I-Do, You-Learn: Techniques for Unsupervised Procedure Learning using Egocentric Videos

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in ***Computer Science and Engineering*** *by Research*

by

Siddhant Bansal
2019900091
`siddhant.bansal@research.iiit.ac.in`

International Institute of Information Technology
Hyderabad - 500 032, INDIA
February 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled **"I-Do, You-Learn: Techniques for Unsupervised Procedure Learning using Egocentric Videos"** by Siddhant Bansal, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Prof. C V Jawahar
Date	Adviser: Prof. Chetan Arora

To Sanjay, Seema, and Astha Bansal

Acknowledgments

Wow! What a fantastic journey this has been. A lot has changed from dropping a cold email to Professor Jawahar to writing this thesis. I began with minimal knowledge of how to do research and ended up publishing at premier conferences. All of this would not have been possible without the guidance of excellent mentors I have had throughout. Firstly, I thank Professor C.V. Jawahar for being an amazing guide throughout the degree. I am grateful for all the opportunities he provided and advised me through various problems I faced during the journey. Not only did he give numerous lessons on multiple aspects of research, but he also taught about navigating different aspects of life. Secondly, I would love to thank Professor Chetan Arora. Though my interaction with Professor Arora was majorly virtual, I never felt his physical absence. He was always available to answer all the questions/concerns I had regarding the projects. I am grateful for all our interactions; he made me feel comfortable having personal and project-related interactions. Finally, I would like to thank Dr. Praveen Krishnan. I worked with him in my initial years at CVIT and learned many things from him! From how to organise a project and prepare for weekly updates to keeping track of experiments and interacting with professors and seniors, he taught me a lot. I am grateful that I got to work with him on my initial projects. This allowed me to work independently on my later projects. I could not have asked for a better set of mentors for my master's. I cherish my interactions with my mentors and remember all the lessons I learned from them.

Before I move further, I would love to go back and acknowledge the person because of who I am where I am, Dr. Rajesh Thakker. He was the Head of the Department at Vishwakarma Government Engineering College, where I did my bachelor's. I worked with him on the Automatic Garbage Collection and Detection project for a while. There he encouraged and supported me in exploring various aspects of computer vision. I still remember the day when he took me to the library and gave me a book on neural networks! I will always be grateful for his support and faith in me. Thank you for being by my side! Dr. Thakker even introduced me to Prof. Shanmuganathan Raman, who further helped me expand my horizon on computer vision and introduced me to various topics. Working in his lab and interacting with the students there further motivated me to pursue research!

"*I cannot even imagine where I would be today were it not for that handful of friends who have given me a heart full of joy. Let's face it, friends make life a lot more fun.*" – Charles R. Swindoll.

I agree with Swindoll. This journey would have been mundane without friends. I made many friends at CVIT who were there with me through the different phases of the degree. I appreciate them all and cherish all the memories we have together. I remember writing my first paper and being confused about various aspects of it. At that time, Rudrabha came and helped me with ideas on adding diagrams and improving other aspects of writing. I appreciate all the discussions (gossip, too :P) we had about various aspects of paper writing, subject selection, and countless other things. As COVID started, it took a lot of work to communicate and collaborate with others on various subjects. During that time, Madhav and I took on similar subjects and discussed multiple doubts. I remember him sitting on a call for seven hours on his birthday! Furthermore, I am grateful to have Prajwal and Sindhu as my seniors, who always answer my endless questions! Speaking of seniors, Avijit has been a fantastic friend and mentor. From discussions on various critical aspects of life to planning trips together (which are yet to happen xD), he has been a constant support. Finally, thanks to Seshadri, Rupak, Shubham, Ravi, George, Soumya, Prafful, Zeeshan, Raghava, and Pranav for being amazing company during various portions of the day. All the fun, discussions, and gossip we had kept me going during the degree.

CVIT is incomplete without Aradhana, Rohita, Ram, Varun, Ann, and Sony! I appreciate Aradhana's help and patience throughout the journey. She has been an invisible pillar of support, from helping with managing the convoluted ECCV trip to giving me a box of rice to put my wet phone into. Rohita, I wonder about the amount of patience she has. She used to handle multiple requests from all of us, not to mention the number of questions I used to ask her. I appreciate all her help while planning the trip to CVPR and sharing with me when Professor Jawahar comes out of his office :P. I can not thank Ram and Varun enough for their help and support through the Ego4D project. They have entertained numerous requests from me and made the project a success for CVIT. Finally, I appreciate the help and support Ann and Sony provided to us. I appreciate them waiting for me to respond to their emails and provide helpful feedback.

"*If you really want to make a friend, go to someone's house and eat with him... The people who give you their food give you their heart.*" – Cesar Chavez

Yuktahaar has been that "house" for me. I know many people at IIIT hate it, but I have countless memories there. More than food, the hall of Yuktahaar has provided me with endless memories, amazing friends, and many interesting conversations. Yuktahaar is where CVIT, CCNSB, CogSci, and Earthquake Engineering collaborate. I remember meeting Nidhi there and making fun of her :P. Learning earthquake engineering from Bharat and Supriya. Understanding the notorious malaria parasite and how our hands have enzymes from Gayathri. Not to mention, teasing Annapoorni on how slow she eats (and walks) was fun. Talking to Chirag and getting an undergrad's perspective of the campus was interesting. Sharing what is going on in the heart over the dinner and listening to how other's day was, is possible only in Yuktahaar.

"In hostel, nights turned into morning, with friends that turn into family" – Anonymous

Cannot disagree with this quote! The time I have had with my friends in the hostel is priceless. I would love to thank Dhawal, Rodo, Nayan, Prateek, Bhoomendra, and Dhruv for making the hostel feel like home. These people have always been there with me, from playing Catan till five in the morning to helping me relax after a paper deadline. I cannot imagine meeting deadlines without the mental comfort of being around fun people.

Towards the end, I would love to thank my friends outside IIIT who have helped me during various aspects of the journey. Mukul Khanna has been a fantastic friend who listens to me rant about multiple stuff and helps me proofread my paper. Similarly, Jehlum Vitasta Pandit has been a constant support and was there to listen to me when I had to take difficult decisions. Not to mention, she helped with annotating the dataset proposed in the thesis!

Finally, I would love to thank my amazing family, who have always believed in me. My parents have been a constant support throughout the journey. From making a Twitter account to follow various computer vision trends and learning about the paper reviewing process to discussing every meeting I had in the past three years, my dad has always been there for me. I do not have words to thank him enough for the support he provides. On the other hand, my mom ensured I was taking care of myself and taking appropriate breaks while working. Being able to call her anytime and sharing how I felt kept me going. It goes without saying that their support is why I could finish my master's. My sister, though jealous of me travelling to various parts of the world, has always appreciated my work. After LACES, she is the one who proofreads my papers. Also, all the gossip that she provides brightens the day! Difficult to convey using words how her calls have helped me achieve this milestone.

As said by Drake, "*Sometimes it's the journey that teaches you a lot about your destination.*" My master's journey has changed me and made me a different, better person.

Abstract

Consider an autonomous agent capable of observing multiple humans making a pizza and making one the next time! Motivated to contribute towards creating systems capable of understanding and reasoning instructions at the human level, in this thesis, we tackle procedure learning. Procedure learning involves identifying the key-steps and determining their logical order to perform a task.

The first portion of this thesis focuses on the datasets curated for procedure learning. Existing datasets commonly consist of third-person videos for learning the procedure, making the manipulated object small in appearance and often occluded by the actor, leading to significant errors. In contrast, we observe that videos obtained from first-person (egocentric) wearable cameras provide an unobstructed and clear view of the action. To this end, for studying procedure learning from egocentric videos, we propose the EgoProceL dataset. However, procedure learning from egocentric videos is challenging because the camera view undergoes extreme changes due to the wearer's head motion and introduces unrelated frames. Due to this, current state-of-the-art methods' assumptions that the actions occur at approximately the same time and are of the same duration do not hold. Instead, we propose to use the signal provided by the temporal correspondences between key-steps across videos. To this end, we present a novel self-supervised Correspond and Cut (CnC) framework that identifies and utilizes the temporal correspondences between the key-steps across multiple videos to learn the procedure. We perform experiments on the benchmark ProceL and CrossTask datasets and achieve state-of-the-art results.

In the second portion of the thesis, we look at various approaches to generate the signal for learning the embedding space. Existing approaches use only one or a couple of videos for this purpose. However, we argue that it makes key-steps discovery challenging as the algorithms lack an inter-videos perspective. To this end, we propose an unsupervised Graph-based Procedure Learning (GPL) framework. GPL consists of the novel UnityGraph that represents all the videos of a task as a graph to obtain both intra-video and inter-videos context. Further, to obtain similar embeddings for the same key-steps, the embeddings of UnityGraph are updated in an unsupervised manner using the Node2Vec algorithm. Finally, to identify the key-steps, we cluster the embeddings using KMeans. We test GPL on benchmark ProceL, CrossTask, and EgoProceL datasets and achieve an average improvement of 2% on third-person datasets and 3.6% on EgoProceL over the state-of-the-art.

We hope this work motivates future research on procedure learning from egocentric videos. Furthermore, the unsupervised approaches proposed in the thesis will help create scalable systems and drive future research toward creative solutions.

Contents

List of Figures

List of Tables

Chapter 1

Procedure Learning: Motivation, Challenges, and Prior Attempts

From fabricating a laptop on an assembly line to preparing a pizza, we follow a series of steps to accomplish the task. Also, each task requires specific domain knowledge to carry it out. Creating a robotic agent that augments humans in performing such tasks requires years of research and development. Furthermore, tuning the same agent for a different task is challenging. To overcome these challenges and create scalable systems, in this thesis, we aim to devise frameworks capable of "watching" human subjects perform a task and learn from it.

Creating a framework capable of learning from a few human demonstrations has various advantages. For example,

1. Such frameworks are **scalable** and capable of learning multiple tasks (*e.g.*, the same framework can learn to assemble a PC and make a brownie!).
2. They are **efficient**. Instead of requiring years of research, such systems can be trained in hours.
3. Owing to the scalability and efficiency of such systems, they can be deployed on various devices, opening up a world of exciting applications.

Furthermore, such frameworks would be helpful for various applications, like,

1. **Automated Systems:** Such systems can enable robotic systems to autonomously learn the steps for performing the task by observing the task being performed. Once the automated system learns the steps, the next time, it can do the task without human assistance.
2. **Monitoring Procedures:** Consider a system trained to know the key-steps for performing a task; if a new person does the same task again, the system will identify if the person misses a step or does a step differently.
3. **Guidance Systems:** A system trained to know the key-steps for performing a task can identify the current step and show the next possible step for performing the task.

To create frameworks capable of learning from a few human demonstrations and high applicability, in this thesis, we focus on **procedure learning**.

