Error Recognition in Procedural Videos using Generalized Task Graph

Shih-Po Lee
Northeastern University
lee.shih@northeastern.edu

Ehsan Elhamifar
Northeastern University
e.elhamifar@northeastern.edu

Abstract

Understanding user actions and their possible mistakes is essential for successful operation of task assistants. In this paper, we develop a unified framework for joint temporal action segmentation and error recognition (recognizing when and which type of error happens) in procedural task videos. We propose a Generalized Task Graph (GTG) whose nodes encode correct steps and background (task-irrelevant actions). We then develop a GTG-Video Alignment algorithm (GTG2Vid) to jointly segment videos into actions and detect frames containing errors. Given that it is infeasible to gather many videos and their annotations for different types of errors, we study a framework that only requires normal (error-free) videos during training. More specifically, we leverage large language models (LLMs) to obtain error descriptions and subsequently use video-language models (VLMs) to generate visually-aligned textual features, which we use for error recognition. We then propose an Error Recognition Module (ERM) to recognize the error frames predicted by GTG2Vid using the generated error features. By extensive experiments on two egocentric datasets of EgoPER and CaptainCook4D, we show that our framework outperforms other baselines on action segmentation, error detection and recognition.

1. Introduction

The integration of computer vision with augmented and virtual reality (AR & VR) is expected to significantly improve the way we learn and develop skills in our daily lives [84]. Imagine wearing a pair of AR glasses and trying to assemble a new piece of furniture. A visual task assistant should recognize your actions and provide guidance on what to do and how to perform them throughout the process. It should also recognize your possible errors and provide feedback to correct them, when needed. This has motivated exciting new research on action recognition [8, 9, 24, 25, 46, 49, 60, 90] and segmentation [22, 37, 44, 45, 53, 56, 57, 74, 78, 79, 85, 96, 97, 104], video grounding [5, 17, 19, 48, 58, 62, 87, 109], learning from instructional videos [11, 32, 61, 63, 64, 86, 103] and very recently on error understanding [13, 26, 28, 40, 59, 91], as well as progress [16] and state [101] prediction.

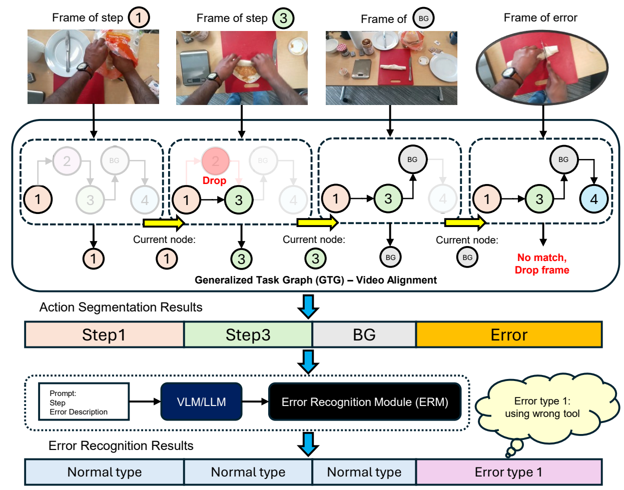

Figure 1. The illustration of our framework for joint temporal action segmentation and error recognition. It consists of 1) Generalized Task Graph (GTG)-Video Alignment for action segmentation and 2) an Error Recognition Module using features from LLM/VLM for error recognition.

Successful operation of a visual task assistant, in particular, requires temporal action segmentation (assigning step labels to frames) and error recognition (recognizing the type of error, if any, in each frame) to seamlessly work together. Additionally, a task graph, which encodes plausible ways to perform a task, has been shown to be critical for temporal action segmentation to effectively mitigate oversegmentation [5, 18, 78, 106]. However, existing methods have three major limitations when it comes to understanding actions and errors.

SOTA Limitations. First, existing works on error understanding can only detect the presence of an error in a video frame without recognizing the type of the error. As recently studied in [40], different types of errors, such as omission, addition, modification, and slip errors, can occur during task execution. However, [40] cannot distinguish between the error types. A task assistant requires understanding the error type to provide appropriate feedback. For example, an actor knocking over a dripper while making coffee requires immediate feedback for correction or spreading jelly onto tortilla using spoon instead of knife needs a feedback after task completion for the actor to better execute the step.

Second, given the large number of possible errors, it is infeasible to gather real training videos and annotations for all possible errors. Also, recent datasets [40, 66] that contain videos with errors have imbalance annotations on different error types, making error recognition a challenging task, especially under the setting when we do not have error videos during training.

Third, recent action segmentation methods [18, 78] that use task graphs assume that there are no errors in videos and assign step labels to the frames according to the task graph. In addition, two recent works that handle both action segmentation and error detection [28, 40] cannot use the task graph for action segmentation. This is because a task graph only contains the correct actions/steps and does not consider possible errors that may occur during the task execution. Thus, following the task graph leads to assigning every frame (including frames that have errors and ones that do not belong to any step) to correct actions/steps, which is undesired.

Paper Contributions. We develop a unified error recognition and temporal action segmentation framework (see Figure 1) to address the aforementioned limitations.

– We propose a Generalized Task Graph (GTG) whose nodes encode normal actions/steps and background (task-irrelevant actions).

– We develop a dynamic programming-based method, GTG-Video Alignment (GTG2Vid), with a dynamic frame and node dropping term to assign frames to GTG nodes for action segmentation and drop frames for error detection.

– Given the lack of error training videos, we leverage the rich knowledge of large language models (LLMs) in conjunction with the rich representation power of video language models (VLMs) to obtain error descriptions for different types of errors and their corresponding features for error recognition.

– We propose an Error Recognition Module (ERM) to recognize the error types, if any, in the frames predicted by GTG2Vid using joint similarities of video frame, normal type, and error type features.

– By extensive experiments on two egocentric datasets, EgoPER and CaptainCook4D, we show that *without using error videos during training*, our method 1) improves the performance of temporal action segmentation on error videos, 2) improves error detection, and 3) recognizes different errors.

2. Related Works

Temporal Action Segmentation. Learning to segment videos into several actions/steps has been studied under unsupervised [4, 20, 21, 27, 35, 36, 75, 80, 89, 92, 95, 109], weakly-supervised [10, 15, 23, 41–43, 47, 54, 54, 55, 70–72, 77, 85], and fully-supervised [3, 7, 14, 22, 30, 31, 34, 37, 44, 45, 50, 56, 65, 69, 74, 81–83, 85, 96, 97, 104] setting. Since procedural step videos follow certain task graphs to complete the task, some prior works [5, 18, 78, 106] additionally adopt task graph information to guide segmentation process. Specifically, G2Vid [18] aims to assign steps/nodes in a task graph to video frames and drop the frames not corresponding to any node. However, G2Vid cannot drop unnecessary nodes in a task graph nor map background frames to nodes to distinguish between background and error frames. In contrast, our framework adds *extra nodes* to a task graph and allows droppable nodes to properly assign nodes to frames.

Error Detection and Recognition for Procedural Tasks. Detecting errors for procedural tasks has recently drawn attention from the video understanding community. Recent works [13, 28, 68, 91] have released relevant datasets to facilitate error understanding. Meanwhile, several works [26, 40, 59, 76] have developed frameworks to detect if errors are present in videos and localize their temporal locations. Recent work [26] focuses on online error detection using action recognition and anticipation models based on videos and action sequences. Another work [40] uses only error-free (normal) videos during training to generate multiple prototypes for action segmentation and error detection. However, none of them has the ability to recognize error types. Our work is able to segment videos and recognize different errors simultaneously without error training videos.

