video frames. By neglecting temporal features, model robustness and accuracy are limited, especially when objects are partially occluded. To overcome these limitations, we propose Spatio-Temporal Occlusion-Resilient Modeling for Procedure Step Recognition (STORM-PSR), a dual-stream framework for PSR that leverages both spatial and temporal features. The assembly state detection stream operates effectively with unobstructed views of the object, while the spatio-temporal stream captures both spatial and temporal features to recognize step completions even under partial occlusion. This stream includes a spatial encoder, pre-trained using a novel weakly supervised approach to capture meaningful spatial representations, and a transformer-based temporal encoder that learns how these spatial features relate over time. STORM-PSR is evaluated on the MECCANO and IndustReal datasets, reducing the average delay between actual and predicted assembly step completions by 11.2% and 26.1%, respectively, compared to prior methods. We demonstrate that this reduction in delay is driven by the spatio-temporal stream, which does not rely on unobstructed views of the object to infer completed steps. The code for STORM-PSR, along with the newly annotated MECCANO labels, is made publicly available at `https://timschoonbeek.github.io/stormpsr`.

Keywords: Computer Vision in Industrial Contexts, Egocentric Vision in Assistive Contexts, Video Understanding, Procedure Step Recognition, Representation Learning

1. Introduction

In the field of assistive computer vision in industrial contexts, understanding and tracking key steps in videos of procedure execution is an active line of research [44, 21, 11, 31, 59, 34, 1]. Procedural tasks often require multiple interdependent actions to be executed correctly, although

*Author accepted manuscript (AAM). Accepted for publication in *Computer Vision and Image Understanding*. The Version of Record will be available in due course; this arXiv record will be updated with the DOI once assigned. Please cite the published article when available.

*equal contribution

[1]Corresponding author: Tim J. Schoonbeek, email: t.j.schoonbeek@tue.nl

reliably recognizing these steps remains challenging due to variable viewing angles, frequent occlusions, and flexible execution orders. Particularly egocentric cameras introduce dynamic perspectives and frequent hand occlusions, obscuring key objects or actions and creating gaps in visual continuity that complicate accurate procedural tracking [36, 43, 20]. Such egocentric technologies can be beneficial for assistive technology in industrial scenarios [37, 29, 30].

Existing work on recognizing and tracking procedural videos take multiple approaches, such as action recognition [21, 44, 36, 11], procedural activity understanding [31, 59, 34], keystep recognition [1, 45, 21, 4], procedural error identification [18], and procedure step recognition [43]. In procedural action recognition, the objective is to categorize actions performed by humans in short video clips. In contrast, the task of procedural *activity* recognition focuses on understanding long-form videos. Within this, the task of *keystep* recognition aims to find key moments and their sequences in long videos of procedure executions. Crucially, the aforementioned approaches lack a measure of completeness and correctness for the recognized actions and steps. Schoonbeek *et al.* [43] address this limitation by proposing a *procedure step recognition* (PSR) task, aiming to identify all *correctly completed* procedural steps in long-form videos, each contributing to the successful execution of a procedure. However, their approach is fundamentally limited by its reliance on assembly state detections (ASD) [19, 32, 51, 41, 50], which only provide reliable predictions for unobstructed views of the object. Such unobstructed video frames are rare, particularly in egocentric videos, where the presence of hands and tools frequently occlude the object of interest due to the nature of hand-object interaction during task execution. Moreover, ASD is based exclusively on spatial information, operating on single frames. Contrarily, PSR is a temporal video understanding task, requiring reasoning over video sequences to assess which steps have been successfully completed. Yet, none of the existing PSR methods have explicitly leveraged temporal features for this purpose. Additionally, the existing approaches are only evaluated on a single dataset, namely the IndustReal [43] dataset. Therefore, the generalization to other data remains unknown. This work explores the hypothesis that procedural step recognition does not require full visibility of the object, but only of the components pertinent to the current step completion. The aim is to determine whether a spatio-temporal model, trained directly on PSR labels, can address the constraints imposed by ASD-based methods. Specifically, STORM-PSR, outlined in Fig. 1, is introduced. STORM-PSR is a two-stream approach for PSR that combines ASD with a spatio-temporal model to recognize procedure steps even under partial occlusions.

The ASD stream detects the assembly stage in each video frame and provides a measure of correctness, since erroneous states can be defined and detected. The PSR task can be addressed through ASD, by inferring which steps are correctly executed between observations of assembly states using prior knowledge of the assembly procedure [43]. Whenever unobstructed images are available, the robust ASD algorithm provides reliable predictions. However, such unobstructed frames are rare in egocentric videos due to hand occlusions and camera motion, resulting in significant delays between the actual completion of a step and its subsequent recognition. Therefore, we propose a second stream, comprising a transformer-based temporal encoder to aggregate spatial features even when only obstructed images are available. This so-called spatio-temporal stream directly predicts correctly completed procedure steps, rather than inferring them between consecutive observations of assembly states. It therefore only requires visibility of the parts of the object that constitute a specific step. For example, to recognize the completion of the step 'install front wheel', the visibility of irrelevant parts, *e.g.*the rear wheel, is not required.

These direct PSR predictions are facilitated through a multi-layer perceptron (MLP) head following the temporal transformer, performing multi-label classification to handle multiple step completions within a single video clip. We introduce key-frame sampling (KFS), a weakly super-

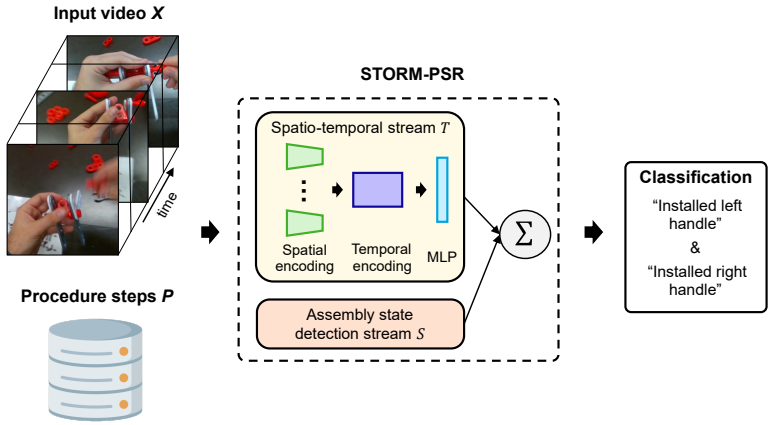


Figure 1: Diagram of the proposed Spatio-Temporal Occlusion-Resilient Modeling for Procedure Step Recognition (STORM-PSR) approach, using assembly state detection for non-occluded frames and temporal modeling to recognize step completions when the assembly object is only partially visible.

vised pre-training approach that leverages sparse step-completion annotations. KFS enables the spatial encoder to provide meaningful representations of assembly states, using a contrastive loss that maximizes agreement between the same states from different videos and occlusion levels, while minimizing the agreement with different states. When available, synthetic data of assembly states can be readily integrated with KFS, thereby reducing reliance on annotated real-world data. For training the temporal transformer, we propose key-clip aware sampling (KCAS). KCAS is based on a bimodal distribution to over-sample step completions (positives) and video frames that are shortly preceding these step completions (hard negatives). Simultaneously, KCAS undersamples potentially ambiguous frames surrounding the exact moment of completion and less informative clips that contain no step completions. Trained with KFS and KCAS, STORM-PSR reduces the average delay (τ) between the completion and corresponding recognition of a step by 26.1% on the IndustReal dataset [43] and 11.2% on the MECCANO dataset [36], which we have annotated with PSR and ASD labels to establish the performance benchmark for this task. In summary, this paper contains the following contributions.

- STORM-PSR: a dual-stream approach that leverages temporal features for PSR, thereby significantly reducing the average delay compared to state-of-the-art methods.
- Key-frame sampling (KFS) for weakly supervised pre-training of the spatial encoder, and key-clip aware sampling (KCAS), a novel sampling strategy for training the temporal encoder.
- Annotations for the MECCANO dataset [36] and a corresponding performance benchmark for PSR and ASD.

The remainder of this article is structured as follows. Section 2 reviews prior work relevant to procedure step recognition, after which Section 3.0 introduces the formal definition of the task and the evaluation metrics used. The proposed framework, STORM-PSR, is described in detail in Section 4. Section 5 reports experimental results on the IndustReal and MECCANO datasets. Section 6 examines the implications of the findings and outlines key limitations. Finally, Section 7 summarizes the main contributions.

2. Related work

2.1. Procedure understanding

Related work explores various approaches to understanding procedural actions, such as action recognition (AR), which focuses on classifying short video clips of procedure executions [44, 36, 11, 5, 57]. Temporal action segmentation (TAS) aims to segment actions, given a long video of procedures [15, 2, 8, 38]. Other works aim to extract key moments and the sequence in which they occur in long-form videos [31, 59, 34, 1, 45, 21, 4] or aim to identify mistakes in procedural egocentric videos [18]. To this end, several approaches utilize narrations [59], textual databases [31], and knowledge graphs [34] to enhance the spatio-temporal features with procedural knowledge. Furthermore, several works explore various attention mechanisms for temporal modeling [46, 52, 58, 47]. Each of the aforementioned tasks is relevant and of interest for procedure understanding, since it is particularly challenging to extract the spatio-temporal features required for acceptable performance. However, these tasks have a limited applicability for use cases with industrial procedures, where a measure of completion and correctness is desirable or even required. For example, while approaches to identify procedural mistakes (*e.g.* [18]) can highlight potentially erroneous actions, it cannot guarantee the *correct completion* of procedural steps: the absence of a correct step completion does not necessitate the presence of a procedural error, and thus can go unnoticed through such approaches. Similarly, action recognition and segmentation approaches provide only a classification of *which* procedure step is being executed, without the notion of whether it has been done *completely* and *correctly*.