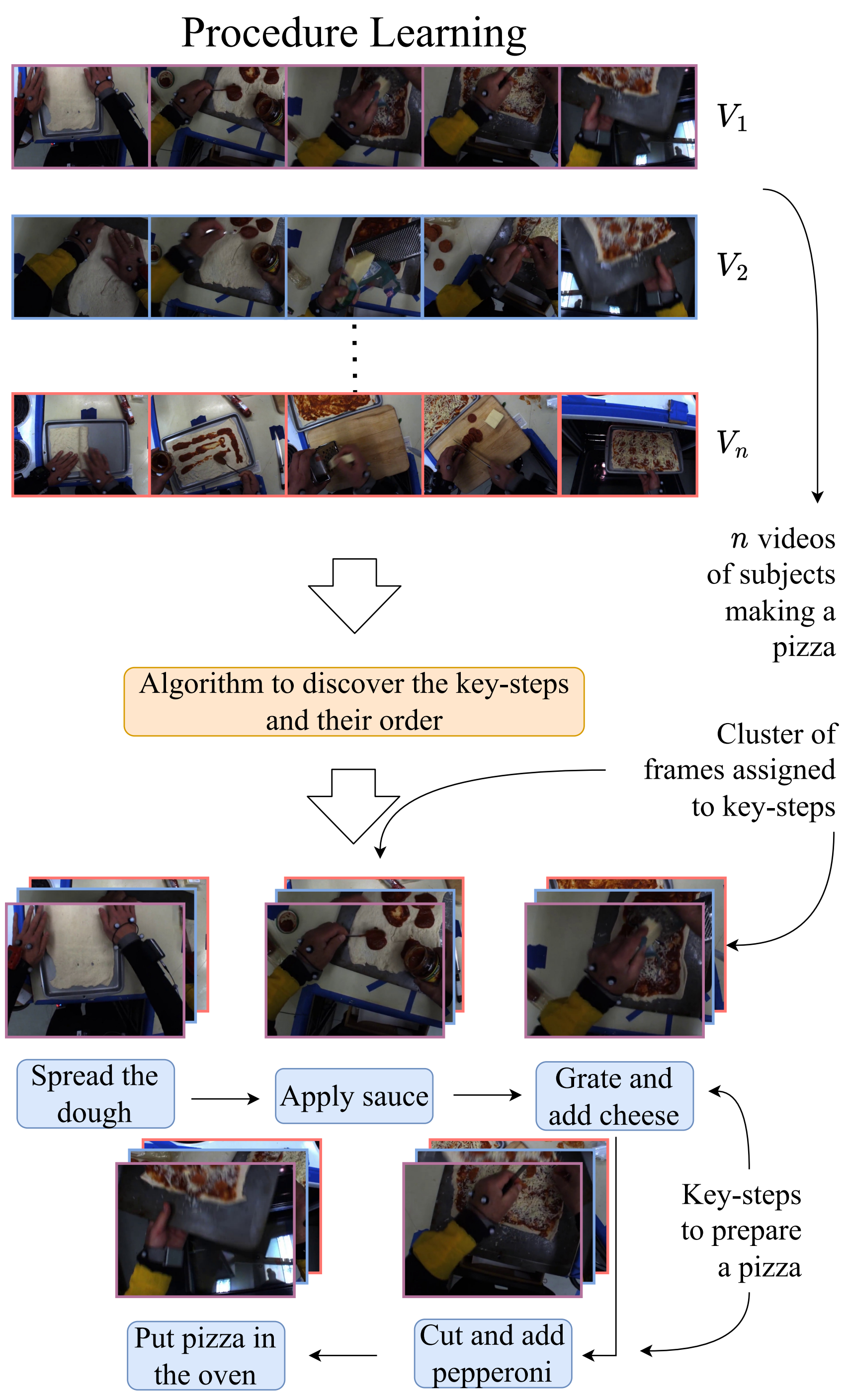

Procedure Learning
V_1
V_2
V_n
n videos of subjects making a pizza
Algorithm to discover the key-steps and their order
Cluster of frames assigned to key-steps
Spread the dough
Apply sauce
Grate and add cheese
Key-steps to prepare a pizza
Cut and add pepperoni
Put pizza in the oven

Figure 1.1 Procedure Learning involves identifying the key-step and their order from multiple videos of the same task. Here, in the input, we have n videos $(V_1, V_2, \ldots, V_n)$ of subjects preparing a sandwich. The output is **(a)** frames (from all the videos) assigned to their respective key-step and **(b)** order of the key-steps.

1.1 What is Procedure Learning?

As shown in Figure 1.1, given a set of instructional videos for the same task, procedure learning [19, 20, 68] broadly consists of two steps,

1. assigning all the frames to the K key-steps (including the background), and
2. discovering the logical ordering of the key-steps required to perform the task.

Formally, we consider n untrimmed videos of the same task, denoted by $V = \{V_i : i \in \mathbb{N}, 1 \leq i \leq n\}$. Each of the n videos can have a different number of frames. A video V_k with m frames is denoted as $V_k = \{f_k^1, f_k^2, \ldots, f_k^m\}$. The goal is to define a framework $f(\theta)$ (with learnable parameters θ) that takes in the n videos and classifies all the $n \times m$ (m can vary with video) into K key-step clusters. Also, we aim to determine the order of K clusters.

Note that we aim to solve the task in an unsupervised/self-supervised manner. Due to this, we do not utilise the labels in our frameworks. The labels are only used during evaluation (refer to Section 3.3.1).

1.1.1 Why is it Difficult to Learn Procedures?

In this thesis, we aim to create systems capable of understanding and reasoning instructions at the human level. For example, as shown in Figure 1.1, the instructions to prepare a pizza are at human level (referred to as key-steps in the thesis). In contrast, computer vision tasks, like action segmentation, approach at a finer level of actions [14, 35, 64, 73] (Section 1.1.2). However, as we show in qualitative analysis in Chapter 3 and Chapter 4, segmenting at human level is challenging. This is mainly because, along with the actions required to perform a task, there are actions that are not the key-steps but may supplement the key-step (for example, taking out butter from the shelf).

Furthermore, due to the definition of the key-steps, a major portion of procedure learning datasets consists of background actions [3, 20, 86]. This not only makes learning procedures challenging but also makes evaluating the learned procedure problematic. For example, when evaluating using F1-Score [19, 20, 44, 68, 77], a model that assigns all the frames to the background, will score high. To fix the issue, in this thesis, we propose an updated evaluation protocol described in Section 3.3.1.

1.1.2 How is Procedure Learning Different from other Tasks?

As shown in Figure 1.1, procedure learning deals with multiple videos of a task. In contrast, action-based tasks deal with a single video [14, 35, 64, 73], hence losing the capability to determine repetitive key-steps across the videos. Secondly, these tasks do not consider the order of the individual events, which is often crucial for identifying key-steps, and/or procedures/recipes. For example, action-based tasks do not capture the difference in the order of key-steps.

Procedure Segmentation [37], on the other hand, deals with dividing a single video into procedure-level segments. Instead, procedure learning deals with generating segments across multiple videos.

Furthermore, similar to action-based tasks, procedure segmentation does not deal with the ordering of the key-steps. Also, as procedure learning deals with localising the key-steps, it differs from the video alignment task. Therefore, considering the utility of procedure learning and its distinctness from existing tasks, we aim to solve it.

1.1.3 Contributions

As mentioned in the previous sections, this thesis deals with procedure learning and aims to solve its challenges. To this end, the following are our core contributions:

1. To facilitate procedure learning from egocentric videos, we create the EgoProceL dataset. The dataset consists of 62 hours of egocentric videos captured by 130 subjects performing 16 tasks.

2. We propose two novel methods to solve procedure learning in an unsupervised manner. The first method deals with utilizing video alignment approaches whereas, the second method exploit graphs for procedure learning.

3. In the first work, we propose the Correspond and Cut (CnC) framework, which utilizes the proposed TC3I loss and PCM to identify the key-steps and their ordering required to perform a task. Furthermore, we investigate the usefulness of egocentric videos over third-person videos for procedure learning. We observe an average improvement of 2.7% in the F1-Score when using egocentric videos instead of third-person videos.

4. In the second work, we propose the Graph-based Procedure Learning (GPL) framework. Contrary to existing graph-based frameworks, GPL does not require node or edge annotations, enabling unsupervised procedure learning. Furthermore, we create a novel graph representation for arbitrary number of videos: UnityGraph. UnityGraph captures **(a)** temporal relationships in the same video and **(b)** semantic relationships across the videos. Also, to identify the background frames, we propose to detect hand-object interactions in egocentric videos. This leads to an improvement of 1.1% in the F1-Score on EgoProceL.

5. To evaluate both the proposed approaches, we perform experiments and ablation on two third-person datasets (ProceL [20] and CrossTask [86]) and the proposed EgoProceL dataset.

1.2 Previous Attempts to Learn Procedures

In this section, we have a look at various prior attempts at procedure learning, representation learning, and key-step ordering.

1.2.1 Representation Learning for Procedure Learning

Previous works on procedure learning have developed methods to learn frame-level features [19, 20, 44, 77]. Kukleva *et al.* [44] learn the representation space by using relative timestamps of the frames. On the other hand, Vidal *et al.* [77] predict the future frame and its timestamps. Elhamifar *et al.* [19] learn and employ attention features for individual frames. Bansal *et al.* [3] exploit temporal correspondences across the videos to generate the signal and learn frame-level embeddings. However, these methods fall short in modelling either temporal or spatial relationships.

1.2.2 Multimodal Procedure Learning

Another class of methods works with multi-modal data, like narrated text and videos [2, 11, 16, 24, 54, 66, 68, 84, 87]. These works use Automatic Speech Recognition (ASR) to obtain the text, which is not perfect. Due to this, the output needs to be manually cleaned, which is not scalable. Additionally, such methods assume an alignment between the text and videos [2, 54, 84], which might not be accurate for most cases [19, 20]. Instead, we use only the visual modality as an input to the framework. Due to this, we eliminate the need to obtain narrations that might be inaccurate and make our framework scalable.

1.2.3 Self-Supervised Representation Learning

Learning a representation space without annotations saves substantial time and energy when creating deep learning solutions. Motivated by this, recent works explore various pretext tasks to generate supervision signals for training deep learning architectures [7, 31, 74, 75, 80]. A few pretext tasks for learning image representations include image colourization [45, 46], object counting [52, 59], solving jigsaw puzzles [6, 39], predicting image rotations [22, 41], and reconstructing input images [32] from noise [78]. Pretext tasks for learning video representations include predicting future frames [1, 13, 29, 38, 72, 79], using temporal order and coherence as labels [23, 47, 56, 82, 83] and predicting the arrow of time [81].

Video representation learning methods mentioned above employ a single video. However, we want to identify similar key-steps in multiple videos for procedure learning.

1.2.4 Learning Key-step Ordering

A majority of the previous works do not capture different key-step ordering to perform the same task. They either assume a strict ordering [20, 44, 77] or do not predict the order [19, 68]. However, we observe that subjects perform the same task in multiple ways, motivating us to capture different ways to accomplish the task.

1.3 Organization of the Thesis

The rest of the thesis is organised as follows:

1. In Chapter 2, we summarise existing procedure learning datasets and discuss their shortcomings. Furthermore, we propose the EgoProceL dataset consisting of egocentric videos for procedure learning.

2. In Chapter 3, we demonstrate the existence of correspondences across the videos and outline the proposed Correspond and Cut (CnC) framework to exploit them for procedure learning. Furthermore, we evaluate CnC on existing third-person and proposed EgoProceL dataset.

3. In Chapter 4, we explore the utility of graphs for procedure learning. We propose the Graph-based Procedure Learning (GPL) framework for procedure learning and evaluate it on third- and first-person datasets.

Chapter 2

The EgoProceL Dataset: Egocentric Videos for Procedure Learning

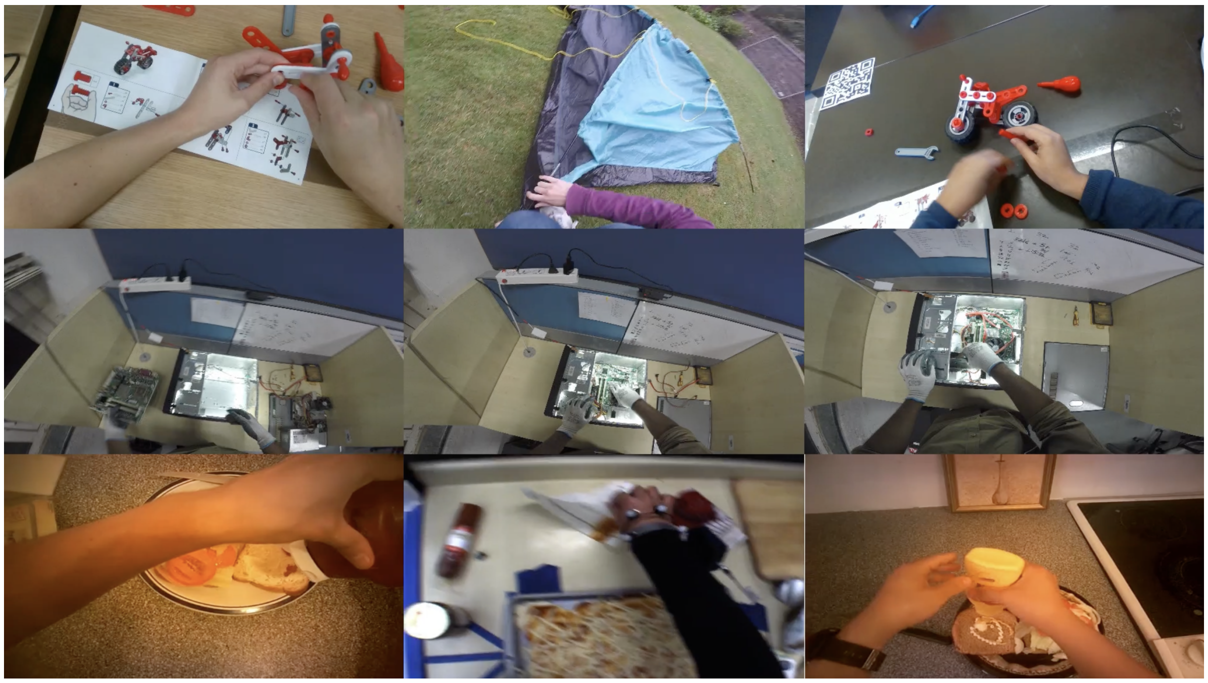


Figure 2.1 EgoProceL is a large-scale dataset for procedure learning. It consists of **62 hours** of egocentric videos recorded by **130 subjects** performing **16 tasks** for procedure learning. EgoProceL contains videos and key-step annotations for multiple tasks from CMU-MMAC [12], EGTEA Gaze+ [51], and individual tasks like toy-bike assembly [62], tent assembly [36], PC assembly, and PC disassembly.