VLMs/LLMs for Anomaly Detection. Anomaly detection in videos [2, 6, 12, 38, 39, 52, 67, 73, 88, 93, 98–100, 105, 107, 108] has been widely explored, and recent studies [33, 102] leverage VLMs and LLMs for further improvement due to their outstanding reasoning capabilities. Recent work in [102] proposes a training-free video anomaly detection framework using VLMs to generate captions over frames followed by LLMs to predict anomaly scores for those captions. Another work in [33] adopts LLMs to generate normal and abnormal descriptions under various scenes and classify frames into normal or abnormal according to the similarities between features of frames and descriptions. In our paper, we leverage LLMs to generate error descriptions for each error type of every step with crafted prompts, followed by VLMs to generate textual features aligned with visual features for error recognition.

3. Graph to Video Alignment Review

Our proposed GTG-Video Alignment method builds upon and generalizes the Graph-to-Video Alignment (G2Vid) [18], which we review here. A task graph (also called flow graph) is a directed acyclic graph $G = (V, E)$, where each node $v \in V$ represents a procedure step and each edge $(v_k, v_i) \in E$ from v_k to v_i implies that v_k must be completed before proceeding to v_i, see Figure 2(a). Thus, all predecessors of a node must be completed before starting that step. Thus, the task graph encodes all possible ways to perform a task (this can be obtained by topological sorting).

To efficiently align a video with the task graph and therefore segment the video into task steps, G2Vid [18] first builds a tSort graph (see Figure 2(a)) that provides a compact representation of all topological sorts of the task graph (each path from the source to the sink node in the tSort graph represents a topological sort of G). G2Vid alignment recovers a traversal of the tSort graph that best aligns with the video. It uses a modified version of Drop-DTW [17] to align nodes and frames, allowing for i) one-to-many matching (one node/step can be assigned to many frames) and ii) unmatched sequence elements (to be able to drop frames that do not correspond to any node/step).

Let $C_{i,j}$ denote the matching cost of node i to frame j, d_j denote the dropping cost of frame j, and $A(i)$ denote the set of predecessors of i. The dynamic programming solution computes the cumulative matching $D^+_{i,j}$, dropping $D^-_{i,j}$ and optimal $D_{i,j}$ costs up to the i-th node and j-th frame as

$$\begin{aligned} D^+_{i,j} &= C_{i,j} + \min\{\min_{k \in A(i)}\{D_{k,j-1}\}, D_{i,j-1}\}, \\ D^-_{i,j} &= d_j + D_{i,j-1}, \quad D_{i,j} = \min\{D^+_{i,j}, D^-_{i,j}\}. \end{aligned} \tag{1}$$

By iterating over all nodes and frames, the dynamic programming outputs the optimal cost and alignment by backtracking the path with minimum cost, starting from the top left and ending at the bottom right of the cumulative matrix.

4. Proposed Method

We develop a unified framework for *Temporal Action Segmentation* (*TAS*) and *Error Recognition* (*ER*) in procedural task videos. We assume having access to only normal (error-free) videos with their frame-wise step labels during training and the task graph during testing. Our proposed framework *addresses three fundamental questions*: (1) How can we build an efficient Generalized Task Graph (GTG) that enables *TAS* and *Error Detection* (*ED*) for videos? (2) How can we obtain features for different errors without collecting real error videos? (3) How can we leverage the error features to perform *ER*? In the following, we first formalize the problem, discuss the overarching goal of our framework, and discuss the details of the method to address these questions.

Problem Setting. For *TAS* and *ER*, given a sequence of video frame features $\mathcal{X}^v = (x^v_1, x^v_2, \ldots, x^v_T)$, our goal is to predict its frame-wise steps $\hat{\mathcal{Y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T)$ and error types $\hat{\mathcal{E}} = (\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_T)$, where T is the number of frames, $x^v_t \in \mathbb{R}^D$ is a pre-extracted feature vector, $\hat{y}_t \in$

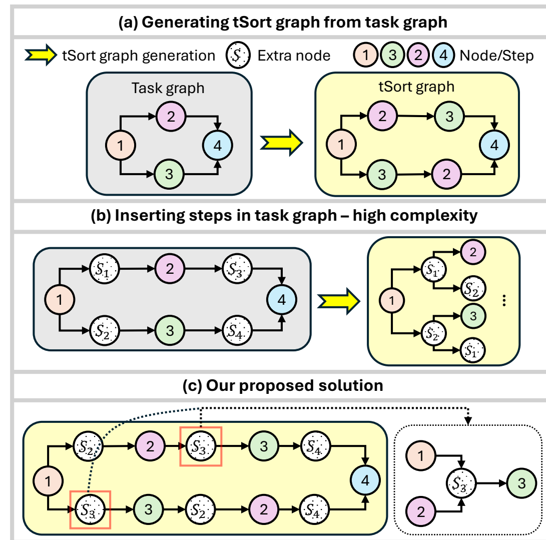

Figure 2. The illustration of (a) an original task graph and its corresponding tSort graph, (b) a naive approach to capture background, (c) our proposed solution, which specifically illustrates the predecessors and successor of *extra node* S_3.

$\{-1, 0, \ldots, S\}$, where S is the number of steps and class -1 and 0 indicate error and background (a task-irrelevant action, e.g., answering phone when making coffee), $\hat{e}_t \in \{0, \ldots, M\}$, where M is the number of error types and class 0 denotes no error happening. During training, we use only normal (error-free) videos and their ground-truth step labels $\mathcal{Y}$. For testing, we assume that the task graph is given.

Framework Overview. We introduce Generalized Task Graph for handling both *TAS* and *ED*. Our key idea is to 1) insert *extra nodes* in the tSort graph to capture the background class that may occur before each step in the task graph, 2) perform *TAS* and *ED* by our GTG2Vid with a dynamic frame and node dropping term for different types of errors, 3) obtain error descriptions and features for all types of errors, respectively, by leveraging LLMs and VLMs and 4) use an Error Recognition Module (ERM) that computes joint similarities of video, normal type and error type features obtained by VLMs to classify the error types for the frames flagged by GTG2Vid.

4.1. Generalized Task Graph (GTG)

We build GTG to handle execution and omission errors and develop GTG2Vid for *TAS* and *ED*.

Handling Execution Errors. We aim to use the task graph to segment videos into steps and to detect execution errors (e.g., slip, correction, modification, and addition errors defined in [40]). Some of the errors are step-relevant (e.g., spreading jelly using spoon instead of knife) and some are not in the task graph (e.g., adding honey to a recipe that does not include honey). However, G2Vid drops both frames with errors as well as background frames (e.g., answering

Algorithm 1 GTG-Video Alignment (GTG2Vid)

Inputs: $C \in \mathbb{R}^{2S\times T}, A(\cdot), d_i^z, d_j^x$ (matching cost matrix, predecessor mapping dictionary, node dropping costs, frame dropping costs.)
$\triangleright$ Initialize DP tables
$D^+_{0,0} = 0; D^+_{i,0} = \infty; D^+_{0,j} = \infty; \forall i \in \{1, \cdots, 2S\}, \forall j \in \{1, \cdots, T\}$ $\triangleright$ Matching table of $\mathcal{X}_n^v$ (frames) and $\mathcal{Z}$ (nodes)
$D^{z-}_{0,0} = 0; D^{z-}_{i,0} = \sum_{k=1}^{i} d_k^z; D^{z-}_{0,j} = \infty; \forall i \in \{1, \cdots, 2S\}, \forall j \in \{1, \cdots, T\}$ $\triangleright$ Dropping table of $\mathcal{Z}$ (nodes)
$D^{-x}_{0,0} = 0; D^{-x}_{i,0} = \infty; D^{-x}_{0,j} = \sum_{k=1}^{j} d_k^x; \forall i \in \{1, \cdots, 2S\}, \forall j \in \{1, \cdots, T\}$ $\triangleright$ Dropping table of $\mathcal{X}_n^v$ (frames)
$D_{0,0} = 0; D_{i,0} = D^{-x}_{i,0}; D_{0,j} = D^{z-}_{0,j}; \forall i \in \{1, \cdots, 2S\}, \forall j \in \{1, \cdots, T\}$ $\triangleright$ Optimal solution table
for $i = 1, \cdots, 2S$ **do** $\triangleright$ Iterate nodes in $\mathcal{Z}$
  for $j = 1, \cdots, T$ **do** $\triangleright$ Iterate frames in $\mathcal{X}_n^v$
    $D^+_{i,j} = -C_{i,j} + \min\{\min_{k \in A(i)} \{D_{k,j-1}\}, D_{i,j-1}\}$ $\triangleright$ Consider matching node $\mathcal{Z}_i$ to frame $x^v_{n,j}$
    $D^{-x}_{i,j} = d_j^x + D_{i,j-1}$ $\triangleright$ Consider dropping frame $x^v_{n,j}$
    $D^{z-}_{i,j} = d_i^z + \min_{k \in A(i)} \{D_{k,j}\}$ $\triangleright$ Consider dropping node $\mathcal{Z}_i$
    $D_{i,j} = \min\{D^+_{i,j}, D^{-x}_{i,j}, D^{z-}_{i,j}\}$ $\triangleright$ Select the optimal path
  end for
end for
$\hat{\mathcal{Y}}^* = \text{traceback}(D)$ $\triangleright$ Compute the optimal alignment by tracing back the path with minimum cost
$\hat{\mathcal{Y}} = \text{frame-assign}(\hat{\mathcal{Y}}^*)$ $\triangleright$ Flag dropping frames as error frames and frames belonging to *extra nodes* as background
Outputs: $D_{2S,T}, \hat{\mathcal{Y}}$