While the aforementioned approaches emphasize human execution of procedural actions, assembly state detection (ASD) focuses on the transformation and progression of an object within a procedure, by detecting subsequent assembly states [19, 32, 51, 41, 50]. The ASD task typically involves training an object detection network with bounding boxes and corresponding state labels, and performs well on datasets where images contain unobstructed views of assembly states. While ASD can provide a measure of correctness for assembly states [42], ASD approaches do not leverage temporal consistency on (even briefly) partially occluded objects. Therefore, in egocentric videos, where assembly states are often partially occluded or out of frame, ASD models are unable to recognize object states.

Such occlusions are evident in procedural datasets such as Assembly101 [44], MECCANO [36], Ikea ASM [5], HA4M [10], and IndustReal [43]. Although each dataset is similar in the task of assembling a relatively small object in industrial-like settings, there are notable differences between the datasets. To mitigate the impact of occlusions, Assembly101 [44] and Ikea ASM [5] provide multiple viewpoints, while operators in IndustReal [43] were instructed to occasionally provide an unobstructed view of the object. For MECCANO [36], no such instructions were provided, resulting in a highly challenging dataset with only sparse non-occluded images of the toy motorcycle assembly. Finally, only IndustReal provides CAD data to generate synthetic images of assembly states to supplement the real-world dataset.

2.2. Representation learning with procedural data

Videos of procedure executions exhibit continuous movements, where objects change gradually, resulting in a high similarity among consecutive frames and larger variations between more distant frames. Consequently, representation learning has emerged as a valuable tool for understanding procedural video data, since it inherently learns frame embeddings that trace the procedure progression through latent space. Bansal *et al.* [4] propose a self-supervised *Correspond and Cut* framework to learn temporally corresponding features by aligning key steps in the

embedding space across different videos of the same procedure. Schoonbeek *et al.* [42] propose a representation learning framework for assembly state recognition that enhances generalization to unseen states, demonstrating that supervised contrastive learning generates meaningful embedding spaces for procedural data. For skeleton-based action recognition in videos, contrastive learning has been used to learn multi-granularity anchor-contrastive representations in a semi-supervised manner [48]. Finally, Vidalmata *et al.* [54] propose a two-stage spatio-temporal approach for temporal action segmentation in instructional videos, employing a U-Net [39] to extract spatial features and an MLP for temporal embedding. Inspired by these works, we propose a weakly supervised representation learning method to pre-train the spatial encoder within a spatio-temporal network.

3. Procedure step recognition

This section formalizes the procedure step recognition (PSR) task as defined in [43], outlines the evaluation metrics used to assess system performance, and reviews current limitations of state-of-the-art approaches based on assembly state detection.

3.1. Task definition

The PSR task aims to detect procedure steps that are correctly completed by a person up to time t, given a sensory input $X_t = (x_1, x_2, \cdots x_t)$ (*e.g.*, video) and $\mathcal{P} = \{p_0, \cdots p_N\}$, a descriptive set of the procedural actions to be performed [43]. The predicted set of completed procedure steps $\hat{Y}_t$ consists of both the recognized steps and their respective completion times up to time t, formulated as

$$\hat{Y}_t = \{(\hat{a}_{\sigma(0)}, \hat{t}_{\sigma(0)}), \cdots (\hat{a}_{\sigma(m)}, \hat{t}_{\sigma(m)})\}, \tag{1}$$

where $\hat{a}_{\sigma(i)}$ represents the predicted completion of action $a_i \in \mathcal{P}$ at time $\hat{t}_{\sigma(i)}$, and $m + 1$ denotes the total number of recognized procedure steps. The ground truth procedure steps completed up to time t are defined as

$$Y_t = \{(a_{\rho(0)}, t_{\rho(0)}), \cdots (a_{\rho(k)}, t_{\rho(k)})\}, \tag{2}$$

where $a_{\rho(i)}$ is the predicted completion of the action $a_i \in \mathcal{P}$ at time $t_{\rho(i)}$, and $k + 1$ denotes the total number of correctly executed steps.

3.2. Evaluation metrics

To quantify the performance of the PSR task, three evaluation metrics are used [43]. First, the procedure order similarity (POS) measures how closely the predicted step sequence aligns with the correct order using the edit distance [13], and is defined as

$$\text{POS} = 1 - \min\left(\frac{\text{DamLev}(Y, \hat{Y})}{|Y|}, 1\right), \tag{3}$$

where the DamLev$(\cdot)$ is a weighted DamLev edit distance [12] function. Importantly, POS does not evaluate conformity to a canonical or "correct" step order. Instead, it measures how accurately the system recovers the order in which the user actually executed the procedure. This allows the POS metric to remain relevant even when steps are swapped or repeated, penalizing the model only if steps are recognized in an order different than the one executed by the user.

The second metric, F_1 score, calculates the harmonic mean of recall and precision for all procedure steps. A true positive (TP) is defined as the prediction of a procedure step $(\hat{a}_{\sigma(i)}, \hat{t}_{\sigma(i)})$ observed at or after the actual completion of the ground truth action $(a_{\rho(j)}, t_{\rho(j)})$, defined as

$$(\hat{t}_{\sigma(j)} \geq t_{\rho(i)}) \wedge (a_i \in Y). \tag{4}$$

A false positive (FP) is defined as a procedure step $\hat{a}_{\sigma(i)}$ that is detected prior to the actual completion of action $a_{\rho(j)}$, or if a_i is not completed, defined as

$$(\hat{t}_{\sigma(j)} < t_{\rho(i)}) \vee (a_i \notin Y). \tag{5}$$

A false negative (FN) is defined as correctly completed procedure step $a_{\rho(j)}$ that is never recognized by the model, thus is defined as

$$(a_i \in Y) \wedge (a_i \notin \hat{Y}). \tag{6}$$

Finally, the average delay (τ) measured in seconds, quantifies the average time between the completion and corresponding recognition of a step and is defined as

$$\tau = \frac{1}{n} \sum_{i=0}^{n-1} (\hat{t}_{\sigma(i)} - t_{\rho(i)}), \tag{7}$$

where n is the number of TPs in the prediction $\hat{Y}$.

Interpreting any of the three metrics in isolation can lead to a misleading assessment of system performance. For example, a model that predicts all steps as completed only at the final frame of a video might achieve a high F_1 score while exhibiting a significantly large average delay (τ) and POS scores. Therefore, a comprehensive evaluation requires analyzing the three metrics together to obtain an accurate understanding of the model's overall performance.

3.3. State-of-the-art and limitations

The state-of-the-art approach to PSR relies exclusively on assembly state detections, indirectly achieving step recognition by inferring steps between consecutive assembly states using prior knowledge of the procedure [43]. However, ASD can only accurately recognize object states that are minimally occluded and entirely within the image boundaries, since ASD algorithms predict the state of the whole assembly object requiring a comprehensive view. Consequently, with a single image offering only partial visibility, predictions would require assumptions about the state of non-visible parts. This limitation causes significant delays between the completion and consequent recognition of steps, as unobstructed views of the object are rare, especially in egocentric videos. Therefore, the proposed approach complements ASD predictions with a novel temporal network, optimized directly to recognize the completion of procedure steps, even under partial occlusions.

4. Method

4.1. Overview

This section presents the proposed approach to PSR, using the Spatio-Temporal Occlusion-Resilient Modeling for Procedure Step Recognition (STORM-PSR) framework. STORM-PSR

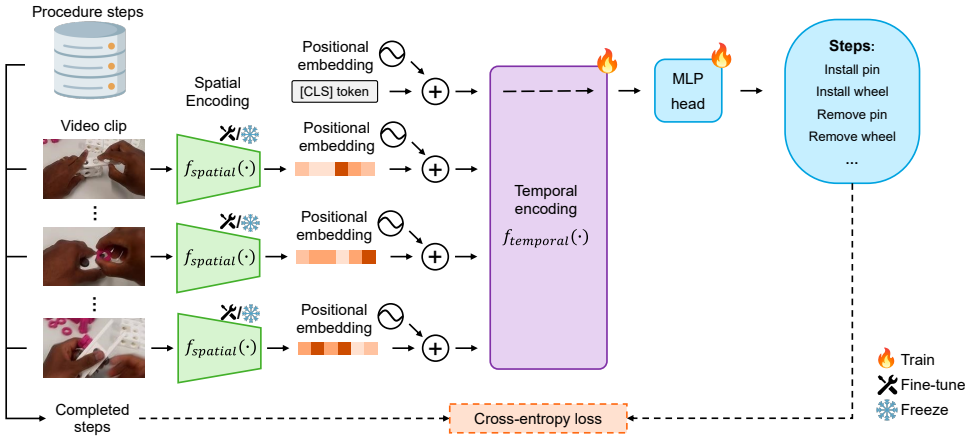


Figure 2: Overview of the spatio-temporal stream of STORM-PSR and its training procedure. The frames of each video clip are passed through the spatial encoder $f_{\text{spatial}}(\cdot)$ to obtain per-frame embeddings. These embeddings are combined with a learnable [CLS] token, and augmented with 1-D positional encoding before being fed into the temporal encoder $f_{\text{temporal}}(\cdot)$. The final state of the [CLS] token provides a clip-level representation, which is fed into the MLP classification head that makes the final predictions. Because multiple procedural steps can be completed within a single clip, the MLP performs multi-label classification. During the training of the temporal encoder, the weights of $f_{\text{spatial}}(\cdot)$ can be either frozen or fine-tuned. The cross-entropy loss is used to optimize the model weights.

combines an assembly state detection (ASD) stream ($\mathcal{S}$) with a spatio-temporal stream ($\mathcal{T}$), as outlined in Fig. 1.