In this chapter, we first have a look at existing third-person procedure learning datasets and discuss their shortcomings (Section 2.1). Eventually, in Section 2.2, we propose the EgoProceL dataset (Figure 2.1) to overcome the shortcomings.

2.1 Existing Datasets for Procedure Learning

In Section 2.1.1, we first summarise significant procedure learning datasets. Later, in Section 2.1.2, we highlight the shortcomings of existing datasets.

2.1.1 Third-Person Procedure Learning Datasets

As discussed in Section 1.2, a variety of attempts have been made for procedure learning. A majority of the works propose and utilise third-person video datasets for the task. The most significant datasets are:

1. **ProceL [20]**: It consists of 47.3 hours of videos from 12 diverse tasks. The tasks range from tying a tie and assembling clarinet to setting up a Chromecast. Each task has 60 videos collected from YouTube and the grammar of key-steps is created by the annotators.

2. **CrossTask [86]**: It contains 375 hours of videos collected from YouTube for 83 tasks. The dataset is further divided into two parts, 18 primary and 65 related tasks. Videos for the primary tasks have been acquired manually and contain the temporal step annotations necessary for evaluating procedure learning models. The primary tasks contain tasks ranging from cooking to auto repair to building floating shelves. On the other hand, videos for related tasks are gathered automatically and do not contain annotations. Due to this, unless otherwise mentioned, we refer to the primary dataset as CrossTask.

3. **Breakfast [42]**: This "in-the-wild" dataset consists of 77 hours of videos recorded by 52 unique participants in 18 different kitchens. The dataset provides coarse-level annotations for all the tasks. It consists of various cooking activities like preparation of coffee, orange juice, chocolate milk, *etc*.

4. **Inria [2]**: It consists of approximately 5 hours of videos of 5 tasks collected from YouTube. The tasks range from changing a tire to re-potting a plant. YouTube's Automatic Speech Recognition (ASR) is used to obtain transcripts for the videos. To facilitate evaluation, the dataset consists of temporal annotations of the main steps required to perform a task.

2.1.2 Issues with Third-person Datasets and How to Overcome them

As discussed in Section 2.1.1, existing instructional videos datasets [2, 20, 37, 42, 55, 73, 85, 86] majorly consist of third-person videos. Here, as shown in Figure 2.2, the camera is kept far from the expert to avoid interference in the actual task. Due to this, the manipulated objects are typically small or sometimes invisible. Additionally, third-person videos can be captured from various positions, leading to wide variations in the camera viewpoints for the same task [12]. Further, as shown in Figure 2.2, most datasets comprise videos scraped from the internet (YouTube) [2,20,37,55,73,86], which are noisy and

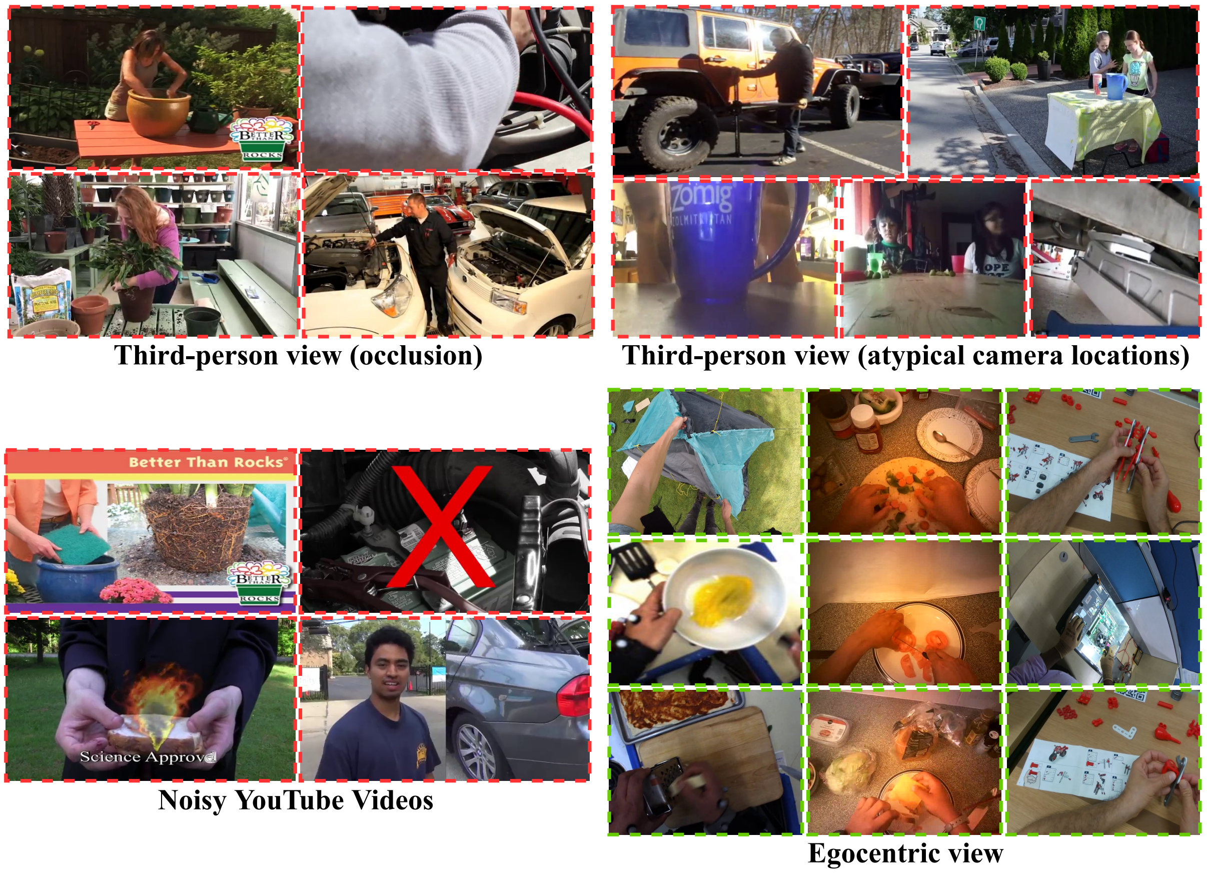

Third-person view (occlusion)
Third-person view (atypical camera locations)
Better Than Rocks
Science Approved
Noisy YouTube Videos
Egocentric view

Figure 2.2 Issues with standard datasets for procedure learning. Existing datasets [2, 20, 42, 55, 73, 85, 86] majorly consist of third-person videos. They contain issues like occlusion and atypical camera locations that make them ill-suited for procedure learning. Additionally, the datasets rely on noisy videos from YouTube [2, 20, 55, 73, 86]. In contrast, we propose to use egocentric videos that overcome the issues posed by third-person videos.

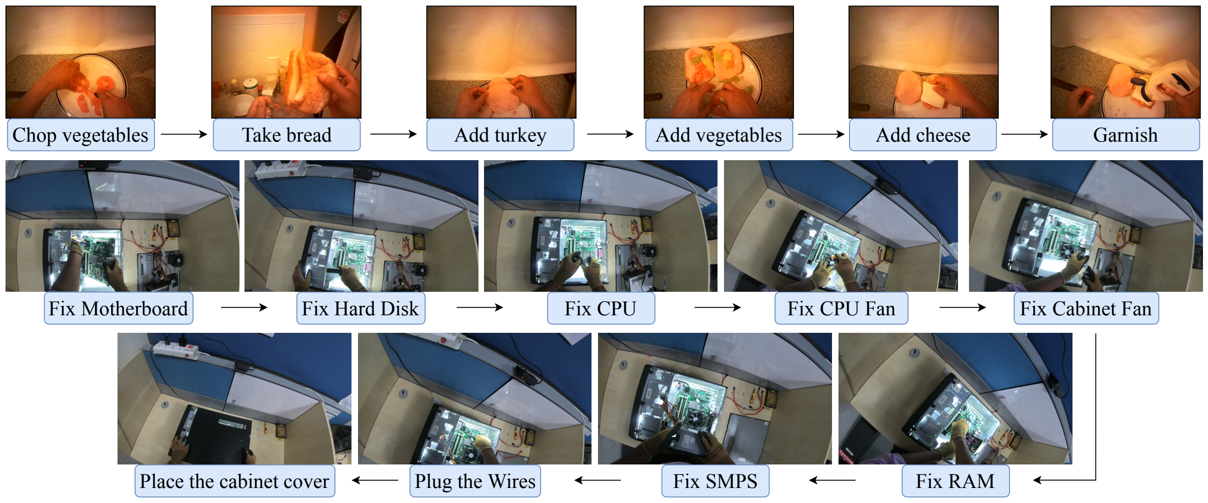


Figure 2.3 Example key-step annotations in EgoProceL for making turkey sandwich [51] and assembling a PC.

have large irrelevant segments. In contrast, egocentric cameras are typically harnessed to the subject's head and have a standardized location. They provide a clearer view of the executed task, including the manipulated objects. As a result, recent works have introduced datasets consisting of egocentric videos [10, 21, 36, 51, 61, 69], which have proven helpful for various tasks [25, 34, 50, 58, 70].

2.2 EgoProceL Dataset for Procedure Learning

Motivated by the advantages of egocentric videos over third-person videos (Section 2.1.2), we propose an egocentric videos dataset for procedure learning: EgoProceL. EgoProceL contains videos and key-step annotations for multiple tasks from CMU-MMAC [12] and EGTEA Gaze+ [51] and individual tasks like toy-bike assembly [62], tent assembly [36], PC assembly, and PC disassembly. EgoProceL consists of 62 hours of annotated egocentric videos, including 16 tasks with an average duration of 13 minutes. To annotate the videos for key-steps, we create a list of key-steps for each task, *e.g.*, assembling a PC requires, 'Fix motherboard', 'Fix hard disk', ..., 'Place the cabinet cover'. We use ELAN [71] to annotate each video by marking the start and end location during which the key-step occurs.

Along with various procedure learning tasks, EgoProceL is appropriate for understanding hand-object interaction, action forecasting and recognition, and a shared study of videos and text. Figure 2.3 shows some example annotations and Table 2.1 compares EgoProceL with existing datasets.

In this section, we first discuss how the videos for EgoProceL were gathered (Section 2.2.1). This is followed by a discussion of the annotation protocols followed (Section 2.2.2). Finally, in Section 2.2.3, we share various statistics for EgoProceL highlighting multiple challenges that the dataset offers.

Table 2.1 Comparison of datasets for Procedure Learning. The average number of key-steps and video length for EgoProceL are the highest, highlighting the complexity of the procedures included in EgoProceL

Dataset	Egocentric View	Manually Created	Avg. key-steps	Avg. Video Length (sec)	#tasks
Breakfast [42]	✗	✓	5.1	137.5	10
Inria [2]	✗	✗	7.1	178.8	5
ProceL [20]	✗	✗	8.3	251.5	12
CrossTask [86]	✗	✗	7.4	297	**18**
EgoProceL (ours)	✓	✓	**8.7**	**769.2**	16

2.2.1 Collecting EgoProceL's videos

2.2.1.1 Protocol for selecting videos from existing Egocentric Datasets

The EgoProceL dataset focuses on the key-steps required to perform a task instead of every action in the video. To construct EgoProceL, we take two approaches: (a) identifying publicly available datasets that we annotate for key-steps; (b) recording new tasks to expand the range of tasks. We follow the following criteria to shortlist from the public datasets:

1. The task should require multiple key-steps to perform. For example, preparing a sandwich involves a minimum of four key-steps [12].

2. Videos of the same task must contain a similar set of key-steps. However, the order of the key-steps can differ.

3. To compare the performance of Correspond and Cut framework proposed in Section 3.2 on egocentric and third-person views, we require a dataset with recordings of the same task in both views.

4. We prefer longer videos with sparse key-steps to generate practical solutions.

We select CMU-MMAC [12], EGTEA Gaze+ [51], MECCANO [62], and EPIC-Tents [36] based on the above criteria. CMU-MMAC contains recordings of subjects performing the same task from one egocentric and four third-person views. Therefore, by using it, we compare the performance of the proposed approach between egocentric and third-person views.

2.2.1.2 Datasets not included in EgoProceL

Here we discuss two potential datasets which we could not use for EgoProceL.

1. **The Charades-Ego dataset** [69], consisting of paired egocentric and third-person videos, is essential for activity recognition. However, it is not practical for procedure learning. The subjects do not perform a series of key-steps to achieve a goal; instead, they perform activities like pouring a drink into a cup and having it. Additionally, the average duration of the videos is 31.2 seconds compared to 13 minutes in EgoProceL, suggesting the briefness of the tasks acted out.