the phone while making a recipe) and thus cannot distinguish between them and understand when an error occurs.

To handle this, we build GTG by inserting *extra nodes* into the task graph to capture background frames. Therefore, by aligning a video with GTG, we can understand which frames correspond to which steps and which frames correspond to background. Additionally, by allowing dropping frames, we can detect execution errors, and by allowing dropping nodes, we can detect omission errors which we will discuss next.

A naive approach is to insert an *extra node* with a unique label between every two nodes (see Figure 2 (b)). Therefore, the task graph can potentially map frames to *extra nodes*. However, the complexity of the tSort graph grows exponentially. To solve this problem, our proposed approach inserts an *extra node* S_i with a unique ID between every node i and its predecessor in the tSort graph. For example, in Figure 2 (c), between node 2 and 3 as well as node 1 and 3, we insert the same *extra node* S_3 to capture the background frames with respect to step 3. Our design 1) prevents exponential growth of the tSort graph, only increasing the number of nodes from S to $2S$, and 2) inherits the correct execution orders in the original tSort graph.

Handling Omission Errors. During execution, an actor may skip one or more steps in the task graph, resulting in omission error. In this case, forcing models to follow the task graph as done in prior works [5, 18, 78, 106] can lead to wrong *TAS* results where some frames are assigned to the skipped steps. To handle omission errors, we introduce a node dropping term to our GTG2Vid to make nodes of our GTG droppable,

$$D^{z-}_{i,j} = d_i^z + \min_{k \in A(i)} \{D_{k,j}\} \tag{2}$$

where $D^{z-}_{i,j}$ is the accumulative cost until node i and frame j in the dynamic programming (DP) table of dropping node and d_i^z is the node dropping cost of node i. A low dropping cost indicates that the corresponding GTG node is not in the video and should be omitted. This also allows dropping *extra nodes*, whenever there is no corresponding frame in the video.

GTG-Video Alignment (GTG2Vid). Algorithm 1 shows the steps of GTG2Vid. $C_{i,j}$ is the matching cost of the i-th node to the j-th frame. We let $C_{i,j}$ to be the predicted probability of step i for j-th frame using an action segmentation model. We train the action segmentation model with smoothing and cross-entropy loss in [22] using framewise labels. In addition, to ensure that our proposed method can drop frames and nodes based on the confidence of predictions, we propose a dynamic dropping cost d^x for frames and d^z for nodes of GTG as

$$\begin{aligned} d_j^x &= -\left(\frac{\sum_{i=1}^{2S} C_{i,j} \log C_{i,j}}{\log \frac{1}{2S}} + \eta\right), \\ d_i^z &= -\tau \left(1 - \max_{j=1,\ldots,T} C_{i,j}\right), \end{aligned} \tag{3}$$

where η and τ are hyper-parameters (larger η and τ lead to dropping more frames and nodes). Our frame dropping cost d^x leads to drop frames with unconfident predictions, indicating that the frames do not correspond to any GTG node and may contain errors. Meanwhile, our node dropping cost d^z allows dropping the nodes whose highest step probabilities over all frames are low. After obtaining the minimum cost path, we generate step predictions $\hat{\mathcal{Y}}$ from $\hat{\mathcal{Y}}^*$ by flagging the dropped frames as error frames and frames belonging to *extra nodes* as background. The complexity of our al-

Method	Quesadilla		Oatmeal		Pinwheel		Coffee		Tea	
	EgoPER									
	w-F1@0	EAcc	w-F1@0	EAcc	w-F1@0	EAcc	w-F1@0	EAcc	w-F1@0	EAcc
Naive Predictor	11.2	25.0	13.0	25.0	10.4	25.0	**18.1**	33.3	11.4	25.0
EgoPED (MV)	7.5	75.0	5.5	75.0	7.4	75.0	10.9	<u>100.0</u>	4.4	75.0
GTG2Vid (MV)	5.1	50.0	2.7	50.0	4.4	50.0	0.6	33.3	4.0	50.0
GTG2Vid (w/o n.f.)	16.3	75.0	16.4	75.0	**29.9**	<u>100.0</u>	1.7	33.3	15.2	75.0
GTG2Vid	**31.7**	**100.0**	**31.3**	**100.0**	17.8	**100.0**	4.5	**100.0**	**22.1**	**100.0**
	CaptainCook4D									
	Hot Chocolate		Sandwich		Burritos		Ramen		Raita	
	w-F1@0	EAcc	w-F1@0	EAcc	w-F1@0	EAcc	w-F1@0	EAcc	w-F1@0	EAcc
Naive Predictor	7.1	20.0	5.1	20.0	7.4	25.0	**7.4**	25.0	7.3	33.3
EgoPED (MV)	7.3	80.0	3.4	80.0	7.1	<u>75.0</u>	3.3	<u>75.0</u>	4.3	66.6
GTG2Vid (MV)	4.4	60.0	4.7	40.0	6.7	50.0	2.7	25.0	11.4	66.6
GTG2Vid (w/o n.f.)	2.8	40.0	6.2	60.0	6.1	25.0	4.6	50.0	11.2	66.6
GTG2Vid	**10.5**	**100.0**	**8.5**	**100.0**	**9.4**	**75.0**	4.4	**75.0**	**23.0**	**100.0**

Table 1. *ER* results on EgoPER and CaptainCook4D.

gorithm is $\mathcal{O}(SQT)$, same as G2Vid [18], where S, Q, and T denote the number of GTG nodes, separate and linearly-ordered threads of GTG, and frames, respectively.

4.2. Obtaining Error Features via LLM/VLM

Our next step is to obtain error features for *ER*. Since we are only given normal videos during training, we i) leverage LLMs to obtain error descriptions for each error type for every step, and ii) use VLMs to generate the textual features that are aligned with visual features. To begin, given a step i and error type m, we generate its error type description using LLMs. The prompt for LLMs combines a description for step i, error definition for error type m, and the number of different error type descriptions to produce. Notice that we directly set normal type description ($m = 0$) as the description of step i. Next, we generate the feature vector $x^e_{i,m}$ for the error description using the text encoder in VLMs, where $x^e \in \mathbb{R}^{S\times M\times D}$ is a tensor containing features of M error type descriptions for S steps. When LLMs are prompted to produce multiple error descriptions for an error type, we average all features to get a single vector. Meanwhile, we use the video encoder in VLMs to generate frame features for videos. In this way, we can ensure that the features of videos and error descriptions are aligned in the VLM feature space. See the supplementary materials for detailed prompts and generated error descriptions.