Specifically, given an input video X and a set of procedural actions $\mathcal{P}$, the objective of the two-stream method, STORM-PSR ($\mathcal{F}$), is to map X and $\mathcal{P}$ to $\hat{Y}$, an ordered set of tuples containing each correctly completed procedure step $p \in \mathcal{P}$ and its moment of completion $t \in [0, T]$, where T is the total duration of video X, such that

$$\hat{Y} = \mathcal{F}(X, \mathcal{P}). \tag{8}$$

Each stream produces a probability distribution over every procedure step $k \in \mathcal{P}$, denoted by $\hat{y}_{\mathcal{S},k} = \mathcal{S}(X, \mathcal{P})$ and $\hat{y}_{\mathcal{T},k} = \mathcal{T}(X, \mathcal{P})$ for the assembly state and spatio-temporal streams, respectively. The prediction probability $\hat{y}_k$ for the correct completion of step k is then computed as the average with equal contribution from both streams,

$$\hat{y}_k = 0.5 \cdot \hat{y}_{\mathcal{S},k} + 0.5 \cdot \hat{y}_{\mathcal{T},k}\,. \tag{9}$$

As outlined in previous work, the ASD stream $\mathcal{S}$ performs well on video frames where the objects are entirely visible, but fails fundamentally when any parts are occluded or outside the image boundaries. We propose to mitigate the issue of occlusion by complementing the ASD stream with a spatio-temporal stream, leveraging temporal consistency when parts are occluded.

4.2. Supervised training of spatio-temporal model

Frames showing full visibility of all assembly parts are rare, yet previous work relies entirely on such frames [43]. To avoid this, we propose directly optimizing a spatio-temporal model to recognize the correct completion of procedure steps. By directly recognizing correctly completed

procedure steps, only the parts relevant to each step need to be visible. Additionally, the temporal model learns to leverage temporal continuity, enabling it to use information from components previously visible, even when temporarily occluded.

The overall structure and training methodology for the spatio-temporal model are illustrated in Fig. 2. The temporal model includes a pre-trained spatial encoder $f_{\text{spatial}}(\cdot)$, a temporal encoder $f_{\text{temporal}}(\cdot)$, and a classification head. The temporal encoder comprises a transformer architecture [53]. The frames of a video clip X_i are processed by the spatial encoder $f_{\text{spatial}}(\cdot)$ and the first layer of it's projection head $g(\cdot)$ to obtain meaningful embeddings, following the approach in [9]. Analogous to BERT [14] and the vision transformer [17], these embeddings are then combined with learnable 1D positional encodings [17]. A classification token, [CLS], is added at the beginning of the spatial representation sequence to aggregate global information about the clip. After propagating this input sequence through the temporal encoder, the output of the [CLS] token is fed into the classification head. The classification head is implemented using an MLP with one hidden layer, designed for multi-label classification at the clip level. The output dimension of the classification head corresponds to the number of unique assembly components in the procedure $\mathcal{P}$, where each component is defined as the object part(s) necessary to complete a procedure step correctly.

Similar to the video transformer network (VTN) [35], the two encoders are trained separately for distinct objectives. The spatial encoder is pre-trained in a weakly supervised manner to learn meaningful spatial features from similar assembly states, as outlined in the following section. Subsequently, the temporal encoder is trained to capture temporal dynamics, by minimizing the multi-label binary cross-entropy loss $\mathcal{L}_{CE}$ to identify which components (if any) have undergone *correctly* completed procedure steps in the video clip X_i. The loss function $\mathcal{L}_{CE}$ is defined as

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\Big[y_{ij}\log\hat{y}_{ij} + (1-y_{ij})\log(1-\hat{y}_{ij})\Big], \tag{10}$$

where N is the number of samples in the batch, C the number of steps to be completed, $y_{ij} \in \{0, 1\}$ the binary ground-truth label for sample i and class j, and $\hat{y}_{ij}$ the predicted probability. A target value of zero is assigned to each procedure step that was not correctly completed within the clip, and a target value of one is assigned if it did.

To generate the step completion labels from video clips, an exclusive OR (XOR) operation is applied on the PSR labels between the first and last frame of the clip. This ensures that each clip is labeled based on the net step changes that occur within its duration. For example, if two distinct steps are completed within a single video clip, the resulting label indicates the completion of both steps. However, if a removal step occurs first within the clip, followed by the re-installation of the same component, the label indicates no net change, since the component's state at the beginning and end of the clip remains the same. This labeling methodology affects the training phase, as clips are sampled independently. During inference, however, video clips are processed consecutively, allowing the model to first recognize the completion of a removal step and later detect the subsequent installation step, once the removal action is no longer present in the sampled clip.

Similarly to [43], a temporal filter is used to only consider predictions that are consistent across multiple frames. To this end, the confidence for each predicted step completion is accumulated over time until it reaches a threshold T, at which point the step completion and time-stamp are added to $\hat{Y}$. On frames where no assembly state is recognized, the accumulated confidence scores for all incomplete steps decay at a fixed rate.

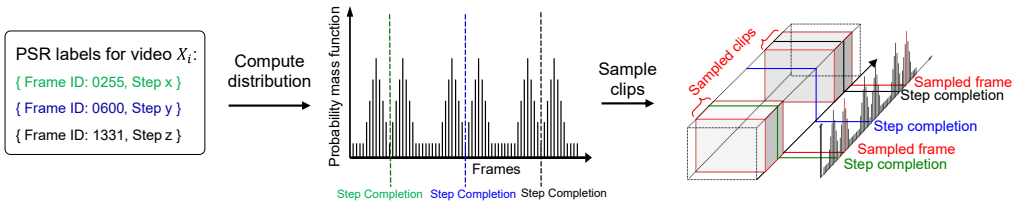


Figure 3: Principle view of generating the probability mass function with bimodal sampling used in key-clip aware sampling (KCAS). The KCAS over-samples hard-negative clips (directly before a step completion occurs) using the first Gaussian of the bimodal distribution, and positive clips (directly after the step completion) with the second Gaussian. Potentially ambiguous frames at the exact moment of step completions are sampled at a reduced rate, as well as frames where no steps are completed. The long tails of the distribution deliberately suppress background clips far from any completion, preventing the mini-batch from being dominated by uninformative content.

4.3. Key-clip aware sampling

Since most video clips do not contain step completions, uniformly sampling input clips for each mini-batch of the spatio-temporal model is inefficient. To address this, we propose using key-clip aware sampling (KCAS), which over-samples video clips around the minority classes (step completions) compared to the background class (moments without step completions). Specifically, KCAS creates a probability mass function, formed by two Gaussian distributions around each step completion time. Let $T = t_1, t_2, \cdots, t_m$ be the set of timestamps corresponding to the completions of procedure steps in video X_i, where t_j represents the completion time of step j, and δ be the separation between the two Gaussian distributions for each step completion. The probability mass function $p_i(x)$ for sampling a clip *ending at* frame x is then defined as

$$p_i(x) = \sum_{t_j \in T} \Big[g(x \mid t_j - \delta, \sigma) + g(x \mid t_j + \delta, \sigma) \Big], \tag{11}$$

where $g(x \mid \mu, \sigma)$ is the probability mass function of a Gaussian distribution with mean μ and standard deviation σ. To ensure that $p_i(x)$ is a valid probability mass function, we normalize it by dividing $p_i(x)$ by the sum over all values of x. When two PSR steps are completed at the same moment, they are seen as a single step completion for creating the sampling distributions, since the objective of KCAS is to over-sample clips where steps occur, regardless of the number of steps completed within this clip.

A bimodal distribution is chosen over a single Gaussian because of potential ambiguity regarding the exact frame of completion for a procedure step, *e.g.*the moment when a nut is tightened to the desired level. We pose that those clips prior to completion can act as hard-negative samples, since they contain important information on the appearance of nearly completed steps. Therefore, as demonstrated in Fig. 3, KCAS explicitly samples *fewer* video clips ending in the ambiguous moment, and *more* clips that end prior to and after the step completions. For each index sampled from the probability mass function, a temporal window of w frames prior to this index is created, from which N_w frames are sampled. To reduce computational cost, a stride can be applied to limit the number of sampled frames ($N_w \leq w$) [49, 55].