2. **The EPIC-Kitchens dataset** [10], consisting of 100 hours of kitchen recordings, comes quite close to our requirements. However, due to the unscripted nature of the dataset (which sets it apart from [51]), it becomes unsuitable. As for procedure learning, we need videos of the same tasks performed multiple times.

2.2.1.3 Capturing Novel Tasks

Though the four datasets selected above include a diverse range of tasks, they do not contain tasks where the subject works in a constrained environment and deals with small objects (*e.g.*, screws). To alleviate this, we include manually recorded videos of assembling and disassembling a Personal Computer (PC). This addition makes the dataset diverse and challenging in terms of variability in the size of objects involved and the complexity of key-steps (*e.g.*, fixing the motherboard requires fastening nine screws). Furthermore, as highlighted in Section 2.2.3, these two tasks contain the highest foreground ratio. In Figure 2.4, we show example videos from five tasks in EgoProceL (including existing and novel egocentric videos).

2.2.2 Annotating EgoProceL

For annotating CMU-MMAC [12], EPIC-Tents [36], MECCANO [62], PC Assembly, and PC Disassembly, a list of key-steps required to perform the task was created upon viewing the videos. Two annotators were asked to identify the key-steps in the videos and temporally mark the start and end locations. Once an annotator added temporal segments to the videos, the other annotator verified them. We use the ELAN software [71] to annotate the videos.

For annotating EGTEA Gaze+ [51], we used the recipes provided by the dataset curators to create the key-step's list for each task. The dataset offers dense activity annotations for all the videos. We created a one-to-many mapping between the key-steps and the provided annotations; this accelerated the annotations process. The mapping generated was used to create key-step annotations for all videos. Three people further watched the videos and verified the annotations generated.

Figure 2.3 shows example annotations for making a turkey sandwich [51] and assembling a PC. To accelerate future research, we release the EgoProceL dataset on the project's web page[1].

[1]Link 1: http://cvit.iiit.ac.in/research/projects/cvit-projects/egoprocel; Mirror link 2: https://sid2697.github.io/

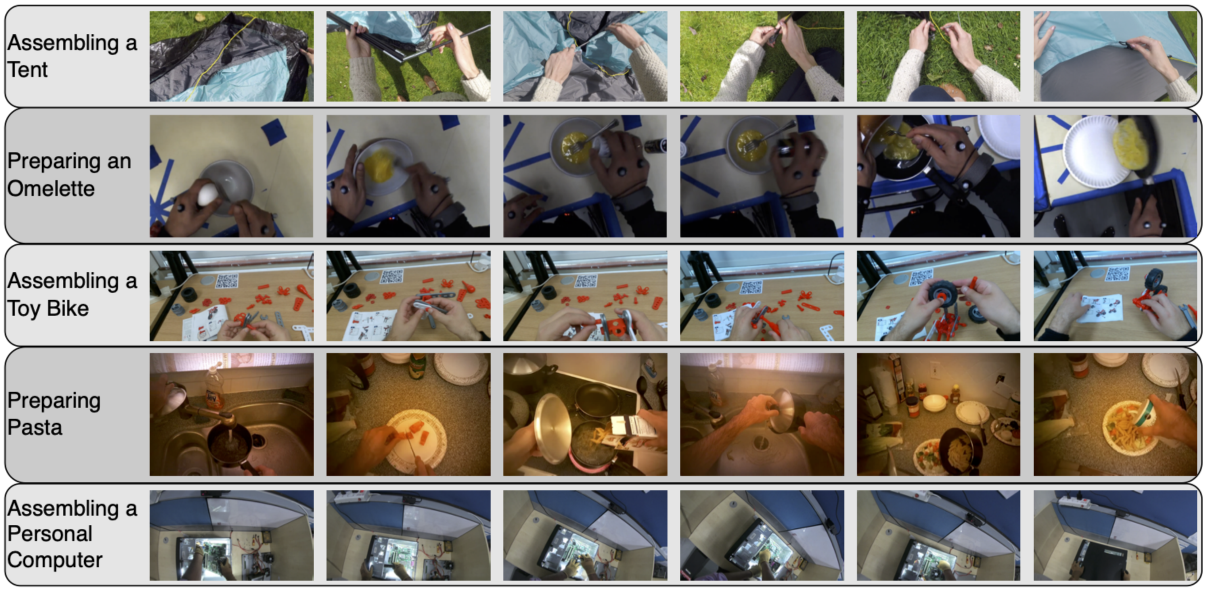


Figure 2.4 Example videos in EgoProceL. Here, we show frames from five different tasks in EgoProceL. The tasks include assembling a tent [36], preparing an omelette [12], assembling a toy bike [62], preparing pasta [51], and assembling a personal computer. In EgoProceL, we provide key-step annotations for eligible existing egocentric datasets (Section 2.2.1) and provide additional tasks (PC Assembly and PC Disassembly).

2.2.3 Task-level details of EgoProceL

In Table 2.2, we share the statistics for each of the 16 tasks in the EgoProceL dataset. Here, N is the number of videos, K is the number of key-steps for a task, u_n is the number of unique key-steps and g_n is the number of annotated key-steps for n^{th} video. We follow [20] to calculate the following statistics.

2.2.3.1 Foreground Ratio

It is the ratio of total duration of the key-steps to the total duration of the video. This helps to understand the amount of background actions a task has. The foreground ratio is inversely proportional to the amount of background. It is calculated as:

$$F = \frac{\sum_{n=1}^{N} \frac{t_k^n}{t_v^n}}{N} \tag{2.1}$$

Here, t_k^n and t_v^n are the key-step duration and video duration for n^{th} video, respectively. The range of F is between 0 and 1.

From Table 2.2, we can see that the tasks have significant variance in the foreground ratio. Conversely, tasks like “PC Assembly” and “Tent Assembly” have a high foreground ratio, suggesting fewer

Table 2.2 Statistics of the EgoProceL dataset across different tasks. The high range of the foreground ratio and repeated steps highlights the complexity of the tasks involved in EgoProceL

Task	Videos Count	Key-steps Count	Foreground Ratio	Missing Key-steps	Repeated Key-steps
PC Assembly	14	9	**0.79**	0.02	0.65
PC Disassembly	15	9	0.72	**0.00**	0.60
Toy Bike Assembly	20	**17**	0.50	0.06	0.32
Tent Assembly	29	12	0.63	0.14	0.73
Bacon and Eggs	16	11	0.15	0.22	0.51
Cheese Burger	10	10	0.22	0.22	0.65
Continental Breakfast	12	10	0.23	0.20	0.36
Greek Salad	10	4	0.25	0.18	0.77
Pasta Salad	19	8	0.25	0.19	**0.86**
Hot Dog Pizza	6	8	0.31	0.13	0.62
Turkey Sandwich	13	6	0.21	0.01	0.52
Brownie	**34**	9	0.44	0.19	0.26
Eggs	33	8	0.26	0.05	0.26
Pepperoni Pizza	33	5	0.53	**0.00**	0.26
Salad	**34**	9	0.32	0.30	0.14
Sandwich	31	4	0.25	0.03	0.37

background actions. On the other hand, tasks like preparing “Bacon and Eggs” and “Turkey Sanwich” have low foreground ratios, suggesting more background actions.

2.2.3.2 Missing Key-steps

This measure captures the count of missed key-steps in each video. It is defined as:

$$M = 1 - \frac{\sum_{n=1}^{N} u_n}{KN} \tag{2.2}$$

The range of M is between 0 and 1. It helps us understand if a task can be done even if we miss some steps. For example, in Table 2.2, “Salad” has the highest missing key-steps ratio suggesting that salad can be made if we miss multiple key-steps. This makes sense, as one can miss adding mayonnaise to the salad but still create an edible salad. On the other hand, tasks like “PC Disassembly” and “Pepperoni Pizza” can not afford to miss key-steps as the task won’t be complete. So, for such tasks, we see a missing key-step ratio of 0.

2.2.3.3 Repeated Key-steps

This measure captures the repetitions of key-steps across the videos. It is defined as:

$$R = 1 - \frac{\sum_{n=1}^{N} u_n}{\sum_{n=1}^{N} g_n} \tag{2.3}$$

The range of R is between 0 and 1. Higher values of R indicate repetitions of key-steps across videos. From Table 2.2, we can see preparing “Pasta Salad” has the highest repeated key-steps and preparing “salad” has the lowest. Methods that do not consider repetitions of the key steps will not perform well for such tasks. As UnityGraph takes repetitions of the key steps into consideration, it performs well.

2.3 Summary

To conclude, in this chapter, we first discuss existing datasets for procedure learning and then look at their shortcomings. To fix the shortcomings, we propose the EgoProceL dataset. For EgoProceL, we have a look at the protocols followed in selecting and annotating the videos. Finally, we have a look at various statistics to highlight multiple aspects of the proposed EgoProceL dataset. In the next chapter, we propose a framework that exploits the correspondences across videos of the same task to learn the procedure.

Chapter 3

Aligning the Videos for Discovering the Procedure

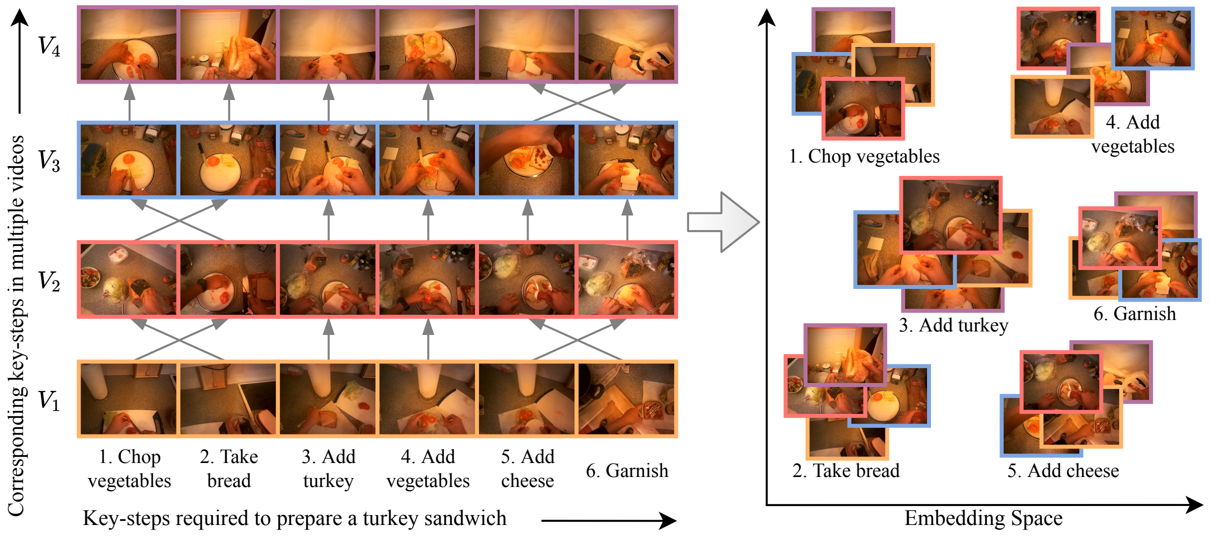


Figure 3.1 Existence of Correspondences across the Videos. The left-hand side figure shows six key-steps required to prepare a turkey sandwich across four egocentric videos. The arrows among the videos highlight the change in the ordering of corresponding key-steps. This thesis utilizes these correspondences and aims to learn an embedding space where the corresponding key-steps have similar embeddings (right-hand side figure). To this end, we propose the Correspond and Cut (CnC) framework, which learns the embedding space and utilizes it to localize the key-steps and identify their ordering.

In this chapter, we have a detailed look at the proposed Correspond and Cut (CnC) framework for procedure learning (Section 3.2). Also, we compare the proposed CnC framework with state-of-the-art methods on third-person datasets (Section 3.4.1). Furthermore, we evaluate CnC on the proposed EgoProceL dataset (Section 3.4.2). Finally, we compare the performance of CnC on different camera views (Section 3.4.3), and also we perform an extensive ablation study (Section 3.5).

3.1 The Motivation behind Aligning the Videos

Imagine showing an autonomous agent how to prepare a sandwich, and it learns the steps required for it! Motivated by this vision, this chapter focuses on developing a framework that allows an agent to identify the steps required to perform a task and their order after observing multiple visual demonstrations by experts. Furthermore, due to the high utility of egocentric videos, we explore procedure learning using the proposed EgoProceL dataset (Chapter 2). However, egocentric videos come with their own set of challenges. For example, the camera view undergoes extreme movements due to the wearer's head motion, introducing frames unrelated to the activity and unavailability of the actor's pose [70].