4.3. Error Recognition Module (ERM)

Our ERM classifies error types for error frames according to their joint similarities of frame, normal type, and error type features from Section 4.2 and obtain *ER* predictions $\hat{\mathcal{E}}$.

Predicting Steps for Error Frames. First, we obtain *TAS* predictions $\hat{\mathcal{Y}}$ by inferring GTG2Vid on a video. Next, we obtain another *TAS* predictions $\hat{\mathcal{Y}}^+ = (\hat{y}^+_1, \hat{y}^+_2, \ldots, \hat{y}^+_T)$, where $\hat{y}^+_t \in \{0, 1, \ldots, S\}$, by inferring GTG2Vid without using the frame and node dropping term, forcing GTG2Vid to map all frames to nodes. It aims to obtain the step predictions for the error frames in $\hat{\mathcal{Y}}$ to find the error type features of the corresponding steps.

Computing Joint Similarities for Error Recognition. Next, we compute framewise joint similarities p for *ER*, where $p_{m,j} \in [0, 1]$ and $p \in \mathbb{R}^{M\times T}$. For each frame j, given step prediction $\hat{y}_j$, $\hat{y}^+_j$, and feature x^v_j, x^e, we find the error type $\hat{m}_j$ and compute its joint similarity $p_{\hat{m}_j,j}$ as

$$\hat{m}_j = \begin{cases} 0, & \text{if } \hat{y}_j \neq -1 \\ \underset{m\in\{1,\ldots,M\}}{\text{argmax}} \; s(x^v_j, x^e, \hat{y}^+_j, m), & \text{otherwise} \end{cases} \tag{4}$$

$$p_{\hat{m}_j,j} = \begin{cases} 1, & \text{if } \hat{y}_j \neq -1 \\ s(x^v_j, x^e, \hat{y}^+_j, \hat{m}_j), & \text{otherwise} \end{cases} \tag{5}$$

where $s(\cdot,\cdot,\cdot,\cdot)$ is our score function that outputs the joint similarity of the frame. Our score function is formulated as

$$s(x^v_j, x^e, y, m) = \text{sigmoid}\left((x^e_{y,0} \odot x^e_{y,m})^T x^v_j\right) \tag{6}$$

where $\odot$ is element-wise multiplication and $x^e_{y,0}$ is the feature of step description (normal type) for step y. The intuition of using normal type feature is that error frames often contain multiple sub-steps, including both normal and error ones (e.g., an actor opens a tortilla bag, grabs a tortilla from it, and accidentally drops it). A single error type feature may be unable to effectively match the corresponding error frames, especially when the number of frames is large and contain multiple sub-steps which the error description does not describe. Therefore, our score function computes the correlation between $x^e_{y,0}, x^e_{y,m}$, and x^v_j. Thus, it jointly considers the similarities of error frames with respect to both normal and error type features. We validate its effectiveness for *ER* in Section 5. Finally, we apply a maximum filter on each error type of p to smoothen the predictions, followed by argmax to obtain the error type predictions $\hat{\mathcal{E}}$.

Method	EgoPER Quesadilla F1@.5	Acc	Edit	Oatmeal F1@.5	Acc	Edit	Pinwheel F1@.5	Acc	Edit	Coffee F1@.5	Acc	Edit	Tea F1@.5	Acc	Edit	All F1@.5	Acc	Edit
G2Vid	57.0	53.7	60.0	68.8	74.5	76.8	49.1	52.7	53.7	59.1	57.1	**75.0**	61.1	56.1	69.4	59.0	58.8	70.0
EgoPED	56.9	61.4	47.3	62.1	54.8	70.3	49.8	54.7	41.0	53.9	59.6	55.4	69.4	**64.5**	58.3	56.4	59.0	54.5
GTG2Vid	**68.1**	**64.5**	**69.4**	**83.3**	**78.5**	**88.6**	**61.3**	**58.4**	**60.9**	**63.9**	**62.5**	73.7	**71.2**	62.6	**76.9**	**69.6**	**65.3**	**73.9**

Method	CaptainCook4D Hot Chocolate F1@.5	Acc	Edit	Sandwich F1@.5	Acc	Edit	Burritos F1@.5	Acc	Edit	Ramen F1@.5	Acc	Edit	Raita F1@.5	Acc	Edit	All F1@.5	Acc	Edit
G2Vid	17.1	19.2	23.8	21.3	21.7	**39.8**	33.4	27.5	**42.9**	28.0	33.9	**50.5**	18.5	23.1	26.1	23.7	25.1	**36.6**
EgoPED	5.0	21.7	13.3	13.1	22.5	18.2	21.1	31.8	21.8	15.7	**37.3**	25.4	**22.6**	**31.3**	23.0	15.5	28.9	20.3
GTG2Vid	**20.8**	**43.5**	**27.8**	**27.2**	**38.5**	38.8	**36.6**	**34.0**	41.3	**31.3**	33.6	42.9	22.1	30.8	**26.1**	**27.6**	**36.1**	35.4

Table 2. *TAS* results on EgoPER and CaptainCook4D.

5. Experiments

5.1. Experimental Setup

Dataset. We evaluate our proposed method on EgoPER [40] and CaptainCook4D [66]. EgoPER consists of 5 tasks with 386 egocentric videos and contains 5 types of errors, including omission, addition, modification, slip, and correction. Captaincook4D has 24 tasks with 94.5 hours and contains 8 types of errors, including preparation, measurement, technique, timing, temperature, missing steps, ordering, and other. For EgoPER, we evaluate our method on all 5 tasks and error types with the same training and testing split in [40]. For CaptainCook4D, to have enough videos for training and testing, we select 5 of the tasks which have at least 5 normal and error videos: *Spiced Hot Chocolate* (*Hot Chocolate*), *Microwave Egg Sandwich* (*Sandwich*), *Breakfast Burritos* (*Burritos*), *Ramen*, and *Cucumber Raita* (*Raita*). We assign all normal videos to the training set and error videos to test set. To fit our setting for recognizing execution errors, we report the following error types in CaptainCook4D: preparation (Prep.), measurement (Mea.), technique (Tec.), timing (Time), and temperature (Temp). See the supplementary materials for breakdown analysis of EgoPER and CaptainCook4D and their detailed training and testing splits.

Evaluation Metrics. We report results on *ER*, *TAS*, and *ED*. In our experiments, we report the F1 score (F1@β) with an overlapping threshold β. We compute the F1 score by averaging (if not specified) over the F1 scores that are independently computed on the segments for each step or error type. For *ER*, we report Error Accuracy (EAcc) and weighted F1@0 score (w-F1@0). As *ER* is challenging under the setting of no error video during training, we set β to 0, meaning that a segment is true positive whenever any frame in the segment is correctly classified. EAcc is the percentage of error types that a model can recognize; that is, the error types with F1@0 score larger than 0. For w-F1@0, we first compute F1@0 on the error type segments, excluding normal ones ($m = 0$), to alleviate the effect of data imbalance (e.g., nearly 90% of the segments in EgoPER are normal). We then compute w-F1@0, where w-F1@0= $\frac{\text{F1@0}}{M-\text{EAcc}\times M+1}$. w-F1@0 penalizes the model that fails to recognize all error types. For *TAS*, we report frame-wise Accuracy (Acc), segment-wise Edit score (Edit) on normal steps, and F1@.5 score. For *ED*, we report F1@.5 score on all segments. We also report F1@.5 score on normal (N.) and error segments (E.) separately for alleviating data imbalance. For omission error detection, we use Omission Accuracy (O-Acc), predicting the accuracy for omission steps, and Omission Intersection over Union (O-IoU), as in [40].