4.4. Weakly supervised training of spatial encoder

The goal of weakly supervised training of the spatial encoder is to encourage it to learn occlusion-robust state representations from limited supervision. Rather than requiring dense

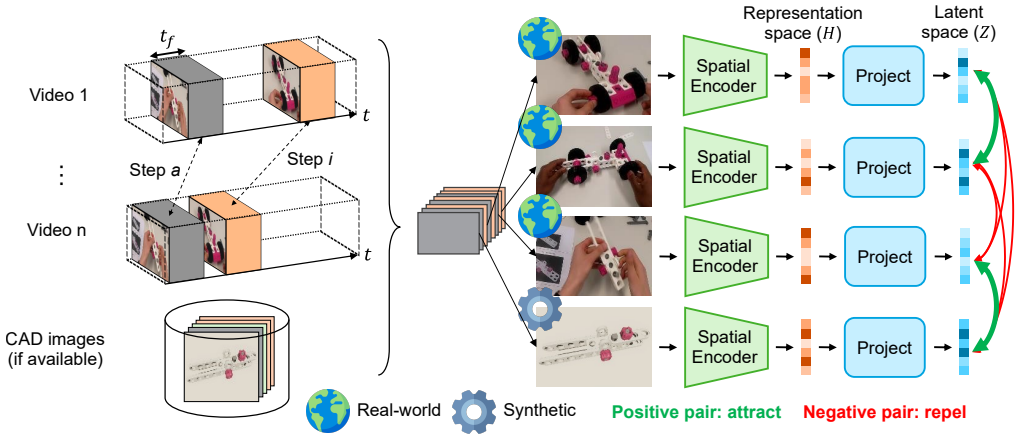


Figure 4: Overview of the key-frame sampling (KFS) weakly supervised pre-training. Each mini-batch consists of real-world images sampled around the time stamps of the PSR labels from different videos, with a sampling window t_f. When CAD data are available, synthetic images may be included. The sampled images are passed through a spatial encoder $f_{\text{spatial}}(\cdot)$, comprising a ViT-S [17] architecture, into representation space H, after which a three-layer MLP $g(\cdot)$ projects the embeddings into another space Z, on which the supervised contrastive loss [27] is calculated.

frame-level assembly state annotations, we leverage PSR labels as weak signals to group visual observations that correspond to the same procedural step, despite differences in viewpoint, occlusion, or tool presence. This enables robust feature learning from real-world observations, with synthetic data serving as an optional supplement that can enhance training.

The spatial encoder $f_{\text{spatial}}(\cdot)$ is pre-trained with contrastive learning, outlined in detail in Fig. 4. The approach, inspired by [42], encourages $f_{\text{spatial}}(\cdot)$ to project instances of the same procedure steps, observed from different angles, videos, and occlusion levels, into similar embedding representations. Each image is processed by $f_{\text{spatial}}(\cdot)$, generating spatial feature embeddings h. For $f_{\text{spatial}}(\cdot)$, an ImageNet-21K pre-trained ViT-S [17] model is employed, of which the last layer is replaced by a one-layer MLP to project the output into a 128-dimensional vector. The embeddings $h \in H$ are then projected into a new embedding space z using a nonlinear projection head $g(\cdot)$, implemented as a three-layer multi-layer perceptron (MLP), as described in [9]. The supervised contrastive loss [27] ($\mathcal{L}_{\text{SupCon}}$) is employed to maximize the agreement of embeddings $z \in Z$ between different views and videos for identical assembly states, whilst minimizing the agreement of embeddings from different states, and is defined as

$$\mathcal{L}_{\text{SupCon}} = \sum_{i \in I} -\log \frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp\left(\frac{g(f(x_i)) \cdot g(f(x_p))}{\tau}\right)}{\sum_{a \in A(i)} \exp\left(\frac{g(f(x_i)) \cdot g(f(x_a))}{\tau}\right)}, \tag{12}$$

where $A(i)$ the set of all indices in mini-batch X, and $P(i) \equiv \{p \in A(i) : (y_p = y_i)\}$ is the set of indices of all positives, and τ is the temperature constant.

Mini-batches are created through key-frame sampling (KFS), leveraging PSR labels to gather images surrounding procedural step completions, as described in Algor. 1. These PSR labels denote the actual completion of steps, regardless of the visibility of irrelevant parts of the assembled object within the frame. In contrast, assembly state detection annotations are present only when the entire object can be observed from a single image, requiring the entire object (including the

Algorithm 1: Key-frame sampling

Input: Video dataset X, PSR labels Y, temporal window per step t_f, # real-world frames per state N_sample, # assembly states N_state, # synthetic images per state N_syn
Output: Mini-batch with images x, labels y

```
x ← empty list
y ← empty list
FPS ← 10 /*Constant */
for stateID ∈ {1 ··· N_state} do
    images← empty list
    syn_images← empty list
    indices ← indices from in Y with stateID
    for i ∈ indices do
        append X[i : i + t_f · FPS] to images append X[i : i + N_syn] to syn_images
    end
    for i ∈ {0 ··· (N_sample + N_syn)} do
        if i < N_state then
            x[i] ← image from images
            y[i] ← stateID
        else
            x[i] ← image from syn_images
            y[i] ← stateID
        end
    end
end
return x, y
```

irrelevant parts) to be largely unobstructed and within the frame. Thus, pre-training on the PSR labels encourages similar assembly states to be grouped based on the relevant visible parts that constitute change.

KFS is weakly supervised, because it relies on sparse procedural step annotations rather than dense, per-frame supervision of assembly states. Specifically, the duration of visibility for the parts constituting a step completion is not annotated, so KFS must assume a fixed temporal window, denoted by t_f and expressed in seconds, after each step completion. As a result, the sampled frames are coarsely annotated, meaning there is no guarantee that they capture the actual parts undergoing change, except for the first frame (t_f=0s). This t_f is the main hyperparameter of KFS. When the value is increased, more variability in occlusion levels and viewpoints are provided, potentially leading to richer feature embeddings. However, this may introduce redundant or noisy samples that do not constitute meaningfully to the step completion.

Further hyperparameters of KFS are the number of real-world samples per step completion in each mini-batch (N_sample). Additionally, if synthetic data (*e.g.* CAD models) are available, the hyperparameter that determines the number of generated images N_syn can be set to readily incorporate the synthetic images in the spatial encoder training.

4.5. Assembly state detection stream

As outlined in Fig. 1, STORM-PSR combines the spatio-temporal stream ($\mathcal{T}$) with an assembly state detection stream ($\mathcal{S}$). This ASD stream comprises the object detection-based PSR

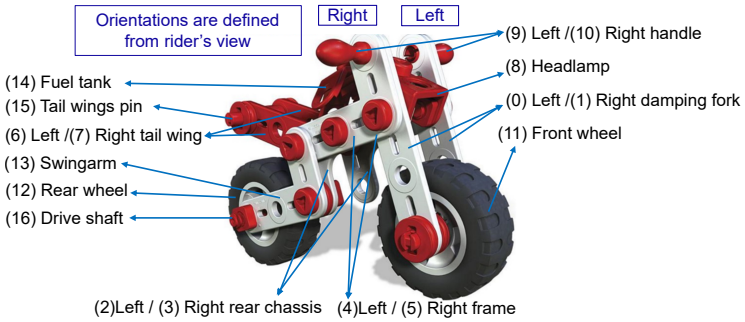


Figure 5: Definition of the 17 toy-motorcycle model assembly components in MECCANO.

baseline from [43], based on a YOLOV8-m backbone [26]. Specifically, this stream indirectly recognizes procedure steps $\hat{y}_S$ in X by determining which steps in $\mathcal{P}$ are required to transform the previously observed assembly state s_{i-1} to the newly recognized state s_i. While the spatio-temporal stream enhances robustness to occlusion, clean frames – in which the entire object is visible – still provide valuable and reliable assembly state information. By integrating both streams, STORM-PSR effectively leverages the strengths of each, using spatial cues when available and relying on temporal consistency when visibility is limited.

5. Experiments

In this section, the two-stream framework for procedure step recognition is implemented and evaluated on the IndustReal [43] and MECCANO [36] datasets. Both datasets contain videos of construction-set toy assemblies recorded from an egocentric perspective. A comparison on specifically these two datasets is interesting for three reasons. First, operators in IndustReal are instructed to periodically provide unobstructed views of the toy car, which is not the case in MECCANO. Therefore, MECCANO contains significantly more occluded frames, making it a more challenging dataset for PSR. Second, the MECCANO dataset is 17% longer (in hours) than IndustReal, but includes approximately half of the step completions and annotated ASD frames. Finally, IndustReal offers synthetic data for all assembly states, providing additional data for the pre-training. Hence, although the datasets are similar, STORM-PSR faces distinct challenges with each dataset.

5.1. MECCANO annotation

Because MECCANO does not include the required ASD and PSR annotations, we have manually annotated the dataset for both tasks, according to the same procedure as [43]. These new labels are made publicly available to stimulate research on PSR. For the MECCANO toy motorcycle, 11 assembly states consisting of 17 components, outlined in Fig. 5.