To overcome the challenges and learn the procedure from egocentric videos, we propose utilizing the signal provided by temporal correspondences across videos. As shown in Figure 3.1, critical moments like putting a slice of turkey on the bread while preparing a turkey sandwich are present across all the videos. To exploit the signal provided by such temporal correspondences, we propose a self-supervised, three-stage, Correspond and Cut (CnC) framework for procedure learning. The first stage of the CnC uses the proposed self-supervised TC3I loss to learn an embedding space such that the same key-steps across the videos have similar embeddings (Figure 3.1). The second stage consists of the proposed ProCut Module (PCM). PCM performs clustering on the learned embeddings and assigns each frame to a key-step. The final stage of CnC creates a key-step sequence for each video and infers relevant ordering to perform the task.

3.2 Correspond and Cut framework for Procedure Learning

Humans often follow the same steps to perform any particular task, though the order of steps might be different. This thesis proposes a methodology that, given a set of videos of humans performing a task, learns similar embeddings across videos for the key-steps required to complete a task. Once we have the embeddings, learning a procedure reduces to clustering the embeddings for localizing the key-steps among all the videos. To learn the embeddings, we exploit temporal correspondences between the videos of the same task. For that purpose, we train a representation learning network using the proposed TC3I loss (Section 3.2.1). TC3I builds on top of existing temporal video alignment methods [18, 30]. After learning the embeddings, we use PCM, shown in Figure 3.2, to cluster and localize the underlying key-steps. PCM models the clustering problem as a multi-label graph cut problem and solves it to localize the key-steps (Section 3.2.2). Once we localize the key-steps using PCM, we use the frame's relative location in a video to generate the key-step ordering for each video (Section 3.2.3).

Notation: CnC takes in $V = \{V_i : i \in \mathbb{N}, 1 \leq i \leq n\}$ untrimmed videos of the same task, where n is the total number of videos. Each of the n videos can have a different number of frames. We denote the embedding function used to generate the frame-level embeddings as f_θ, which is a neural network with parameters θ. A video V_k with m frames is denoted as $V_k = \{f_k^1, f_k^2, \ldots, f_k^m\}$ and the video's

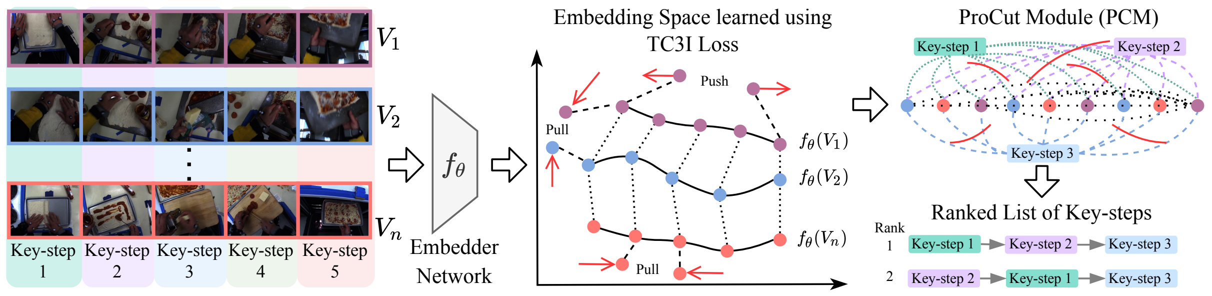


Figure 3.2 Correspond and Cut (CnC) framework for Procedure Learning. CnC takes in multiple videos from the same task and passes them through the embedder network trained using the proposed TC3I loss. The goal of the embedder network is to learn similar embeddings for corresponding key-steps from multiple videos and for temporally close frames. The ProCut Module (PCM) localizes the key-steps required for performing the task. PCM converts the clustering problem to a multi-label graph cut problem solved using the Alpha Expansion algorithm [5]. The output provides the assignment of frames to the respective key-steps and their ordering.

frame-level embeddings are denoted as $f_\theta(V_k) = \{v_k^1, v_k^2, \ldots, v_k^m\}$. We assume K key-steps in a task, where K is a hyper-parameter.

3.2.1 TC3I Loss for Learning the Embeddings

We aim to learn similar embeddings for the frames with comparable semantic information across different temporal locations from multiple videos. For that purpose, we use Temporal Cycle Consistency (TCC) [18] to find correspondences across time in videos.

Consider two videos V_1 and V_2, with lengths p and q, respectively. To check if a point v_1^i in V_1 is cycle consistent, its nearest neighbour $v_2^j = \arg\min_{v_2 \in V_2} \|v_1^i - v_2\|$ is calculated in V_2. Then the process is repeated for v_2^j in V_1 to get $v_1^k = \arg\min_{v_1 \in V_1} \|v_2^j - v_1\|$. If $i = k$, then the point is considered as cycle consistent. An acceptable embedding space consists of a maximum number of cycle-consistent points for a pair of sequences. Specifically, for a point v_1^i in V_1, we determine its soft nearest neighbor $\widetilde{v}_2$ in V_2 by using the softmax function as follows:

$$\widetilde{v}_2 = \sum_j \alpha_j v_2^j, \quad \text{where} \quad \alpha_j = \frac{e^{-\|v_1^i - v_2^j\|^2}}{\sum_k e^{-\|v_1^i - v_2^k\|^2}}. \tag{3.1}$$

Here α_j signifies the *similarity* between v_1^i and individual $v_2^j \in V_2$. Once we have the soft nearest neighbor, a similarity vector β_1^i is calculated. β defines the proximity between $\widetilde{v}_2$ and each frame $v_1^k \in V_1$ as:

$$\beta_1^i[k] = \frac{e^{-\|\widetilde{v}_2 - v_1^k\|^2}}{\sum_j e^{-\|\widetilde{v}_2 - v_1^j\|^2}}. \tag{3.2}$$

As β is a discrete distribution of similarities over time, it peaks around the i^{th} time index. To avoid this, a Gaussian prior is applied to β by minimizing the normalized squared distance $\frac{|i-\mu|^2}{\sigma^2}$ as the objective. By applying additional variance regularization, β is enforced to be peaky around i. Hence, the final cycle consistency loss between videos V_1 and V_2, corresponding to i^{th} frame of V_1 is:

$$L(V_1, V_2, v_1^i) = \frac{|i-\mu|^2}{\sigma^2} + \lambda \log(\sigma). \tag{3.3}$$

Here, $\mu_i = \sum_k \beta_1^i[k] \times k$ and $\sigma_i^2 = \sum_k \beta_1^i[k] \times (k-\mu_i)^2$, and λ is the regularization weight. Formulating TCC in this way ensures the model is not heavily penalized when it cycles back to close-by frames.

We observe that there are many repetitive frames in egocentric videos because of which cycle consistency loss often loops back to similar but temporally far-away frames. To alleviate the issue, we utilize the Contrastive-Inverse Difference Moment (C-IDM) loss [30] (a modified form of Inverse Difference Moment [9]) for applying temporal regularization on each video. The C-IDM loss between the two frames i and j of a video V_1 is computed as:

$$I(V_1, i, j) = (1 - \mathcal{N}(i,j))\, \gamma(i,j) \max(0, \zeta - d(i,j)) + \mathcal{N}(i,j) \frac{d(i,j)}{\gamma(i,j)}. \tag{3.4}$$

Here, $\gamma(i,j) = (i-j)^2 + 1$, $d(i,j)$ is the Euclidean distance between $f_\theta(v_1^i)$, and $f_\theta(v_1^j)$, ζ is the margin parameter, and $\mathcal{N}$ is the neighborhood function such that, $\mathcal{N}(i,j) = 1$ if $|i-j| \leq \sigma$, and 0 otherwise. Here, σ is the window size for separating temporally far away frames. The C-IDM loss encourages the embeddings of the temporally close frames to be similar and the embeddings of temporally far frames to be dissimilar. The final loss combines both TCC and C-IDM (referred to as TC3I loss from now on):

$$\begin{aligned} \text{TC3I}(V_1, V_2) = &\sum_{i \in V_1} L(V_1, V_2, v_1^i) + \sum_{j \in V_2} L(V_1, V_2, v_2^j) \\ &+ \xi \sum_{i \in V_1} \sum_{j \in V_1} I(V_1, i, j) + \xi \sum_{i \in V_2} \sum_{j \in V_2} I(V_2, i, j). \end{aligned} \tag{3.5}$$

Here, ξ is a regularization parameter.

3.2.2 ProCut Module for Learning the Key-steps

Once we learn the embeddings, we aim to localize the key-steps required for performing the task. Kukleva *et al.* [44] localize the key-steps by generating K clusters of embeddings using the K-Means algorithm [53]. However, they need to assume a fixed order of key-steps to assign frames to the key-steps. Instead, we propose a novel ProCut Module (PCM) for this purpose. PCM converts the clustering problem to a multi-label graph cut problem [26], as described below.

Let $G = \langle V, E \rangle$ be a graph consisting of a set of nodes V and a set of directed edges E connecting them. The node set V consists of K *terminal nodes* representing the key-steps, and *non-terminal nodes* (equal to the number of frames) representing the embeddings of the frames generated using the Embedder network. There are two kinds of edges in the graph: *t-links* connecting non-terminal nodes to the terminal nodes, and *n-links* connecting two non-terminal nodes.

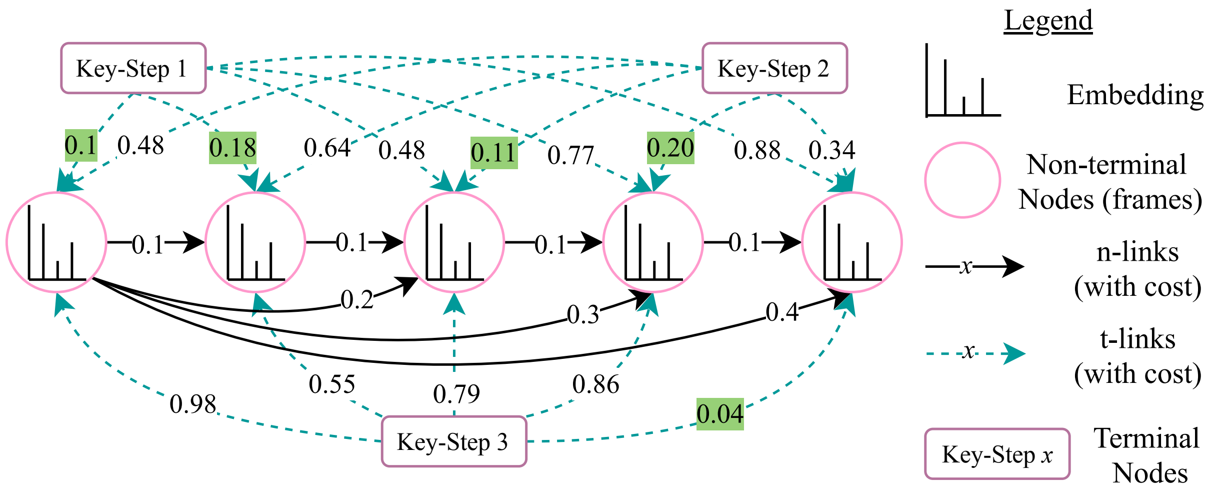


Figure 3.3 ProCut Module (PCM). Non-terminal nodes in the graph represent the embeddings of the frames. Terminal nodes represent the key-steps required to perform the task. The terminal and non-terminal nodes are connected using the t-links. Non-terminal nodes are connected using the n-links. The numbers inscribed in arrows represent the cost of using the respective link. Costs highlighted in green represent the lowest cost to assign a frame to the key-step. For brevity, n-links are shown only for the first non-terminal node. Diagram best viewed in colour.

We use the Fuzzy C-Means algorithm [17] to assign a cost to the *t-links*. The algorithm performs soft clustering and calculates the probability of a frame belonging to each cluster. We subtract the probability value from 1 to obtain the cost of assigning a frame to each cluster. The cost value for the *n-links* is assigned based on the temporal distance between the nodes. For example, if the nodes are temporally closer (*e.g.*, nodes at positions 1 and 2 in Figure 3.3), the cost of assigning the same label to them is lower; otherwise (*e.g.*, for nodes at positions 1 and 5 in Figure 3.3), the cost is high. After creating the graph G, we use α-Expansion [5] to find the minimum cost cut. We use the discovered cut to assign frames to K labels. As shown in Figure 3.3, the lowest costs (highlighted in green) result in assigning the first and second frames to key-step 1, the third and fourth frames to key-step 2, and the last to key-step 3.