Baselines. Given the lack of baselines for joint *TAS*, *ER*, and *ED*, we report the performance of EgoPED (MV), a modified version of EgoPED [40] that can additionally perform *ER*. EgoPED (MV) computes the cosine similarities between frames and error types using the error type features generated in Section 4.2. Then it assigns frames to the type with the highest similarity and uses the majority voting (MV) strategy in [40] over frames within a segment to decide the final error type for each segment. EgoPED (MV) detects error segments with step thresholds equal to mean similarities over videos in the validation set. For *ER* and *ED*, since both datasets have data imbalance, we report the performance of Naive Predictor as another baseline. Naive Predictor classifies frames into the same error type for *ER* and into normal frames for *ED*. Specifically, it classifies the frames into modification errors for EgoPER and preparation errors for CaptainCook4D. For *TAS*, we compare with G2Vid [18] to show how G2Vid performs with standard task graphs. To run G2Vid with our cost matrix, we remove all the background entries.

Implementation Details. We employ GPT-4o mini [1] as our LLM for generating error descriptions and VideoCLIP [94] pre-trained on Ego4D [29] as our VLM for extracting both visual and textual features for videos and error type descriptions. For each task in both EgoPER and CaptainCook4D, we generate at least 2 descriptions for each error type of each action. We set the step description as the description of normal type. We use the backbone in [51] as

Method	EgoPER																	
	Quesadilla			Oatmeal			Pinwheel			Coffee			Tea			All		
	N.	E.	F1@.5	N.	E.	F1@.5	N.	E.	F1@.5	N.	E.	F1@.5	N.	E.	F1@.5	N.	E.	F1@.5
Naive Predictor	14.4	0.0	7.2	26.1	0.0	13.1	23.4	0.0	11.7	**44.1**	0.0	**22.1**	4.8	0.0	2.4	22.6	0.0	11.3
EgoPED	30.1	**25.2**	27.7	29.3	12.9	21.1	14.6	**24.5**	19.6	9.5	**8.9**	9.2	36.3	34.0	35.1	24.0	21.1	22.5
GTG2Vid	**40.4**	22.6	**31.5**	**58.5**	**36.4**	**47.4**	**30.6**	16.0	**23.3**	33.1	4.0	18.6	**51.5**	**37.0**	**44.2**	**42.8**	**23.2**	**33.0**
	CaptainCook4D																	
	Hot Chocolate			Sandwich			Burritos			Ramen			Raita			All		
	N.	E.	F1@.5	N.	E.	F1@.5	N.	E.	F1@.5	N.	E.	F1@.5	N.	E.	F1@.5	N.	E.	F1@.5
Naive Predictor	8.2	0.0	4.1	6.5	0.0	3.2	3.8	0.0	1.9	4.1	0.0	2.0	3.1	0.0	1.6	5.1	0.0	2.6
EgoPED	8.1	2.4	5.2	16.3	6.8	11.5	16.3	8.2	12.2	12.4	**10.7**	11.6	14.1	9.5	11.8	13.4	7.5	10.5
GTG2Vid	**15.8**	**20.9**	**18.3**	**16.7**	**12.8**	**14.8**	**16.9**	**19.7**	**18.3**	**22.6**	8.2	**15.4**	**18.2**	**9.9**	**14.0**	**18.0**	**14.3**	**16.2**

Table 3. *ED* results of different methods on EgoPER and CaptainCook4D. N. and E. denote the F1@.5 score for normal and error segments.

our action segmentation model. We train the model using Adam optimizer with learning rate 0.0001 for 3000 iterations. For *ER*, if $\hat{y}_j = -1$ and $\hat{y}_j^+$ is a background class, we directly assign an addition error and preparation error to $\hat{m}_j$ for EgoPER and CaptainCook4D, respectively, and assign the corresponding similarity to 1. For *ED*, we produce *ED* predictions based on $\hat{\mathcal{E}}$ by classifying different error types into a single error type. In practice, we set $\tau = S$ where S is the number of nodes in GTG and $\eta = 0$.

5.2. Experimental Results

Error Recognition (*ER*). Table 1 compares *ER* performance between different methods on EgoPER and CaptainCook4D. Our proposed framework, GTG2Vid, demonstrates the best recognition ability than Naive Predictor and EgoPED (MV). Specifically, our proposed method achieves 31.7%, 31.3%, 17.8%, and 22.1% on w-F1@0 compared to 7.5%, 5.5%, 7.4%, and 4.4% by EgoPED (MV) for *quesadilla*, *oatmeal*, *pinwheel*, and *tea*. On the other hand, our proposed method outperforms EgoPED (MV) on w-F1@0 for all tasks of CaptainCook4D. In addition, our proposed method achieves overall higher EAcc than EgoPED (MV), indicating that our method can recognize more error types on both datasets. Notice that our performance on w-F1@0 for *coffee* is worse than EgoPED (MV) because EgoPED predicts most of the segments as errors, shown in Table 3. Please see supplementary materials for the F1@0 score of each type of error.

Next, we investigate the effectiveness of our ERM and our proposed score function (Eq. 6). By replacing ERM with MV, the performance drops significantly (more than 10% on w-F1@0 across all tasks of EgoPER). On the other hand, our method, GTG2Vid (w/o n.f.), without using normal type features suffers a drop on EAcc for most of the tasks, indicating that our proposed function for computing joint similarity has a better ability to recognize different error types. Furthermore, Table 4 shows the Top-1 and Top-2 accuracy of different scoring methods given the features of ground-truth error segments (average over frame features) and generated error type features. Our method gains a per-

Method	Quesadilla		Pinwheel		Tea	
	Top-1	Top-2	Top-1	Top-2	Top-1	Top-2
Uniform	25.0	50.0	25.0	50.0	25.0	50.0
Cos. Sim.	42.6	65.8	27.7	48.4	27.8	45.3
ERM	**54.6**	**81.5**	**38.4**	**60.3**	**34.3**	**53.7**

Table 4. Performance of feature matching methods on EgoPER. Top-1 and Top-2 denote the top-1 and top-2 accuracy over all error segments.

formance boost on Top-1 and Top-2 accuracy. Specifically, we obtain 10.7% improvement on Top-1 accuracy compared to cosine similarity (Cos. Sim.) for *pinwheel*, which verifies our claim in Section 4.3.

Temporal Action Segmentation (*TAS*). For *TAS* performance shown in Table 2, our method obtains a 13.2%, 6.3%, and 19.4% higher F1@.5 , Acc and Edit score than EgoPED on All of EgoPER. For CaptainCook4D, our method obtains a 12.1%, 7.2%, and 15.1% higher F1, Acc, and Edit score than EgoPED on All. Furthermore, our method have more robust *TAS* performance compared to EgoPED, which suffers a significant performance drop in *Hot Chocolate* and *Sandwich*. Notice that our Edit score on All is comparable to G2Vid since there are not many omission errors in test videos. Still, our method achieves a higher F1@.5 score than G2Vid, which cannot distinguish between background frames and errors.

Error Detection (*ED*). Table 3 shows the *ED* performance of different methods on EgoPER. Our method achieves 33.0% higher on F1@.5 score, compared to 22.5% by EgoPED for All in EgoPER. Although EgoPED can obtain a higher F1 score on error segments for *quesadilla*, *pinwheel*, and *coffee*, its detection ability for normal segments decreases significantly, specifically a 23.6% drop on F1@.5 score for *coffee* compared to our method. Therefore, compared to EgoPED, our method achieves a better balance between normal and error segments. For All in CaptainCook4D, GTG2Vid outperforms EgoPED, achieving 16.2% on F1@.5 compared to 10.5% by EgoPED.