To create the PSR labels, annotators were instructed to mark the assembly states and frame indexes when the participants complete an installation or removal action for each component. Figure 6 demonstrates the installation, removal, and incorrectly installed class instances. A long-tail distribution is observed, where the expected 17 correct installation actions (one per component) constitute approximately 85% of the dataset. The removal actions are scarce since they only occur after (infrequent) execution errors. Based on the occurrences of the step completions, it is

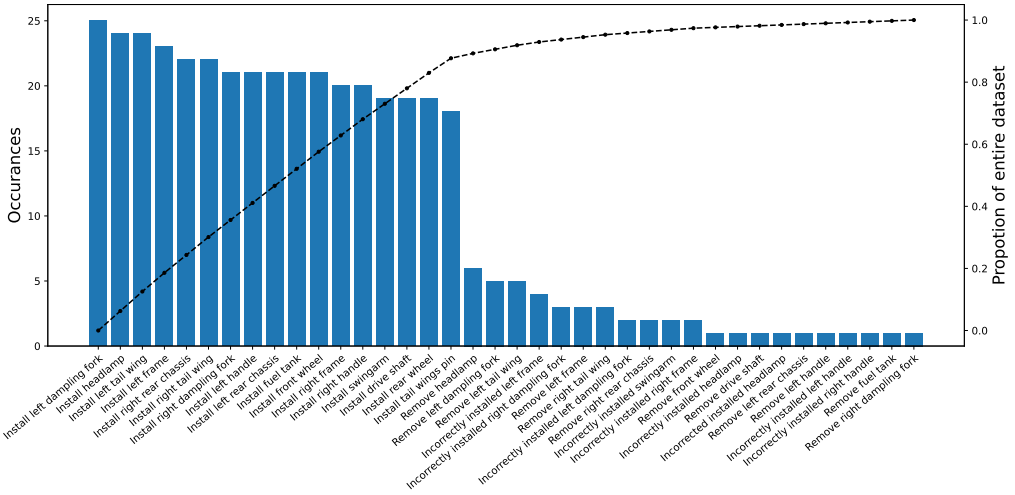


Figure 6: Distribution of the new procedure step recognition labels for the MECCANO dataset.

possible to see where mistakes are commonly made. For instance, the action "remove headlamp" is commonly performed, indicating that a mistake is frequently made after "install headlamp" to correct it. In total, the dataset contains 431 correctly executed procedure steps (25±4.0 steps per recording) and 30 incorrect step completions (1.5±0.9 per recording).

For the ASD annotations, the same procedure for annotation as for IndustReal is used. Therefore, bounding boxes are only created if a human annotator can recognize the assembly state based on a single image, without using further temporal knowledge from the assembly videos. In total, 13.6K image frames are annotated with a bounding box, which is approximately half of the frames annotated for IndustReal (26.9K frames). The training set contains 6,948 images, the validation set contains 3,053 images, and the test set comprises 3,656 images. A benchmark performance for ASD on the MECCANO dataset is provided in Section Appendix A.

5.2. Implementation details

Both the spatial and the temporal encoder are optimized using SGD [40] with an initial learning rate of 10^{-3}, momentum of 0.9, five warm-up epochs, and a cosine-annealing learning rate scheduler. All input images are resized to 224×224 pixels. The encoders are trained separately for 100 epochs, after which the spatial encoder is unlocked with learning rate 10^{-5}, to jointly converge further for 75 epochs.

For the spatial encoder, pre-trained with KFS, the number of frames eligible to be sampled after a step completion is empirically set to t_f=2 seconds and the number of states per batch is set to the total number of unique states in the dataset. Per state, N_{sample}=16 images are sampled if no synthetic data are used, otherwise N_{sample} and N_{syn} are set to 8. A margin of 0.01 and temperature of 0.07 are set for the SupCon loss, and the best epoch is selected based on the precision on the validation set.

For the temporal encoder, 6 stacked self-attention layers, 8 attention heads, and an MLP output size of 4,096 is used. The batch size B and temporal window size w are both set to 256. Within this window, N_w=64 spatial embeddings are sampled with equal spacing. For KCAS,

Method	IndustReal [43]			MECCANO [36]		
	POS↑	F_1 ↑	τ ↓	POS↑	F_1 ↑	τ ↓
IndustReal [43]	<u>0.797</u>	<u>0.891</u>	21.0	<u>0.354</u>	**0.545**	<u>99.8</u>
Spatio-temp. stream	0.497	0.506	**14.2**	0.206	0.247	120.3
STORM-PSR	**0.812**	**0.901**	<u>15.5</u>	**0.377**	<u>0.497</u>	**88.6**

Table 1: Performance of STORM-PSR and its two streams for procedure step recognition on the IndustReal [43] and MECCANO [36] datasets. **Bold** text indicates the highest performance and <u>underlined</u> text indicates the second best.

the standard deviation σ is set to 45 frames, and δ, the separation between the two Gaussian distributions, is set to 80. The best checkpoint is selected based on the F_1 score on the validation set.

As in [43], predictions are accumulated over consecutive video frames until a cumulative confidence score of T is reached. Confidence scores for prediction-less frames are decayed at a rate of 25%. For PSR inference with the two-stream approach, the thresholds T=6.0 for IndustReal and T=1.0 for MECCANO are established based on empirical results on the validation set. STORM-PSR operates at 75.1 frames/second on a NVIDIA A100 GPU.

5.3. Procedure step recognition results

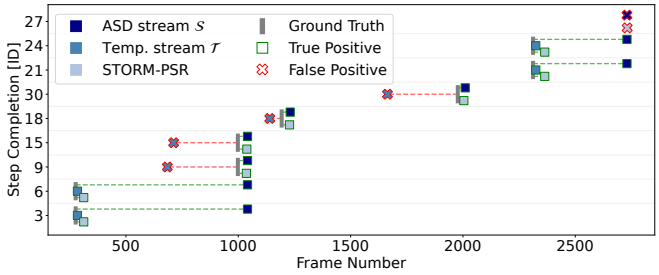
Table 1 presents the performance of STORM-PSR across both datasets. STORM-PSR reduces the average delay (τ) by 26.1% for IndustReal and 11.2% for MECCANO, outperforming the state-of-the-art baseline. Simultaneously, STORM-PSR achieves state-of-the-art performance on IndustReal for the POS and F_1 metrics. On the MECCANO dataset, STORM-PSR improves the POS (+6%) but reduces the F_1 score (-9%).

The temporal stream alone does not outperform the object detection stream on either dataset. This is likely because the datasets contain relatively few step completions, compared to the number of ASD-annotated frames. Additionally, spatio-temporal features are more challenging to learn due to increased dimensionality, the need to learn temporal coherence, and computational complexity [6, 25, 24]. These factors lead to more false positives for the temporal stream compared to the ASD stream. Combining the object-detection and temporal streams achieves state-of-the-art performance by leveraging the strengths of both approaches.

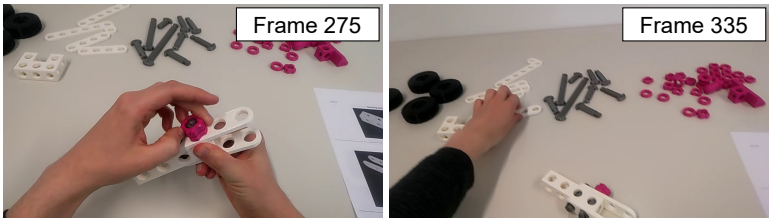
A qualitative example of predictions from both individual streams and their combined output (STORM-PSR) is provided in Fig. 7, along with images illustrating typical occlusions types. This example clearly demonstrates how the two streams complement each other. When the temporal stream predicts with high confidence, while the ASD stream predicts with lower confidence (*e.g.*due to occlusions), procedure steps can still be recognized. The STORM-PSR accumulates the confidences from both streams (reducing false positives), allowing steps to be recognized when either stream is highly confident, or both streams maintain moderate confidence over several frames.

5.4. Assembly state detection on MECCANO

The assembly state detection stream $\mathcal{S}$, using a YOLOv8-M model, is also evaluated for ASD on the MECCANO dataset [36]. The model obtains a mean average precision (mAP@50) of only 0.120, compared to 0.838 for the best model on the IndustReal dataset [43]. The latter is trained on real-world data, supplemented with images generated using the provided CAD



(a) PSR predictions from the ASD and spatio-temporal streams, as well as STORM-PSR, which combines the both streams.



(b) Two video frames demonstrating the completion of steps 3 and 6.

Figure 7: Qualitative demonstration of STORM-PSR on the IndustReal test set. The ASD stream fails to predict Steps 3 and 6, until it encounters unobstructed frames, resulting in a delay of 765 frames. The temporal stream recognizes Steps 3 and 6, but predicts false positives (with lower confidence) for other steps.

models. Nonetheless, even without leveraging these synthetic data, an mAP@50 of 0.753 is still achieved. The performance gap between IndustReal and MECCANO clearly highlights the difficulty of recognizing entire assembly states in videos with frequent occlusions, since limited visibility of the object parts significantly impairs ASD models.

5.5. Ablation studies

The following ablation studies isolate and quantify the contributions of key components within the proposed framework, including the sampling strategies (KFS and KCAS), temporal modeling backbones, sampling distributions, temporal receptive field, and key-frame sampling parameters. These experiments provide insight into design choices and their impact on procedure step recognition performance.

5.5.1. Sampling techniques and different backbones for temporal modeling

To evaluate the contributions of our sampling strategies (KFS and KCAS), as well as the the choice of temporal backbone, a three–way ablation is reported in Table 2. We compare long short-term memory networks (LSTM) [22], temporal convolutional networks (TCN) [3], and the transformer backbone. Each backbone is trained (i) without key-frame sampling (KFS) or key-clip aware sampling (KCAS), (ii) with KFS only, and (iii) with both KFS and KCAS. Finally, the end-to-end STORM-PSR results are reported for the same three settings to show how improvements in the temporal stream translate to the full two-stream system.