3.2.3 Determining Order of the Key-steps

When it comes to determining the ordering of the key-steps, it makes sense to allow each video to have a distinct key-step ordering, as there can be multiple ways to perform a task. However, current works either use a fixed order of key-steps to decode all the videos [20, 44, 77] or do not predict the ordering [19, 68]. One of the advantages of using CnC to determine the key-step is that it allows each video to have its independent order of the key-steps.

To infer the sequential order of key-steps, we calculate the normalized time for each frame v_i^n in video V_i consisting of p frames as $T(v_i^n) = \frac{n}{p}$ [44, 77]. Then we calculate the time instant for each cluster as the average normalized for frames assigned to it. The clusters are then arranged in increasing order of the average time, providing us with the sequence of key-steps used to perform the task in a video. Once we have key-step order for all the videos of the same task, we generate their ranked list based on the number of times the subjects followed a particular order. The order followed the most ends up being at the top of the ranked list. Doing this enables us to determine different sequential orders of key-steps to accomplish a task.

3.3 Experiments

3.3.1 An Updated Evaluation Protocol

Current works mostly use frame-wise metrics to evaluate the models developed for procedure learning [19, 20, 44, 68, 77]. While these metrics evaluate the procedure reasonably well compared to simply calculating the accuracy, they do not suit datasets with significant class imbalance. Furthermore, procedure learning datasets consist of significant background frames [86]. Hence, a model assigning all the frames to the background might achieve high scores. We propose to solve this problem by calculating the scores via the contribution of each key-step, leading to lower scores when models assign most of the frames to the background. Further, when comparing with the previous works, (a) we use CnC on standard third-person benchmark datasets [20, 86] and (b) employ existing metrics to evaluate.

Current works compute framewise F1-Score and IoU scores for key-step localization [19, 20, 44, 68, 77]. The F1-Score is a harmonic mean of precision and recall scores. For calculating recall, the ratio between the number of frames having correct key-steps prediction and the number of ground truth key-step frames across all the key-steps of a video is calculated. For precision, the denominator is the number of frames assigned to the key-steps. For calculations, the one-to-one mapping between the ground truth and prediction is generated using the Hungarian algorithm [43] following [2, 19, 20, 44, 68]. However, these metrics tend to assign high scores to models that assign most frames to a single cluster, as the key-step with most frames matches with the background frame's label in the ground truth. Furthermore, most of the frames are background for untrimmed procedure learning videos, resulting in high scores.

Shen *et al.* [68] attempt to solve this problem by analyzing the MoF score, but as pointed out in [44], MoF is not always suitable for an imbalanced dataset. Instead, we propose calculating the framewise scores for each key-step separately and then taking the mean of the scores over all the key-steps. This penalises the cases when there is a large performance difference for different key-step, *e.g.*, when all the frames get assigned to a single key-step. Upon following this protocol, the scores for all the methods decrease. This thesis presents the results generated using the proposed evaluation protocol unless otherwise mentioned.

3.3.2 Implementation Details

We use ResNet-50 [31] as our backbone network to extract the features. Motivated by [18], for training the Embedder network, we use a pair of training videos at a time, select frames at random within the videos, and optimize the proposed TC3I loss until convergence. The features are extracted from the *Conv4c* layer, and a stack of c context frames features is created along the temporal dimension. We reshape our input video frames to 224×224. To aggregate the temporal information, we pass the combined features through two 3D convolutional layers followed by a 3D global max pooling layer, two fully-connected layers, and a linear projection layer to output the embeddings of dimension 128. We set the value of K to 7 and compare the performance of CnC with the other values of K in Table 3.7. Furthermore, for all our experiments, we follow the task-specific settings laid out in [19]. We use PyTorch [60] for all our experiments.

3.3.2.1 List of Hyper-parameters

Table 3.1 lists the hyper-parameters used for CnC.

Table 3.1 Hyper-parameter values for CnC

Hyper-parameter	Value
No. of key-steps (K)	7
No. of sampled frames	32
Batch Size	5
Learning Rate	10^{-4}
Weight Decay	10^{-5}
Window size (σ)	300
Margin (ζ)	2.0
Regularization parameter (ξ)	1.0
No. of context frames (c)	2
Context stride	15
Embedding Dimension	128
Optimizer	Adam [40]

3.3.3 Baselines

We consider three baseline methods to evaluate CnC on EgoProceL:

1. **Random.** Here we predict the labels by randomly sampling predictions from a uniform distribution with K values representing K key-steps.

2. **TC3I + HC.** Instead of PCM, we use the K-Means algorithm and generate K clusters from the representation space.

3. **TC3I + SS.** Here, instead of PCM, we use subset selection for the key-step assignment. The algorithm takes in the frame's embeddings and M (hyper-parameter) latent states obtained using K-Means [53]. It then selects a subset S (of size K) of the states as key-steps and finds the frames' assignments. We use the greedy algorithm used in [19] to perform subset selection. Refer to the supplementary material for the hyper-parameter values.

3.4 Results

3.4.1 Third-person Videos

Table 3.2 Procedure Learning from Third-person Videos. Comparison between state-of-the-art methods and CnC on benchmark third-person video datasets [20, 86]. Our method outperforms all the techniques using videos only (in F-Score). It even manages to give at par performance compared to the techniques using multi-modal input. **P**, **R**, and **F** represent precision, recall, and F-score, respectively

	Input Modality	ProceL [20]			CrossTask [86]		
		P	**R**	**F**	**P**	**R**	**F**
Uniform	Video	12.4	9.4	10.3	8.7	9.8	9.0
Alayrc *et al.* [2]	Video + Narrations	12.3	3.7	5.5	6.8	3.4	4.5
Kukleva *et al.* [44]	Video	11.7	30.2	16.4	9.8	35.9	15.3
Elhamifar *et al.* [19]	Video	9.5	26.7	14.0	10.1	**41.6**	16.3
Fried *et al.* [24]	Video	–	–	–	–	28.8	–
Shen *et al.* [68]	Video + Narrations	16.5	**31.8**	21.1	15.2	35.5	21.0
CnC (*ours*)	Video	**20.7**	22.6	**21.6**	**22.8**	22.5	**22.6**

To test the generalizability of CnC on third-person videos and to ensure a fair comparison with existing methods [2, 19, 24, 44, 68], we perform experiments on third-person procedure learning benchmark datasets: ProceL [20] and CrossTask [86]. We obtain the results of previous works from [68]. Note that here we use the evaluation protocol employed by the previous works [19, 20, 44, 68]. As seen in Table 3.2, CnC outperforms other methods (in terms of the F-Score) utilizing only videos as the input

Table 3.3 Procedure Learning Results obtained on EgoProceL. Here, CnC performs the best, highlighting the effectiveness of the TC3I loss and PCM

	EgoProceL											
	CMU-MMAC		EGTEA G.		MECCANO		EPIC-Tents		PC Assembly		PC Disas.	
	F1	IoU	F1	IoU	F1	IoU	F1	IoU	F1	IoU	F1	IoU
Random	15.7	5.9	15.3	4.6	13.4	5.3	14.1	6.5	15.1	7.2	15.3	7.1
TC3I + HC	19.2	9.0	20.8	7.9	16.6	**8.0**	15.4	7.8	21.7	11.0	24.9	14.1
TC3I + SS	19.7	8.9	20.4	7.9	16.3	7.8	15.9	7.8	24.8	11.9	23.6	14.0
CnC	**22.7**	**11.1**	**21.7**	**9.5**	**18.1**	7.8	**17.2**	**8.3**	**25.1**	**12.8**	**27.0**	**14.8**

modality. Further, with only video as the input modality, CnC even manages to perform at par with multi-modal methods. Previous works have used different forms of self-supervision. For example, [19] use the pseudo-labels provided by subset selection, and [44] utilize the relative time-stamps of video frames. Instead, the comparison in Table 3.2 shows that the signal provided by corresponding frames is superior for the task of procedure learning.

3.4.2 Egocentric Videos

Table 3.3 summarises the results obtained on EgoProceL using the baselines and proposed CnC. CnC performs higher than all three baselines. This is due to (a) the ability of the TC3I loss to learn the representation space where similar key-steps lie close without enforcing any ordering or temporal constraints. Moreover, TC3I adds temporal coherency to the learned representations by adopting the C-IDM loss [30] (Figure 3.4). (b) PCM gains a comprehensive view of the problem by considering the cost of assigning each frame belonging to every key-step and its temporal relationship with the other frames. CnC performs better on long sequences as the TC3I loss compensates by searching for corresponding frames in the entire length of the videos, making it possible to learn a reasonable representation space despite the length of the videos. Further, the results in Table 3.3 show that PCM is superior for key-frame clustering and assignment along with TC3I as it results in the highest F-Score and IoU on EgoProceL. The gain in performance is because PCM considers the cost of assigning each frame to every key-step and its temporal relationship with the other frames (Figure 3.4). This allows PCM to gain a comprehensive view of the problem compared to HC, which does not consider the cost of each frame belonging to other key-steps and SS, which has lower generalisation power [19].

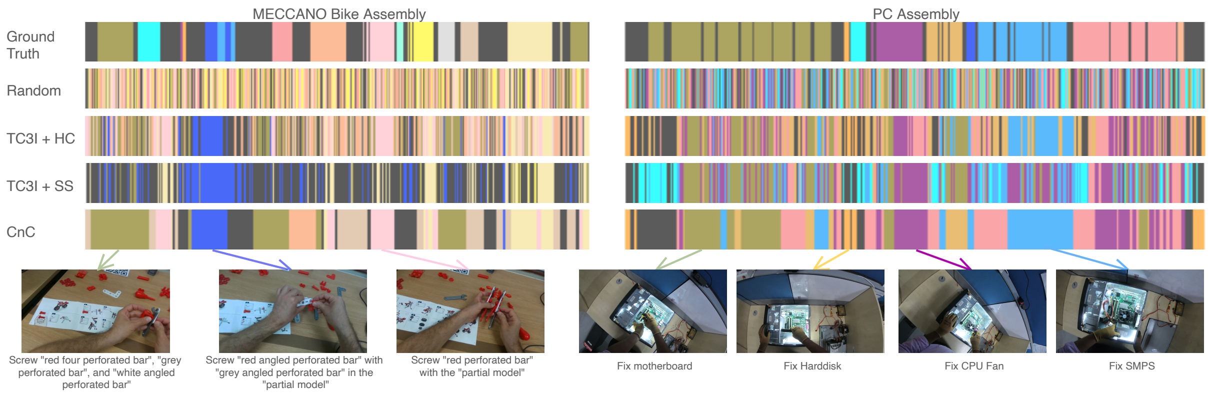


Figure 3.4 Qualitative results for MECCANO and PC Assembly highlight the effectiveness of CnC. Additionally, PCM outperforms HC and SS when clustering the key-steps. Furthermore, due to the TC3I loss, CnC correctly identifies the key-steps that are short (fix a hard disk in PC Assembly). The gray segments denote the background.

Table 3.4 Egocentric vs. Third-person results. We use different views from [12] for comparison. We obtain better results using CnC on egocentric videos highlighting their effectiveness

View	Precision	Recall	F1-Score	IoU
Third-person (Top)	17.4	18.4	17.9	8.1
Third-person (Back)	18.8	21.5	20.0	8.5
Third-person (LHS)	21.2	22.7	21.8	9.7
Third-person (RHS)	19.8	21.7	20.6	8.7
Egocentric	**21.6**	**24.4**	**22.7**	**11.1**

3.4.3 Egocentric vs. Third-person Videos

Here, we compare the results obtained after training CnC on multiple views from CMU-MMAC [12]. As seen in Table 3.4, the frame-wise F1-Score and IoU scores are the highest for the egocentric view. This is because egocentric videos offer lower occlusion by the expert's body and provide higher visibility of hand-object interactions. This highlights one of the central hypotheses of this thesis: the effectiveness of using egocentric videos over third-person videos for procedure learning. Also, we observe that the results vary for third-person videos due to the camera placement. This increases one variable when creating data for procedure learning. Alternatively, egocentric videos use head-mounted cameras, eliminating uncertainty.

3.5 Ablation Study

Here, we quantitatively evaluate our design choices.

3.5.1 Effectiveness of the TC3I Loss

Here, we replace the TC3I loss in CnC with TCC [18], LAV [30], and a combination of LAV and TCC [30] to study the efficacy of the proposed TC3I loss.