Ablation of Node and Frame Dropping Term. We show ablation studies and investigate the effectiveness of our pro-

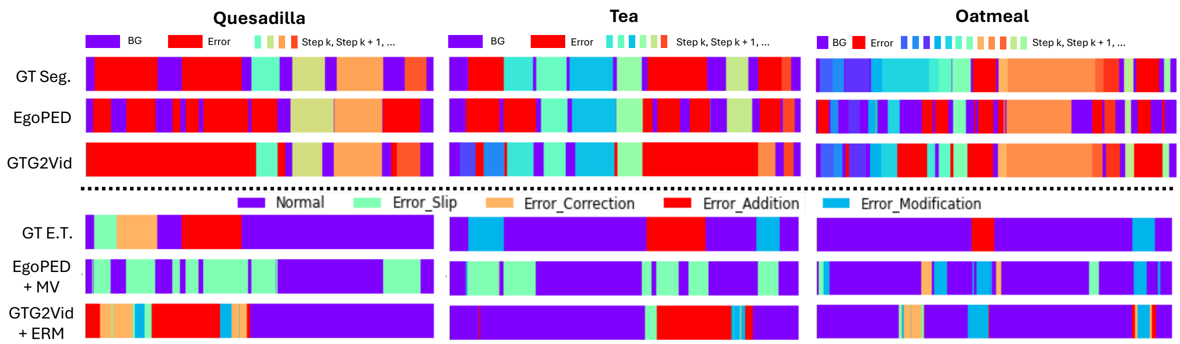

Figure 3. Qualitative results for *quesadilla*, *tea*, and *oatmeal* in EgoPER. The top part shows the results of *TAS* for ground-truths, EgoPED, and our method. The bottom part shows *ER* results for ground-truths, EgoPED, and our method. Seg. and E.T. denote action segmentation and error type, respectively.

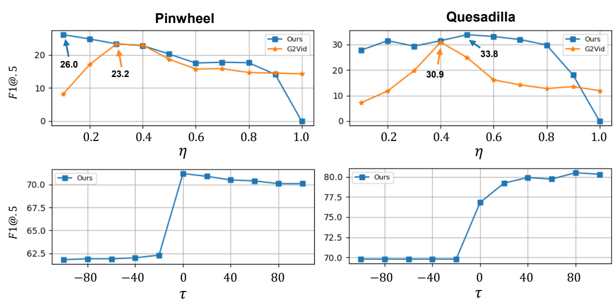

Figure 4. Ablation of our proposed node and frame dropping term.

posed node and frame dropping term in GTG2Vid. The two upper plots in Figure 4 show F1@.5 scores of our proposed frame dropping term (Ours) with different η and a naive dropping cost (G2Vid) [18] with different k top percentile of the values in the cost matrix, for *pinwheel* and *quesadilla*. Our method achieves 26.0% and 33.8% on F1@.5 score compared to 23.2% and 30.9% by G2Vid. On the other hand, the two bottom plots in Figure 4 show the F1 scores for *TAS* with different τ for our node dropping term. Smaller τ indicates GTG2Vid drops a fewer number of nodes. When τ is larger or equal to 0, GTG2Vid starts dropping nodes, improving *TAS* performance on videos with omission steps.

In addition, Figure 5 shows the performance of omission detection for different methods on EgoPER. First, since G2Vid is unable to drop nodes of a graph, it cannot predict any omitted steps, resulting in 0% on O-Acc and O-IoU. Second, although our method suffers a performance drop on O-IoU (5.7% lower than EgoPED on All), we gain a performance boost on O-Acc (10% higher than EgoPED on All), indicating that our method successfully drops nodes of a task graph according to the input videos and has a stronger capability for detecting omitted steps. To conclude, our proposed method uses 1) dynamic frame dropping term to precisely drop error segments and 2) dynamic node dropping term to drop unnecessary nodes of task graphs for better *TAS* and omission detection performance.

Qualitative Analysis. We visualize *ER* results in Fig. 3.

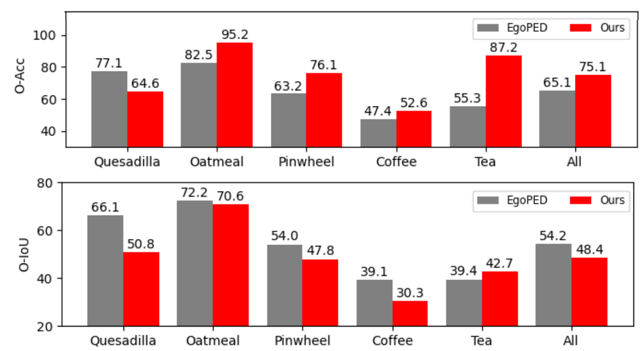

Figure 5. Omission detection results of different methods on EgoPER.

Our method can recognize different types of errors, e.g., addition errors (red) on *tea*, addition and correction (orange) errors on *quesadilla*, and modification errors (blue) on *oatmeal*. Our method obtains more accurate *TAS*, *ED*, and *ER* results for error videos on *quesadilla*, *tea* and *oatmeal* compared to EgoPED, thanks to dynamic frame and node dropping term in GTG2Vid and our proposed ERM. See supplementary materials for more qualitative visualization.

6. Conclusions

We studied joint *ER* and *TAS* in procedural task videos. Our proposed framework consists of 1) generalized task graphs to capture correct steps and background, 2) a GTG-Video Alignment algorithm to find the optimal alignment between nodes in GTG and video frames with dynamic dropping terms, 3) a procedure for generating error descriptions for each error type of every step using LLMs and the corresponding features using VLMs, and 4) an Error Recognition Module to effectively perform *ER* using joint similarities. Our experiments on two datasets showed that our proposed method can obtain promising results on *ER*, *ED*, and *TAS*.

Acknowledgement

This work was funded, in part, by ARPA-H (1AY2AX000062), DARPA PTG (HR00112220001), NSF (IIS-2115110), ONR (N000142512287) and ARO (W911NF2110276).

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 6

[2] Andra Acsintoae, Andrei Florescu, Mariana-Iuliana Georgescu, Tudor Mare, Paul Sumedrea, Radu Tudor Ionescu, Fahad Shahbaz Khan, , and Mubarak Shah. Ubnormal: New benchmark for supervised open-set video anomaly detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[3] Hyemin Ahn and Dongheui Lee. Refining action segmentation with hierarchical video representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16302–16310, 2021. 2

[4] J. B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien. Unsupervised learning from narrated instruction videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2

[5] Kumar Ashutosh, Santhosh Kumar Ramakrishnan, Triantafyllos Afouras, and Kristen Grauman. Video-mined task graphs for keystep recognition in instructional videos. *Neural Information Processing Systems*, 2023. 1, 2, 4

[6] Marcella Astrid, Muhammad Zaigham Zaheer, Jae-Yeong Lee, and Seung-Ik Lee. Learning not to reconstruct anomalies. *arXiv: 2110.09742*, 2021. 2

[7] Nadine Behrmann, S. Alireza Golestaneh, Zico Kolter, Juergen Gall, and Mehdi Noroozi. Unified fully and timestamp supervised temporal action segmentation via sequence to sequence translation. In *ECCV*, 2022. 2

[8] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021. 1

[9] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1

[10] Chien-Yi Chang, De-An Huang, Yanan Sui, Li Fei-Fei, and Juan Carlos Niebles. D3tw: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[11] Chien-Yi Chang, De-An Huang, Danfei Xu, Ehsan Adeli, Li Fei-Fei, and Juan Carlos Niebles. Procedure planning in instructional videos. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI*, 2020. 1

[12] Hanqiu Deng, Zhaoxiang Zhang, Shihao Zou, , and Xingyu Li. Bi-directional frame interpolation for unsupervised video anomaly detection. *IEEE Winter Conference on Applications of Computer Vision*, 2023. 2

[13] Guodong Ding, Fadime Sener, Shugao Ma, and Angela Yao. Every mistake counts in assembly. *arXiv: 2307.16453*, 2023. 1, 2

[14] Guodong Ding, Hans Golong, and Angela Yao. Coherent temporal synthesis for incremental action segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 2

[15] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[16] G. Donahue and E. Elhamifar. Learning to predict activity progress by self-supervised video alignment. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1

[17] Nikita Dvornik, Isma Hadji, Konstantinos G Derpanis, Animesh Garg, and Allan D Jepson. Drop-dtw: Aligning common signal between sequences while dropping outliers. In *NeurIPS*, 2021. 1, 3