Spatio-temp. stream	Sampling KFS	Sampling KCAS	IndustReal [43] POS (↑)	IndustReal [43] F_1 (↑)	IndustReal [43] τ [s] (↓)	MECCANO [36] POS (↑)	MECCANO [36] F_1 (↑)	MECCANO [36] τ [s] (↓)
LSTM			0.000	0.000	-	0.000	0.000	-
LSTM	✓		0.180	0.303	33.2	0.047	0.146	222.8
LSTM	✓	✓	0.204	0.365	40.9	0.037	0.248	121.6
TCN			0.000	0.000	-	0.000	0.000	-
TCN	✓		0.124	0.185	25.4	0.000	0.000	-
TCN	✓	✓	0.195	0.414	49.4	0.134	0.181	153.7
Transformer			0.000	0.000	-	0.000	0.000	-
Transformer	✓		0.356	0.419	24.4	0.180	0.138	283.7
Transformer	✓	✓	0.497	0.506	**14.2**	0.206	0.247	120.3
STORM-PSR			0.467	0.511	62.6	0.269	0.358	<u>102.9</u>
STORM-PSR	✓		<u>0.766</u>	<u>0.892</u>	30.0	<u>0.329</u>	<u>0.459</u>	135.8
STORM-PSR	✓	✓	**0.812**	**0.901**	<u>15.5</u>	**0.377**	**0.497**	**88.6**

Table 2: An overview of the impact that different backbones, KFS, and KCAS have on the performance of the temporal model. The evaluated backbones consist of a long short-term memory (LSTM), temporal convolutional network (TCN) and transformer temporal model. Additionally, the performance of STORM-PSR (with the transformer backbone) is reported without KFS and KCAS, with only KFS, and with both. **Bold** text indicates the highest performance and <u>underlined</u> text indicates the second best.

The impact of key-frame sampling (KFS) and key-clip aware sampling (KCAS) on the temporal stream performance is highlighted in Table 2. Without KFS pre-training of the spatial encoder, the temporal encoder is not able to learn meaningful spatio-temporal features and fails to make any PSR predictions. With KFS pre-training, but without KCAS, the temporal encoder is able to learn from the meaningful spatial features. However, a clear improvement in performance is noted when KCAS is used to effectively over-sample positive and hard-negative clips. On both datasets, both the POS and F_1 improve 14% to 79% when KCAS is used.

Among the evaluated temporal backbones, the transformer consistently outperforms the LSTM and TCN across all metrics when paired with both key-frame and key-clip aware sampling. On the IndustReal dataset, the transformer achieves the highest POS (0.497), F_1 (0.506) score, and the lowest delay (14.2s), with the LSTM and TCN backbones performing between 18% and 250% worse. On MECCANO this trend is less pronounced due to the generally decreased scores on that dataset. These results indicate that the transformer is better suited for capturing long-range temporal dependencies required for recognizing procedure step completions under partial occlusions, compared to the LSTM and TCN.

5.5.2. Key-clip sampling for temporal modeling

A second ablation study evaluates the impact of the KCAS probability mass function by comparing uniform, Gaussian, and bi-modal sampling methods. In this experiment, the same spatial encoder weights are used for all three sampling methods. After training the temporal encoder, both encoders are jointly fine-tuned for 75 epochs. As shown in Table 3, the results confirm the hypothesis that sampling before and after key moments performs best, with the bi-modal distribution achieving the best performance across most metrics. This outcome is in line with expectations, since the temporal encoder is exposed to more positive samples during training compared to uniform sampling. In comparison to Gaussian sampling, KCAS selects fewer ambiguous clips by lowering the probability at the moment of step completions, while increasing

Sampling	IndustReal [43]			MECCANO [36]		
	POS↑	F_1 ↑	τ ↓	POS↑	F_1 ↑	τ ↓
Uniform	0.356	0.419	24.4	0.180	0.138	283.7
Gaussian	0.419	0.382	22.4	**0.212**	0.175	143.7
KCAS	**0.497**	**0.506**	**14.2**	0.206	**0.247**	**120.3**

Table 3: Comparison of two baseline sampling approaches and the novel key-clip aware sampling (KCAS). For uniform sampling, clips are randomly selected, regardless of the corresponding class label. For Gaussian sampling, all clips with at least one step completion are regarded as positive samples, which are over-sampled accordingly. KCAS uses a bimodal sampling distribution, to explicitly over-sample (i) hard negative samples directly before step completions and (ii) positive samples, while under-sampling the negatives.

Model	IndustReal [43]			MECCANO [36]		
	POS↑	F_1 ↑	τ ↓	POS↑	F_1 ↑	τ ↓
MLP-16	0.226	0.346	42.2	0.168	0.166	162.7
MLP-64	0.285	0.357	34.0	0.202	0.130	172.3
MLP-256	**0.407**	0.330	**23.0**	0.249	0.156	115.5
LSTM-16	0.121	0.238	27.2	0.190	0.189	194.5
LSTM-64	0.200	0.286	30.2	0.173	0.207	258.9
LSTM-256	0.277	0.470	30.3	0.156	**0.225**	132.4
TCN-16	0.119	0.144	34.2	0.172	0.113	163.9
TCN-64	0.203	0.235	25.6	0.185	0.130	186.1
TCN-256	0.265	0.502	43.1	0.195	0.151	127.8
Tr.-16	0.228	0.218	38.1	0.207	0.212	125.7
Tr.-64	0.304	0.364	45.6	0.289	0.212	137.0
Tr.-256	0.406	**0.514**	25.3	**0.351**	0.163	**108.5**

Table 4: Comparison of PSR performance of a multi-layer perceptron (MLP), long short-term memory (LSTM), temporal convolutional network (TCN) and transformer (Tr.) temporal model. Each model is trained with three receptive fields, namely 16, 64, and 256 frames, which is indicated behind the model abbreviation (*i.e.* Tr-16 is the transformer model with a window size of 16 frames).

hard-negative samples. This encourages the temporal encoder to learn subtle yet critical differences that constitute step completions.

5.5.3. Temporal receptive field

The temporal encoder is tasked with recognizing (potentially occluded) steps based on spatial features within a given temporal window of size w. This final ablation study examines the effect of the size of the temporal window on the performance for the PSR task. Furthermore, the study compares temporal self-attention using a transformer to temporal aggregation of spatial features with a simple MLP, an LSTM, and an TCN. These models are trained with the same spatial encoder for different temporal window sizes w, with KCAS. All frames within the window are sampled (w=N_w), except when w=256, where N_w=64 is obtained by applying a temporal stride of 4 to reduce the computational load. Finally, the same spatial encoder is used for all experiments and is not fine-tuned together with the temporal encoder.

The results of this test, presented in table 4, demonstrate that increasing the temporal window

t_f [s]	IndustReal [43]			MECCANO [36]		
	POS↑	F_1 ↑	τ ↓	POS↑	F_1 ↑	τ ↓
0.5	0.308	0.511	43.3	<u>0.193</u>	<u>0.241</u>	192.8
1.0	0.311	<u>0.514</u>	48.0	0.159	0.199	<u>114.7</u>
2.0	**0.406**	<u>0.514</u>	**25.3**	**0.351**	0.163	**108.5**
4.0	<u>0.350</u>	**0.526**	<u>39.8</u>	0.127	0.233	252.2
8.0	0.346	0.508	52.7	0.126	**0.298**	150.4

Table 5: Ablation study on the time after a step completion to sample from for the weakly-supervised training of the spatial encoder.

size w improves performance. This is in line with expectations, since a larger temporal window offers more context to identify step completions. Moreover, the attention-based temporal encoder outperforms the aggregation-based MLP with the same temporal window, although the difference is not substantial.

5.5.4. Time after step completions to sample for KFS

The time after a step completion from which to sample real-world frames, denoted as t_f, is an important hyperparameter in the weakly-supervised key-frame sampling (KFS) strategy. It determines the diversity and quantity of visual representations available during contrastive pretraining of the spatial encoder. A larger t_f introduces more variability in occlusion levels, viewpoints, and context, potentially leading to richer and more robust feature embeddings. However, excessively increasing this value may introduce redundant or noisy samples that dilute the discriminative signal of step completions. This ablation study investigates how varying t_f affects the downstream performance of the spatio-temporal stream and the overall STORM-PSR model, aiming to identify an optimal trade-off between training efficiency and feature quality.

Table 5 shows the results for this ablation study. The results show the intricate balance between the three metrics used to evaluate PSR. Nonetheless, for both datasets, a t_f of 2 seconds results in the highest POS score and the lowest average delay τ.

6. Discussion

This work aims to reduce delays and enhance robustness of procedure step recognition algorithms by enabling step predictions even when parts of the assembly are occluded. This is achieved using a two-stream framework called STORM-PSR, where one stream focuses on robust state detection under minimal occlusions, while the second directly predicts steps based on spatio-temporal features. STORM-PSR is evaluated on two datasets, IndustReal [43] and MECCANO [36]. For the latter, PSR and ASD labels were initially unavailable, so these were manually annotated, and are published alongside this work. On IndustReal [43], STORM-PSR achieves state-of-the-art performance, and on MECCANO it establishes a new benchmark.