Table 3.5 Effectiveness of the TC3I loss. Here, we replace the TC3I loss in CnC with TCC, LAV, and a combination of LAV and TCC. For the majority of the cases, the proposed TC3I loss performs well as it focuses on frame-level correspondences and adds temporal coherency by adopting the C-IDM loss

Experiment	CMU-MMAC [12]			EGTEA Gaze+ [51]		
	Precision	F-Score	IoU	Precision	F-Score	IoU
TCC + PCM	18.5	19.7	9.5	17.5	19.7	8.8
LAV + TCC + PCM	18.8	19.7	9.0	16.4	18.6	7.5
LAV + PCM	20.6	21.1	9.4	17.4	19.1	8.0
TC3I + PCM (CnC)	**21.6**	**22.7**	**11.1**	**19.6**	**21.7**	**9.5**

Experiment	MECCANO [62]			EPIC-Tent [36]		
	Precision	F-Score	IoU	Precision	F-Score	IoU
TCC+PCM	15.1	17.9	**8.7**	14.2	14.9	7.8
LAV+TCC+PCM	13.4	15.6	7.3	16.0	16.7	**8.5**
LAV+PCM	14.6	17.4	7.1	15.2	15.8	8.3
TC3I+PCM (CnC)	**15.5**	**18.1**	7.8	**17.1**	**17.2**	8.3

Experiment	PC Assembly			PC Disassembly		
	Precision	F-Score	IoU	Precision	F-Score	IoU
TCC+PCM	19.9	21.7	11.6	22.0	23.4	12.2
LAV+TCC+PCM	21.6	21.1	10.8	21.0	24.3	12.3
LAV+PCM	21.5	22.7	11.7	26.4	26.5	12.9
TC3I+PCM (CnC)	**25.0**	**25.1**	**12.8**	**28.4**	**27.0**	**14.8**

TC3I loss, in Table 3.5, obtains the highest F-Score and IoU. As observed in our initial set of experiments, TCC loss lacks temporal coherency, due to which temporally close frames do not lie close in the learned representation space, resulting in lower results when compared to TC3I and LAV, which account for temporal coherency using the C-IDM loss. For LAV + TCC, our observations are consistent with [30] because there is no performance gain when directly combining LAV and TCC losses since LAV works on L2-normalised embeddings, whereas TCC does not [30]. The LAV loss performs better than TCC and LAV + TCC; however, the results are not better than TC3I because the Soft-DTW used in LAV accounts for global alignment. However, LAV does not focus on the per-frame features [30], which is beneficial when looking for similar key-steps in different videos. The TC3I loss overcomes these issues by focusing on correspondences in multiple videos at frame level and adding temporal coherency by adopting the C-IDM loss.

Table 3.6 Effectivenss of PCM. Results after replacing PCM with HC and SS with different losses

Experiment	CMU-MMAC [12]				EGTEA Gaze+ [51]			
	Precision	Recall	F-Score	IoU	Precision	Recall	F-Score	IoU
TCC+HC	17.06	19.47	18.08	8.55	16.78	20.00	18.25	8.33
TCC+SS	17.34	19.71	18.31	8.66	16.96	20.29	18.48	8.18
TCC+PCM	**18.46**	**21.45**	**19.71**	**9.46**	**17.46**	**22.71**	**19.74**	**8.81**
LAV+TCC+HC	17.37	18.40	17.76	8.61	**16.59**	19.72	18.02	7.35
LAV+TCC+SS	17.46	17.94	17.57	8.53	16.16	20.05	17.90	7.39
LAV+TCC+PCM	**18.80**	**21.11**	**19.71**	**9.03**	16.44	**21.40**	**18.60**	**7.45**
LAV+HC	18.44	19.78	19.07	8.66	16.59	18.18	17.35	7.87
LAV+SS	17.82	18.99	18.36	8.53	16.08	18.13	17.04	7.87
LAV+PCM	**20.62**	**21.95**	**21.11**	**9.40**	**17.42**	**21.17**	**19.12**	**8.02**
TC3I+HC	18.47	20.27	19.15	8.98	18.74	23.70	20.82	7.93
TC3I+SS	18.53	21.13	19.66	8.86	17.71	24.09	20.36	7.94
CnC	**21.62**	**24.38**	**22.72**	**11.08**	**19.58**	**24.68**	**21.72**	**9.51**

3.5.2 Effectiveness of PCM

Table 3.6 shows the results after using various losses with HC, SS, and PCM for procedure learning [12, 51]. Nearly all the experiments using PCM achieve the highest scores for other losses. Additionally, we achieve the highest scores with CnC. Due to the characteristics of TC3I loss and PCM, the results are consistent with our previous observations.

Table 3.7 Selecting K. Results with various values of K. Numbers in bold are highest in the respective row, and underlined numbers are highest in the respective column

Experiment	CMU-MMAC [12]				EGTEA Gaze+ [51]			
	K=7	K=10	K=12	K=15	K=7	K=10	K=12	K=15
Random	$\mathbf{15.7}$	12.7	11.6	10.4	$\mathbf{15.4}$	12.3	11.4	10.4
TC3I + HC	$\mathbf{19.2}$	17.4	16.3	16.8	$\mathbf{20.8}$	17.8	16.7	17.3
TC3I + SS	$\mathbf{19.7}$	17.3	17.0	15.7	$\mathbf{20.4}$	17.8	16.7	16.8
CnC	$\underline{\mathbf{22.7}}$	$\underline{19.1}$	$\underline{20.4}$	$\underline{20.1}$	$\underline{\mathbf{21.7}}$	$\underline{19.9}$	$\underline{19.9}$	$\underline{19.9}$

Experiment	MECCANO [62]				EPIC-Tents [36]			
	K=7	K=10	K=12	K=15	K=7	K=10	K=12	K=15
Random	$\mathbf{13.4}$	10.1	8.8	7.4	$\mathbf{14.1}$	10.6	9.1	8.3
TC3I+HC	$\mathbf{16.6}$	14.0	11.4	10.8	$\mathbf{15.4}$	$\underline{12.1}$	10.6	9.9
TC3I+SS	$\mathbf{16.3}$	12.6	12.2	10.7	$\mathbf{15.9}$	11.9	10.7	$\underline{10.4}$
CnC	$\underline{\mathbf{18.1}}$	$\underline{15.2}$	$\underline{13.5}$	$\underline{11.9}$	$\underline{\mathbf{17.2}}$	11.1	$\underline{12.1}$	9.46

Experiment	PC Assembly				PC Disassembly			
	K=7	K=10	K=12	K=15	K=7	K=10	K=12	K=15
Random	$\mathbf{15.1}$	11.0	10.4	9.2	$\mathbf{15.3}$	11.8	10.7	9.6
TC3I+HC	$\mathbf{21.7}$	17.3	$\underline{20.7}$	19.2	$\mathbf{24.9}$	18.3	18.0	20.7
TC3I+SS	$\mathbf{24.7}$	18.1	18.1	$\underline{19.7}$	$\mathbf{23.6}$	19.7	21.0	20.7
CnC	$\underline{\mathbf{25.1}}$	$\underline{18.7}$	$\underline{20.7}$	19.0	$\underline{\mathbf{27.0}}$	$\underline{26.5}$	$\underline{24.5}$	$\underline{23.6}$

3.5.3 Selecting the value of K

Table 3.7 contains the results of CnC and the baselines as the function of K. Additionally, it features the results after replacing PCM with HC and SS as the function of K. Here, key observations are: (a) CnC performs the best when $K = 7$, (b) the results do not change significantly for CnC as K increases.

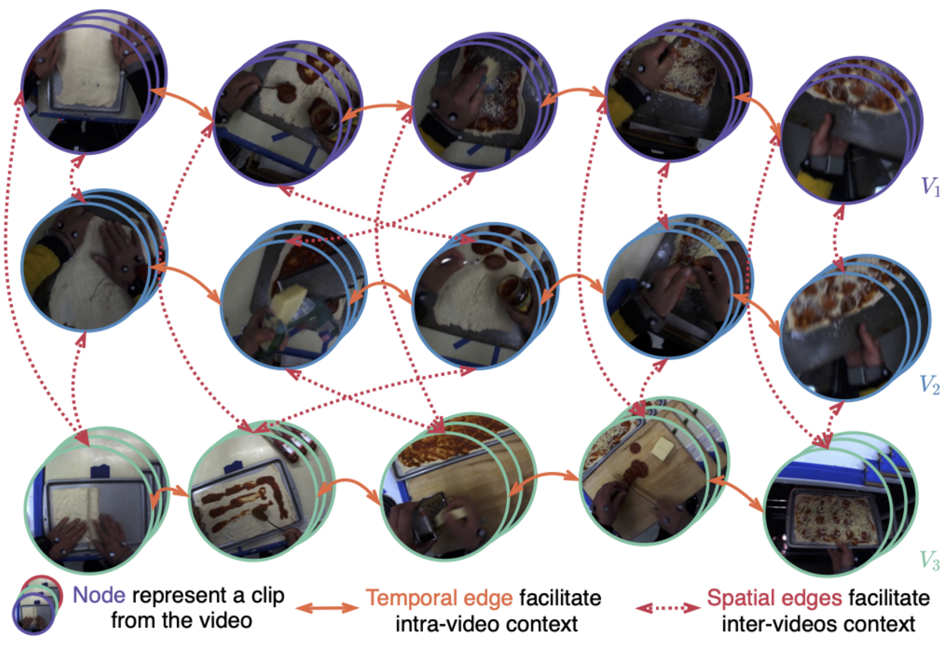
However, we observe a decline in the results for HC and SS as K increases, highlighting the effectiveness of PCM for key-step localisation.

3.6 Summary

Learning procedures from the visual demonstration of a task by an expert is an important step in scaling the learning capabilities of autonomous agents. Unlike current state-of-the-art techniques, instead of third-person videos, we have proposed procedure learning from the first-person viewpoint. We proposed a new technique, CnC, for procedure learning from egocentric videos that utilize the proposed TC3I loss to learn an embedding space in a self-supervised fashion. Finally, we employ PCM to identify the key-steps. Our results demonstrate the superiority of using the egocentric view and the effectiveness of the proposed technique for procedure learning. In the next chapter, we explore and evaluate the utility of graphs for procedure learning.

Chapter 4

Graphs for Procedure Learning



V_1
V_2
V_3
Node represent a clip from the video
Temporal edge facilitate intra-video context
Spatial edges facilitate inter-videos context

Figure 4.1 UnityGraph for three pizza making videos. UnityGraph facilitates procedure learning by creating a unified representation of an arbitrary number of videos from the same category. Here, the nodes represent a clip from the video. Further, the temporal edges connect temporally close frames, allowing intra-video context, whereas the spatial edges connect semantically similar frames across the videos, enabling inter-videos context.

In this chapter, we first motivate using graphs for procedure learning (Section 4.1). Then we take a detailed look at the proposed Graph-based Procedure Learning (GPL) framework (Section 4.2). Furthermore, we compare GPL with the state-of-the-art on third-person datasets (Section 4.4.1) and with CnC on EgoProceL (Section 4.4.2). Finally, we perform an extensive ablation study and determine GPL hyper-parameters for both third- and first-person views (Section 4.5).

4.1 The Motivation behind Utilizing Graphs

Consider developing a robot capable of assembling a phone in a factory. Hard coding the sequence of steps required to piece together the phone will require years of effort. Instead, it would be useful if a robot could observe a person fabricating the phone multiple times and learns from it! Driven by this objective, the focus of this chapter is on the task of unsupervised procedure learning from videos.

As shown in Figure 4.1 and discussed in Section 1.1, procedure learning deals with multiple videos of a task and captures the difference in the order of key-steps, as shown by V2 and V3 (Figure 4.1). Furthermore, as procedure learning deals with localising the key-steps, it differs from the video alignment task. Therefore, considering the utility of procedure learning and its distinctness from existing tasks, we devise the Graph-based Procedure Learning (GPL) framework.

GPL is a three-staged unsupervised framework for procedure learning. The first stage of GPL consists of UnityGraph, shown in Figure 4.1. UnityGraph is a graph that models an arbitrary number of videos from the same task. For creating UnityGraph, the video clips are first passed through a pre-trained I3D ResNet-50 [7, 31] to get a node. The nodes are later connected based on **(a)** semantic similarity across videos (spatial edges) and **(b)** temporal closeness in the same video (temporal edges). Due to this structure, UnityGraph captures both the inter-video and intra-videos context. This sets UnityGraph apart from the previous approaches that estimate the procedure using one [19, 20, 44, 77] or two [3] videos.

As a graph enables us to create a spatial edge between two nodes from different videos irrespective of the key-step order (Figure 4.1), it enables us to overcome previous works' key-step ordering constraints [44, 77]. Also, the range of granularity (number of frames) to create the nodes is controllable, allowing us to test various configurations. Finally, as shown in Figure 4.1, the edges capture two forms of relationships **(a)** temporal across the same video and **(b)** semantic across the videos, enabling us to model inter-video and intra-videos context.