[18] Nikita Dvornik, Isma Hadji, Hai Pham, Dhaivat Bhatt, Brais Martinez, Afsaneh Fazly, and Allan D Jepson. Flow graph to video grounding for weakly-supervised multi-step localization. In *European Conference on Computer Vision*, pages 319–335. Springer, 2022. 1, 2, 3, 4, 5, 6, 8

[19] Nikita Dvornik, Isma Hadji, Ran Zhang, Konstantinos Derpanis, Animesh Garg, Richard Wildes, and Allan Jepson. Stepformer: Self-supervised step discovery and localization in instructional videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2023. 1

[20] E. Elhamifar and D. Huynh. Self-supervised multi-task procedure learning from instructional videos. *European Conference on Computer Vision*, 2020. 2

[21] E. Elhamifar and Z. Naing. Unsupervised procedure learning via joint dynamic summarization. *International Conference on Computer Vision*, 2019. 2

[22] Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3575–3584, 2019. 1, 2, 4

[23] Mohsen Fayyaz and Jurgen Gall. Sct: Set constrained temporal transformer for set supervised action segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[24] C. Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[25] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 1

[26] Alessandro Flaborea, Guido Melendugno, Leonardo Pliniq, Luca Scofanoq, Edoardo Matteisq, Antonino Furnari, Giovanni Farinella, and Fabio Galasso. Prego: online mistake detection in procedural egocentric videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2

[27] Daniel Fried, Jean-Baptiste Alayrac, Phil Blunsom, Chris Dyer, Stephen Clark, and Aida Nematzadeh. Learning to segment actions from observation and narration. *Annual Meeting of the Association for Computational Linguistics*, 2020. 2

[28] Reza Ghoddoosian, Isht Dwivedi, Nakul Agarwal, and Behzad Dariush. Weakly-supervised action segmentation and unseen error detection in anomalous instructional videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10128–10138, 2023. 1, 2

[29] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh K. Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Z. Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Christian Fuegen, Abrham Gebreselasie, Cristina González, James M. Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jáchym Kolár, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz Puentes, Merey Ramazanova, Leda Sari, Kiran K. Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Yunyi Zhu, Pablo Arbeláez, David J. Crandall, Dima Damen, Giovanni Maria Farinella, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard A. Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. Ego4d: Around the world in 3,000 hours of egocentric video. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18973–18990, 2021. 6

[30] Yifei Huang, Yusuke Sugano, and Yoichi Sato. Improving action segmentation via graph-based temporal reasoning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[31] Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2322–2331, 2021. 2

[32] Mohaiminul Islam, Tushar Nagarajan, Huiyu Wang, Fu-Jen Chu, Kris Kitani, Gedas Bertasius, and Xitong Yang. Propose, assess, search: Harnessing llms for goal-oriented planning in instructional videos. *European Conference on Computer Vision*, 2024. 1

[33] Jaehyun Kim, Seongwook Yoon, Taehyeon Choi, and Sanghoon Sull. Unsupervised video anomaly detection based on similarity with predefined text descriptions. *Sensors*, 2023. 2

[34] H. Kuehne, J. Gall, and T. Serre. An end-to-end generative framework for video segmentation and recognition. *IEEE Winter Conference on Applications of Computer Vision*, 2016. 2

[35] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised learning of action classes with continuous temporal embedding. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[36] Sateesh Kumar, Sanjay Haresh, Awais Ahmed, Andrey Konin, M. Zeeshan Zia, and Quoc-Huy Tran. Unsupervised action segmentation by joint representation learning and online clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20174–20185, 2022. 2

[37] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2

[38] Jungbeom Lee, Jihun Yi, Chaehun Shin, and Sungroh Yoon. Bbam: Bounding box attribution map for weakly supervised semantic and instance segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 2

[39] Sangmin Lee, Hak Gu Kim, and Yong Man Ro. Bman: Bidirectional multi-scale aggregation networks for abnormal event detection. *IEEE Transactions on Image Processing*, 2019. 2

[40] S. Lee, Z. Lu, Z. Zhang, M. Hoai, and E. Elhamifar. Error detection in egocentric procedural task videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2, 3, 6

[41] Jun Li and Sinisa Todorovic. Set-constrained viterbi for set-supervised action segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[42] J. Li and S. Todorovic. Anchor-constrained viterbi for set-supervised action segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[43] J. Li, P. Lei, and S. Todorovic. Weakly supervised energy-based learning for action segmentation. *International Conference on Computer Vision*, 2019. 2

[44] M. Li, L. Chen, Y. Duarr, Z. Hu, J. Feng, J. Zhou, and J. Lu. Bridge-prompt: Towards ordinal action understanding in instructional videos. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2

[45] Shi-Jie Li, Yazan AbuFarha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. 1, 2

[46] 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 *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[47] Zhe Li, Yazan Abu Farha, and Jurgen Gall. Temporal action segmentation from timestamp supervision. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[48] Zeqian Li, Qirui Chen, Tengda Han, Ya Zhan, Yanfeng Wang, and Weidi Xie. Multi-sentence grounding for long-term instructional video. *European Conference on Computer Vision*, 2024. 1

[49] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 1

[50] Daochang Liu, Qiyue Li, AnhDung Dinh, Tingting Jiang, Mubarak Shah, and Chang Xu. Diffusion action segmentation. *arXiv preprint arXiv:2303.17959*, 2023. 2

[51] Daochang Liu, Qiyue Li, Anh-Dung Dinh, Tingting Jiang, Mubarak Shah, and Chang Xu. Diffusion action segmentation. 2023. 6

[52] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[53] Yang Liu, Jiayu Huo, Jingjing Peng, Rachel Sparks, Prokar Dasgupta, Alejandro Granados, and Sebastien Ourselin. Skit: a fast key information video transformer for online surgical phase recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21074–21084, 2023. 1

[54] Z. Lu and E. Elhamifar. Weakly-supervised action segmentation and alignment via transcript-aware union-of-subspaces learning. *International Conference on Computer Vision*, 2021. 2

[55] Z. Lu and E. Elhamifar. Set-supervised action learning in procedural task videos via pairwise order consistency. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[56] Z. Lu and E. Elhamifar. Fact: Frame-action cross-attention temporal modeling for efficient action segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2

[57] Z. Lu and E. Elhamifar. Multi-modal few-shot temporal action segmentation. *International Conference on Computer Vision*, 2025. 1

[58] Z. Lu, A. Iftekhar, G. Mittal, T. Meng, X. Wang, C. Zhao, R. Kukkala, E. Elhamifar, and M. Chen. Decafnet: Delegate and conquer for efficient temporal grounding in long videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2025. 1

[59] Antonino Furnari Luigi Seminara, Giovanni Maria Farinella. Differentiable task graph learning: Procedural activity representation and online mistake detection from egocentric videos. *Neural Information Processing Systems*, 2024. 1, 2

[60] K. Mangalam, H. Fan, Y. Li, C. Wu, B. Xiong, C. Feichtenhofer, and J. Malik. Reversible vision transformers. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[61] A. Miech, J-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman. End-to-end learning of visual representations from uncurated instructional videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1

[62] Fangzhou Mu, Sicheng Mo, and Yin Li. Snag: Scalable and accurate video grounding. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1

[63] Tushar Nagarajan and Lorenzo Torresani. Step differences in instructional video. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1

[64] 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. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1

[65] Zhanzhong Pang, Fadime Sener, Shrinivas Ramasubramanian, and Angela Yao. Long-tail temporal action segmentation with group-wise temporal logit adjustment. *European Conference on Computer Vision*, 2024. 2

[66] Rohith Peddi, Shivvrat Arya, Bharath Challa, Likhitha Pallapothula, Akshay Vyas, Bhavya Gouripeddi, Jikai Wang, Qifan Zhang, Vasundhara Komaragiri, Eric Ragan, Nicholas Ruozzi, Yu Xiang, and Vibhav Gogate. CaptainCook4D: A Dataset for Understanding Errors in Procedural Activities, 2024. 2, 6