STORM-PSR significantly reduces the delay in recognizing the completion of steps. While the delay reduction is in line with expectations, a larger improvement for the temporal stream of STORM-PSR on the MECCANO dataset was expected due to its high number of occluded step completions. In fact, a 26% delay reduction (τ) is observed on IndustReal, compared to only 11%

on MECCANO. Simultaneously, STORM-PSR improves the performance on both other metrics (F_1-score and POS) compared to the baseline. On the other hand, STORM-PSR on the MECCANO dataset only improves the POS metric, while reducing the F_1-score by approximately 0.05. On this point, the interaction between the three metrics is important to highlight, because the average delay τ can only be calculated on true positive recognitions of the model [43]. Therefore, while the benefit of STORM-PSR on IndustReal is readily observed (since all metrics improve), the benefit on MECCANO is less pronounced and more complex to understand. First, spatio-temporal models generally require a larger amount of data compared to models relying exclusively on spatial features, without the complexity of temporal learning [24, 6]. The MECCANO and IndustReal are of approximately the same total video length, but MECCANO contains only 1.1 procedure steps are completed per minute of video data, compared to 2.2 steps per minute in IndustReal. We hypothesize that this higher data sparsity in MECCANO poses challenges for spatio-temporal learning, disproportionally affecting the spatio-temporal stream (and thereby STORM-PSR) compared to the ASD stream, that only relies on spatial features. This difference is further amplified by the lack of synthetic data for MECCANO, which is used in STORM-PSR on IndustReal during pre-training of the spatial encoder. Second, the reduction in delay obtained by MECCANO may occasionally lead to earlier predictions that slightly precede true step completions, counting them as false positive predictions and thus negatively impacting F_1-score. Finally, the frequency and nature of occlusions in the MECCANO dataset differs from that in the IndustReal dataset on three important aspects. Firstly, operators in IndustReal, contrary to MECCANO, are asked to provide unobstructed views with some frequency, resulting in more frequent images without occlusions. Secondly, MECCANO participants use a tool, whereas the participants in the IndustReal dataset only use their hands, creating a different type of occlusion. Thirdly, the IndustReal dataset is recorded with a projected virtual box that outlines the FoV of the cameras. Therefore, participants at least somewhat consistently keep the object of interest within this FoV. Contrarily, videos in the MECCANO dataset frequently do not contain any frames with the object visible for the entire length of the temporal window, limiting the performance of STORM-PSR on this data.

To address this data sparsity in the MECCANO dataset without additional video recordings and labor-intensive annotations, MECCANO could be further extended with synthetic data. Given the commercial application of the MECCANO parts, using CAD models for creating synthetic data might not be feasible. Instead, approaches such as neural radiance fields (NeRFs) [33] may allow for synthetic view generation based on real-world captures, providing an alternative to CAD-based synthesis. Additionally, video augmentations can be used to increase data variability [28, 7].

Further experiments may explore larger temporal windows, as the largest tested in this study spans 256 frames, corresponding to video clips of 25.6 seconds in IndustReal and 21.3 seconds in MECCANO. Considering the time between step completions in both datasets, increasing this window size may be beneficial. To use larger temporal windows without encountering computational limitations, frames can be sampled at a reduced rate. Alternatively, a selection procedure can be applied to retain only the most informative frames (*e.g.*through attention mechanisms), since not all frames are equally valuable [56, 23], and approaches that consider the entire video sequence can be explored [16]. The computation cost of STORM-PSR compared to the baseline, consisting only of the ASD stream, is significantly higher. Specifically, the ASD stream operates at 284.8 FPS on an A100 GPU, while STORM-PSR with the transformer backbone operates at 75.1 FPS. With the LSTM backbone, STORM-PSR achieves 72.9 FPS, and with a TCN backbone it operates at 75.2 FPS. This trade-off between computational complexity and improved

recognition delay is an important factor for practical applicability. Although STORM-PSR requires more computation, its ability to substantially reduce recognition delay provides a clear advantage in scenarios where timely predictions are essential.

Another interesting direction of future research is regarding the fusion of the two streams. In the present work, a linear late-fusion approach is adopted to reduce the risk of overfitting, particularly given the limited size of the two evaluated datasets. Introducing learning-based fusion techniques can allow the model to weight the two streams differently to optimally exploit the complementary strengths of either stream. Especially in the context of larger-scale data, such adaptive fusion mechanisms may further improve performance. Another promising direction for future work lies in adapting existing methods originally developed for related tasks to the PSR setting. While these methods were not designed for PSR and require significant adjustments, they can offer comparative baselines and help contextualize the strengths and limitations of STORM-PSR.

7. Conclusion

This work proposes STORM-PSR, a novel approach to procedure step recognition based on spatio-temporal features. The STORM-PSR framework combines assembly state detections, which are robust when objects are entirely visible and unoccluded, with a temporal model to predict step completions. This is the first work to directly optimize for PSR, rather than inferring step completions from observations of entire assembly states. To effectively train the temporal model, we introduce key-frame and key-clip aware sampling strategies. STORM-PSR is evaluated on an existing PSR dataset, obtaining state-of-the-art performance, and on a new dataset where it sets a benchmark.

The results demonstrate that waiting for unobstructed views of entire assembly states leads to significant delays between the completion and corresponding recognition of procedure steps. We demonstrate that this delay can be reduced using a spatio-temporal model capable of learning from visible assembly parts, without requiring assumptions about occluded parts. A limitation of the proposed method is that the model does not explicitly learn the entire procedure due to its restricted temporal receptive field. Furthermore, to learn the spatio-temporal dynamics, the model requires significant training examples, which are cumbersome to collect and annotate. In future work, this data scarcity may be mitigated by, *e.g.*using a video-frame selection procedure that retains only the most informative frames for processing by the temporal network.

Acknowledgements

This work is partially executed ASML Research and received funding from ASML and TKI grant number TKI2112P07.

References

[1] Kumar Ashutosh, Santhosh Kumar Ramakrishnan, Triantafyllos Afouras, and Kristen Grauman. Video-mined task graphs for keystep recognition in instructional videos. *Advances in Neural Information Processing Systems*, 36, 2024.

[2] Emad Bahrami, Gianpiero Francesca, and Juergen Gall. How much temporal long-term context is needed for action segmentation? In *ICCV*, pages 10351–10361, 2023.

[3] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[4] Siddhant Bansal, Chetan Arora, and CV Jawahar. My view is the best view: Procedure learning from egocentric videos. In *ECCV*, pages 657–675. Springer, 2022.

[5] Yizhak Ben-Shabat, Xin Yu, Fatemeh Saleh, Dylan Campbell, Cristian Rodriguez-Opazo, Hongdong Li, and Stephen Gould. The ikea asm dataset: Understanding people assembling furniture through actions, objects and pose. In *WACV*, pages 847–859, 2021.

[6] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pages 6299–6308, 2017.

[7] Nino Cauli and Diego Reforgiato Recupero. Survey on videos data augmentation for deep learning models. *Future Internet*, 14(3):93, 2022.

[8] Min-Hung Chen, Baopu Li, Yingze Bao, and Ghassan AlRegib. Action segmentation with mixed temporal domain adaptation. In *WACV*, pages 605–614, 2020.

[9] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

[10] Grazia Cicirelli, Roberto Marani, Laura Romeo, Manuel García Domínguez, Jónathan Heras, Anna G Perri, and Tiziana D'Orazio. The ha4m dataset: Multi-modal monitoring of an assembly task for human action recognition in manufacturing. *Scientific Data*, 9(1):745, 2022.

[11] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE TPAMI*, 43(11):4125–4141, 2021.

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

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

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Tech.s, Vol. 1*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[15] Guodong Ding, Fadime Sener, and Angela Yao. Temporal action segmentation: An analysis of modern techniques. *IEEE TPAMI*, 2023.

[16] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634, 2015.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[18] Alessandro Flaborea, Guido Maria D'Amely Di Melendugno, Leonardo Plini, Luca Scofano, Edoardo De Matteis, Antonino Furnari, Giovanni Maria Farinella, and Fabio Galasso. Prego: online mistake detection in procedural egocentric videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18483–18492, 2024.

[19] Mingyu Fu, Wei Fang, Shan Gao, Jianhao Hong, and Yizhou Chen. Edge computing-driven scene-aware intelligent augmented reality assembly. *The Int. J. of Adv. Manufacturing Technology*, 2022.

[20] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.

[21] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-exo4d: Understanding skilled human activity from first-and third-person perspectives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19383–19400, 2024.

[22] Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[23] Yizeng Han, Dongchen Han, Zeyu Liu, Yulin Wang, Xuran Pan, Yifan Pu, Chao Deng, Junlan Feng, Shiji Song, and Gao Huang. Dynamic perceiver for efficient visual recognition. In *ICCV*, pages 5992–6002, 2023.

[24] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, pages 6546–6555, 2018.

[25] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE TPAMI*, 35(1):221–231, 2012.

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

[27] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *NeurIPS*, pages 18661–18673. Curran Associates, Inc., 2020.

[28] Youngjoong Kwon, Stefano Petrangeli, Dahun Kim, Haoliang Wang, Eunbyung Park, Viswanathan Swaminathan, and Henry Fuchs. Rotationally-temporally consistent novel view synthesis of human performance video. In *ECCV*, pages 387–402. Springer, 2020.

[29] M. Leo, G. Medioni, M. Trivedi, T. Kanade, and G.M. Farinella. Computer vision for assistive technologies. *Computer Vision and Image Understanding*, 154:1–15, 2017.

[30] Marco Leo, Antonino Furnari, Gerard G. Medioni, Mohan Trivedi, and Giovanni M. Farinella. Deep learning for assistive computer vision. In *Computer Vision – ECCV 2018 Workshops*, pages 3–14, Cham, 2019. Springer International Publishing.

[31] Xudong Lin, Fabio Petroni, Gedas Bertasius, Marcus Rohrbach, Shih-Fu Chang, and Lorenzo Torresani. Learning to recognize procedural activities with distant supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13853–13863, 2022.

[32] Hangfan Liu, Yongzhi Su, Jason Rambach, Alain Pagani, and Didier Stricker. Tga: Two-level group attention for assembly state detection. In *ISMAR-Adjunct*, 2020.

[33] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[34] Kumaranage Ravindu Yasas Nagasinghe, Honglu Zhou, Malitha Gunawardhana, Martin Renqiang Min, Daniel Harari, and Muhammad Haris Khan. Why not use your textbook? knowledge-enhanced procedure planning of instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18816–18826, 2024.

[35] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. *CoRR*, abs/2102.00719, 2021.

[36] 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 *WACV*, 2021.

[37] Ivan Rodin, Antonino Furnari, Dimitrios Mavroeidis, and Giovanni Maria Farinella. Predicting the future from first person (egocentric) vision: A survey. *Computer Vision and Image Understanding*, 211:103252, 2021.

[38] Laura Romeo, Roberto Marani, Anna Gina Perri, and Juergen Gall. Multi-modal temporal action segmentation for manufacturing scenarios. *Engineering Applications of Artificial Intelligence*, 148: 110320, 2025.

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[40] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[41] Hannah Schieber, Shiyu Li, Niklas Corell, Philipp Beckerle, Julian Kreimeier, and Daniel Roth. ASDF: Assembly state detection utilizing late fusion by integrating 6d pose estimation. In *arXiv*, 2024.

[42] Tim J Schoonbeek, Goutham Balachandran, Hans Onvlee, Tim Houben, Shao-Hsuan Hung, Jacek Kustra, Peter HN de With, and Fons van der Sommen. Supervised representation learning towards generalizable assembly state recognition. *IEEE Robotics and Automation Letters*, 2024.

[43] Tim J Schoonbeek, Tim Houben, Hans Onvlee, Fons van der Sommen, et al. Industreal: A dataset for procedure step recognition handling execution errors in egocentric videos in an industrial-like setting. In *WACV*, pages 4365–4374, 2024.

[44] F. Sener, D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, and A. Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. *CVPR*, 2022.

[45] Anshul Shah, Benjamin Lundell, Harpreet Sawhney, and Rama Chellappa. Steps: Self-supervised key step extraction and localization from unlabeled procedural videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10375–10387, 2023.

[46] Xiangbo Shu, Jinhui Tang, Guo-Jun Qi, Wei Liu, and Jian Yang. Hierarchical long short-term concurrent memory for human interaction recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(3):1110–1118, 2019.

[47] Xiangbo Shu, Liyan Zhang, Guo-Jun Qi, Wei Liu, and Jinhui Tang. Spatiotemporal co-attention recurrent neural networks for human-skeleton motion prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):3300–3315, 2021.

[48] Xiangbo Shu, Binqian Xu, Liyan Zhang, and Jinhui Tang. Multi-granularity anchor-contrastive representation learning for semi-supervised skeleton-based action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7559–7576, 2022.

[49] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *NeurIPS*, 27, 2014.

[50] Ana Stanescu, Peter Mohr, Mateusz Kozinski, Shohei Mori, Dieter Schmalstieg, and Denis Kalkofen. State-aware configuration detection for augmented reality step-by-step tutorials. In *ISMAR*, pages 157–166, 2023.

[51] Yongzhi Su, Jason Rambach, Nareg Minaskan, Paul Lesur, Alain Pagani, and Didier Stricker. Deep multi-state object pose estimation for augmented reality assembly. In *ISMAR-Adjunct*, 2019.

[52] Jinhui Tang, Xiangbo Shu, Rui Yan, and Liyan Zhang. Coherence constrained graph lstm for group activity recognition. *IEEE transactions on pattern analysis and machine intelligence*, 44(2):636–647, 2019.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*. Curran Associates, Inc., 2017.

[54] Rosaura G VidalMata, Walter J Scheirer, Anna Kukleva, David Cox, and Hilde Kuehne. Joint visual-temporal embedding for unsupervised learning of actions in untrimmed sequences. In *WACV*, pages 1238–1247, 2021.

[55] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, pages 20–36. Springer, 2016.

[56] Yulin Wang, Zhaoxi Chen, Haojun Jiang, Shiji Song, Yizeng Han, and Gao Huang. Adaptive focus for efficient video recognition. In *ICCV*, pages 16249–16258, 2021.

[57] Qianqian Xiong, Jianjing Zhang, Peng Wang, Dongdong Liu, and Robert X Gao. Transferable two-stream convolutional neural network for human action recognition. *J. of Manufacturing Systems*, 56: 605–614, 2020.

[58] Rui Yan, Lingxi Xie, Jinhui Tang, Xiangbo Shu, and Qi Tian. Higcin: Hierarchical graph-based cross inference network for group activity recognition. *IEEE transactions on pattern analysis and machine intelligence*, 45(6):6955–6968, 2020.

[59] Yiwu Zhong, Licheng Yu, Yang Bai, Shangwen Li, Xueting Yan, and Yin Li. Learning procedure-aware video representation from instructional videos and their narrations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14825–14835, 2023.

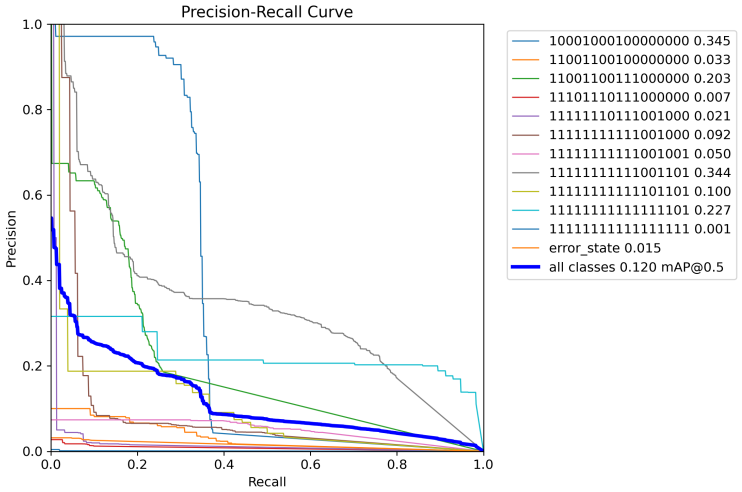
Appendix A. Assembly state detection on MECCANO

The assembly states for assembly state detection are outlined in table A.6, where each component corresponds to the components highlighted in Fig. 5 of the main paper.

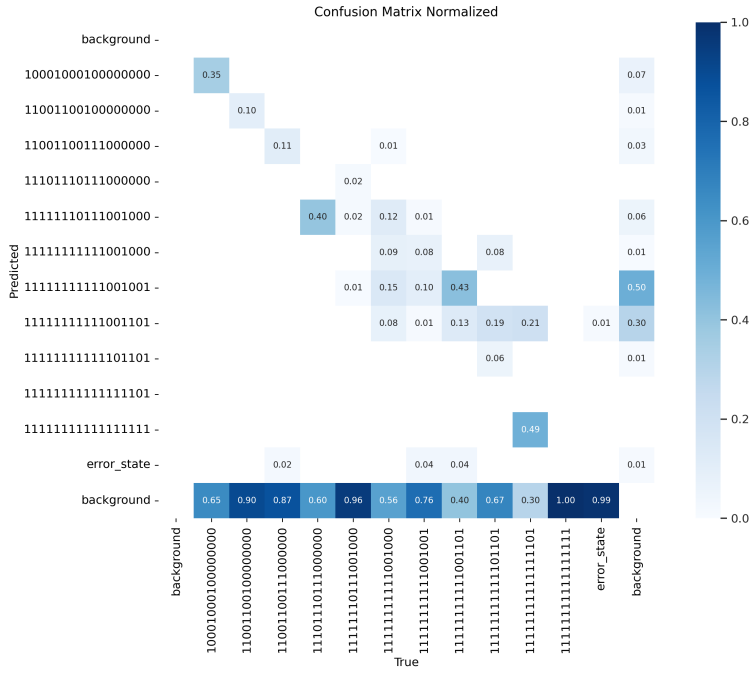
The MECCANO test-set performance of a YOLOv8-M model, trained for ASD on the newly annotated labels, is shown in Fig. A.8. These results clearly outline a performance difference with IndustReal. The model, trained for ASD on MECCANO, obtains a mean average precision (mAP@50) of 0.120, compared to 0.838 for the best model, trained on real-world and synthetic data, on the IndustReal dataset. The best model of IndustReal that does not use synthetic data, still achieves an mAP@50 of 0.753. This performance gap demonstrates the difficulty of recognizing entire assembly states on the MECCANO dataset and highlights the need for alternative approaches, such as STORM-PSR.

Table A.6: Definition of assembly state with corresponding assembly state label and components being installed compare to the previous assembly state.

Assembly state	Installed components	Assembly state label
0	Initial state	00000000000000000
1	left damping fork, left frame, headlamp	10001000100000000
2	right damping fork, right frame	11001100100000000
3	left handle, right handle	11001100111000000
4	left rear chassis, left tail wings	11101110111000000
5	right rear chassis, swimarm	11111110111001000
6	right tail wing	11111111111001000
7	driving shaft	11111111111001001
8	fuel tank	11111111111001101
9	front wheel	11111111111101101
10	rear wheel	11111111111111101
11	tail wing pin	11111111111111111



(a) Precision-recall curve.



(b) Normalized confusion matrix.

Figure A.8: Assembly state detection performance on MECCANO.