[67] Didik Purwanto, Yie-Tarng Chen, and Wen-Hsien Fang. Dance with self-attention: A new look of conditional random fields on anomaly detection in videos. *IEEE International Conference on Computer Vision*, 2021. 2

[68] Yicheng Qian, Weixin Luo, Dongze Lian, Xu Tang, Peilin Zhao, and Shenghua Gao. Svip: Sequence verification for procedures in videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[69] Camillo Quattrocchi, Antonino Furnari, Daniele Mauro, Mario Giuffrida, and Giovanni Farinella. Synchronization is all you need: Exocentric-to-egocentric transfer for temporal action segmentation with unlabeled synchronized video pairs. *European Conference on Computer Vision*, 2024. 2

[70] Rahul Rahaman, Dipika Singhania, Alexandre Thiery, and Angela Yao. A generalized and robust framework for timestamp supervision in temporal action segmentation. In *Computer Vision–ECCV 2022: 17th European Conference*, 2022. 2

[71] A. Richard, H. Kuehne, and J. Gall. Action sets: Weakly supervised action segmentation without ordering constraints. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[72] A. Richard, H. Kuehne, A. Iqbal, and J. Gall. Neuralnetwork-viterbi: A framework for weakly supervised video learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[73] Nicolae-Cătălin Ristea, Neelu Madan, Radu Tudor Ionescu, Kamal Nasrollahi, Fahad Shahbaz Khan, Thomas B. Moeslund, and Mubarak Shah. Self-supervised predictive convolutional attentive block for anomaly detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[74] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 1, 2

[75] Saquib Sarfraz, Naila Murray, Vivek Sharma, Ali Diba, Luc Van Gool, and Rainer Stiefelhagen. Temporally-weighted

hierarchical clustering for unsupervised action segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[76] Fadime Sener, Dibyadip Chatterjee, Daniel Shelepov, Kun He, Dipika Singhania, Robert Wang, and Angela Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[77] Y. Shen and E. Elhamifar. Semi-weakly-supervised learning of complex actions from instructional task videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[78] Y. Shen and E. Elhamifar. Progress-aware online action segmentation for egocentric procedural task videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2, 4

[79] Y. Shen and E. Elhamifar. Understanding multi-task activities from single-task videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2025. 1

[80] Y. Shen, L. Wang, and E. Elhamifar. Learning to segment actions from visual and language instructions via differentiable weak sequence alignment. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 2

[81] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta. Asynchronous temporal fields for action recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[82] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. A multi-stream bi-directional recurrent neural network for finegrained action detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[83] Dipika Singhania, Rahul Rahaman, and Angela Yao. Coarse to fine multi-resolution temporal convolutional network. *CoRR*, abs/2105.10859, 2021. 2

[84] Dean Slawson. The case for vr-immersive and ai-adaptive soft skills training. *Training Industry Magazine*, 2018. 1

[85] Yaser Souri, Mohsen Fayyaz, Luca Minciullo, Gianpiero Francesca, and Juergen Gall. Fast Weakly Supervised Action Segmentation Using Mutual Consistency. *PAMI*, 2021. 1, 2

[86] Tomáš Souček, Dima Damen, Michael Wray, Ivan Laptev, and Josef Sivic. Genhowto: Learning to generate actions and state transformations from instructional videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1

[87] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[88] Kamalakar Vijay Thakare, Yash Raghuwanshi, Debi Prosad Dogra, Heeseung Choi, and Ig-Jae Kim. Dyannet: A scene dynamicity guided self-trained video anomaly detection network. *IEEE Winter Conference on Applications of Computer Vision*, 2023. 2

[89] Quoc-Huy Tran, Ahmed Mehmood, Muhammad Ahmed, Muhammad Naufil, Anas Zafar, Andrey Konin, and Zeeshan Zia. Permutation-aware activity segmentation via unsupervised frame-to-segment alignment. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6426–6436, 2024. 2

[90] Mengmeng Wang, Jiazheng Xing, and Yong Liu. Actionclip: A new paradigm for video action recognition. *CoRR*, 2021. 1

[91] Xin Wang, Taein Kwon, Mahdi Rad1 Bowen Pan, Ishani Chakraborty, and Sean Andrist. Holoassist: an egocentric human interaction dataset for interactive ai assistants in the real world. *IEEE International Conference on Computer Vision*, 2023. 1, 2

[92] Zhe Wang, Hao Chen, Xinyu Li, Chunhui Liu, Yuanjun Xiong, Joseph Tighe, and Charless Fowlkes. Sscap: Self-supervised co-occurrence action parsing for unsupervised temporal action segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1819–1828, 2022. 2

[93] Jhih-Ciang Wu, He-Yen Hsieh, Ding-Jie Chen, Chiou-Shann Fuh, and Tyng-Luh Liu. Self-supervised sparse representation for video anomaly detection. *European Conference on Computer Vision*, 2022. 2

[94] Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. VideoCLIP: Contrastive pre-training for zero-shot video-text understanding. 2021. 6

[95] Ming Xu and Stephen Gould. Temporally consistent unbalanced optimal transport for unsupervised action segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 2

[96] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 2018. 1, 2

[97] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. Asformer: Transformer for action segmentation. In *The British Machine Vision Conference (BMVC)*, 2021. 1, 2

[98] Guang Yu, Siqi Wang, Zhiping Cai, Xinwang Liu, Chuanfu Xu, and Chengkun Wu. Deep anomaly discovery from unlabeled videos via normality advantage and self-paced refinement. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[99] Muhammad Zaigham Zaheer, Arif Mahmood, Marcella Astrid, and Seung-Ik Lee. Claws: Clustering assisted weakly supervised learning with normalcy suppression for anomalous event detection. *European Conference on Computer Vision*, 2020.

[100] M Zaigham Zaheer, Arif Mahmood, M Haris Khan, Mattia Segu, Fisher Yu, and Seung-Ik Lee. Generative cooperative learning for unsupervised video anomaly detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[101] P. Zameni, Y. Shen, and E. Elhamifar. Moscato: Predicting multiple object state change through actions. *International Conference on Computer Vision*, 2025. 1

[102] Luca Zanella, Willi Menapace, Massimiliano Mancini, Yiming Wang, and Elisa Ricci. Harnessing large language

models for training-free video anomaly detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 2

[103] Ali Zare, Yulei Niu, Hammad Ayyubi, and Shih-Fu Chang. Rap: Retrieval-augmented planner for adaptive procedure planning in instructional videos. *European Conference on Computer Vision*, 2024. 1

[104] Junbin Zhang, Pei-Hsuan Tsai, and Meng-Hsun Tsai. Semantic2graph: Graph-based multi-modal feature fusion for action segmentation in videos, 2022. 1, 2

[105] Bin Zhao, Li Fei-Fei, , and Eric P. Xing. Online detection of unusual events in videos via dynamic sparse coding. *IEEE Conference on Computer Vision and Pattern Recognition*, 2011. 2

[106] Honglu Zhou, Roberto Martín-Martín, Mubbasir Kapadia, Silvio Savarese, and Juan Carlos Niebles. Procedure-aware pretraining for instructional video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10727–10738, 2023. 1, 2, 4

[107] Joey Tianyi Zhou, Jiawei Du, Hongyuan Zhu, Xi Peng, Yong Liu, , and Rick Siow Mong Goh. Anomalynet: An anomaly detection network for video surveillance. *IEEE Transactions on Information Forensics and Security*, 2019. 2

[108] Yuansheng Zhu, Wentao Bao, , and Qi Yu. Towards open set video anomaly detection. *European Conference on Computer Vision*, 2022. 2

[109] D. Zhukov, J. B. Alayrac, R. G. Cinbis, D. Fouhey, I. Laptev, and J. Sivic. Cross-task weakly supervised learning from instructional videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2