Most works explore procedure learning in a supervised [57, 65, 85] or weakly supervised [4, 8, 15, 33, 48, 49, 63, 64, 86] setting. Supervised methods require frame-level key-step annotations, making them unscalable [3]. On the other hand, weakly supervised learning methods require an ordered or unordered list of key-steps. Creating the lists requires viewing the videos or defining heuristics leading to scalability issues [19, 20]. Instead, the second stage of GPL aims to enhance UnityGraph's node embeddings in an unsupervised manner. To this end, we employ the Node2Vec algorithm [27]. After updating the embeddings, in the final stage, we perform KMeans clustering and localise the key-steps required to perform the task.

The works closely related to ours employ various methods to create frame-level features to identify the procedure. Kukleva *et al.* [44] use the signal provided by the relative timestamp of the frame. Elhamifar *et al.* [19] discover and utilise the attention features from individual frames. These works exploit different attributes of videos to extract the signal. However, they fall short in creating a representation to utilise the relationship between all the frames across the input videos. In this work, we propose UnityGraph, which first creates a clip-level representation and then captures the correspondences between the key-steps across videos.

4.2 Graph-based Procedure Learning (GPL)

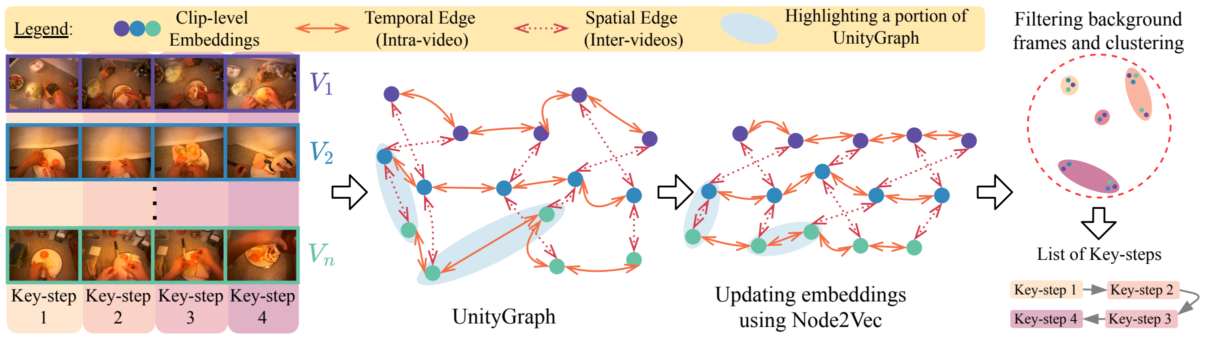

Legend:
Clip-level Embeddings
Temporal Edge (Intra-video)
Spatial Edge (Inter-videos)
Highlighting a portion of UnityGraph
Filtering background frames and clustering
V_1
V_2
V_n
Key-step 1
Key-step 2
Key-step 3
Key-step 4
UnityGraph
Updating embeddings using Node2Vec
List of Key-steps
Key-step 1
Key-step 2
Key-step 3
Key-step 4

Figure 4.2 Graph-based Procedure Learning (GPL) framework. Given multiple videos of the same task, we create UnityGraph. Using the Node2Vec algorithm, we exploit the structure of UnityGraph to enhance the node embeddings in an unsupervised manner. For example, the temporal and spatial clips that were originally far in the embedding space are closer after Node2Vec (highlighted in blue). Finally, we cluster the embeddings using KMeans and filter the background frames to obtain the key-steps required to perform the task.

Autonomously inferring the key-steps required to perform a task opens up the possibility of creating a variety of autonomous, guidance, and assistive systems. The majority of previous works generate self-supervised signals from either single or a couple of videos. However, to better discover the procedure, getting the signal across all the videos is crucial. To this end, we utilise the capability of graphs to represent abstract video data. As shown in Figure 4.2, the first part of GPL framework consists of the proposed UnityGraph. It is a novel graph representation for an arbitrary number of videos of a task (Section 4.2.1).

The initial features of UnityGraph are created using a pre-trained I3D ResNet-50 [7, 31]. To further improve the features, as shown in Figure 4.2, the next step in GPL involves updating the embeddings using the Node2Vec algorithm in an unsupervised manner. Once the embeddings are learned, they are clustered using the KMeans algorithm (Section 4.2.2). The final step of GPL involves ordering the discovered clusters based on the average timestamps of the constituting frames (Section 4.2.2).

Notations: As shown in Figure 4.2, GPL takes in n untrimmed videos of the same task, denoted by $V = \{V_i : i \in \mathbb{N}, 1 \leq i \leq n\}$. Note that the n videos can have a different number of frames. A video V_x with m frames is divided into multiple clips using a sampling rate, stride, and window size of σ, ω, and ψ, respectively. The clips are then passed through a pre-trained I3D ResNet-50, denoted as f_θ (with parameters θ), used to generate node-level embeddings of dimension d for UnityGraph. The clips for video V_x with z clips are denoted as $V_x = \{c_x^1, c_x^2, \ldots, c_x^z\}$ and the video's node-level embeddings are denoted as $f_\theta(V_x) = \{v_x^1, v_x^2, \ldots, v_x^z\}$. Furthermore, we assume K key-steps in a task, where K is a hyper-parameter.

4.2.1 Representing Videos using UnityGraph

We make the following assumptions when creating UnityGraph: **(a)** To compensate for the high frame rate and long action duration, we create UnityGraph's nodes at the clip level. **(b)** Using a 3D CNN, each clip is converted to an embedding. The motivation here is that the sampled clip either contains one action or none. **(c)** To keep the problem tractable, we assume the objects and actions are semantically similar across the task's videos.

4.2.1.1 Creating UnityGraph's Nodes and Edges

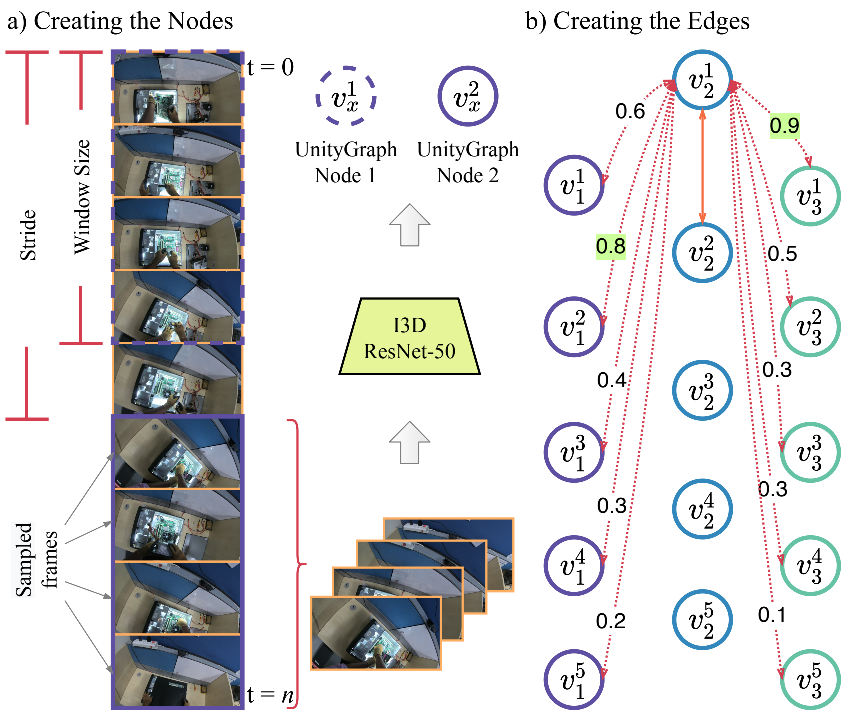


Figure 4.3 Creating UnityGraph's Nodes and Edges. a) Given window size (ψ), stride (ω), and sampling rate (σ), a clip from a video is passed through a pre-trained I3D ResNet-50 to generate the node's embedding. b) We consider nodes from three videos (V_1, V_2, V_3). For brevity, we show the similarity scores between v_2^1 and all the nodes in V_1 and V_3. Edges with the highest semantic similarity (marked in green) are retained.

Figure 4.3 summarises the creation of UnityGraph. A node v_x^1 in UnityGraph is the embedding of a clip c_x^1 for video V_x. The embedding is a d-dimensional vector created using an I3D ResNet-50 [7, 31]. For example, for V_1 with $z = 5$, the nodes are created as:

$$v_1^i = f_\theta(c_1^i), \text{where } i \in \{1, \dots, z\} \tag{4.1}$$

Creating nodes in this way helps with **(a)** converting a volume of frames (the clip) to an embedding and **(b)** comparing and modifying the embeddings to understand the procedure. Figure 4.3 (a) summarises this process.

Once the nodes are created, the graph is completed by creating edges between them. The edges are created at two levels **(a)** spatial, which facilitate inter-videos connection, and **(b)** temporal, which facilitate intra-video connection.

To better understand the process, consider two videos (V_1, V_2) from Figure 4.3 (b). Let us focus on creating an edge between the first node (v_2^1) from V_2 and nodes from V_1. The goal is to find the node in V_1 having the highest semantic similarity with v_2^1. To this end, we calculate the cosine similarity (S_C) between v_2^1 and all the nodes in V_1:

$$S_C(v_2^1, v_1^i) = \frac{\sum_{j=1}^{d} v_{2j}^1 v_{1j}^i}{\sqrt{\sum_{j=1}^{d}(v_{2j}^1)^2}\sqrt{\sum_{j=1}^{d}(v_{1j}^i)^2}}, \text{where } i \in \{1, \ldots, z\}. \tag{4.2}$$

The spatial edge is created between the node with the highest similarity score:

$$\text{Edge}(v_2^1, v_1^i) = \begin{cases} 1, \text{if} \max(S_C(v_2^1, v_1^i)) \\ 0, \text{otherwise} \end{cases}, \text{where } i \in \{1, \ldots, z\}. \tag{4.3}$$

To create the temporal edges, we connect the neighboring nodes from the same video. Let us consider creating temporal edges for V_2:

$$\text{Edge}(v_2^i, v_2^j) = \begin{cases} 1, \text{if } |i - j| = 1 \\ 0, \text{otherwise} \end{cases}, \text{where } i, j \in \{1, \ldots, z\}. \tag{4.4}$$

To summarise, UnityGraph consists of nodes created using Equation (4.1). The nodes are spatially connected using Equation (4.3) and temporally connected using Equation (4.4).

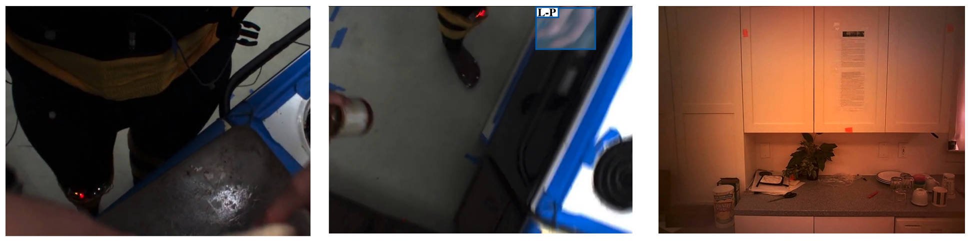
4.2.1.2 Detecting the Background Frames

Procedure learning datasets majorly consists of background frames [3, 86], making it difficult to determine the procedure. We observe that a majority of the background frames involve people searching for objects, reading instructions, and waiting for an automated step to finish. Furthermore, as shown in Fig. 4.4, these activities do not involve hand-object interaction. Therefore, we argue that the frames lacking hand-object interactions represent the background. Figure 4.4 shows how we utilise Shan *et al.*'s [67] hand-object interaction detection model to identify the background frames.

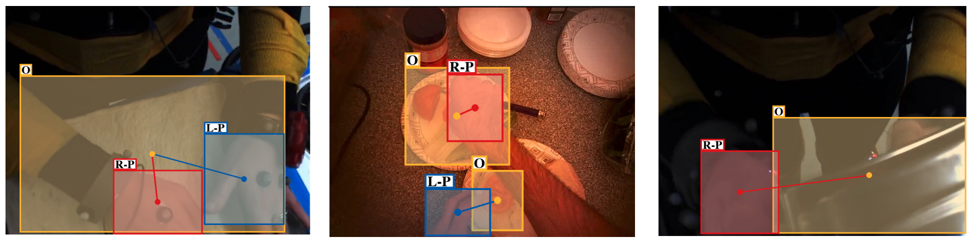
4.2.2 Identifying Key-steps and their Order

4.2.2.1 Updating and Clustering UnityGraph's Embeddings

As illustrated in Figure 4.2, using default embeddings obtained from the pre-trained network can result in embeddings lying far from each other. To further improve the embeddings in an unsupervised



a) No hand-object interaction; Background



b) Hand Object Interaction; Foreground

Figure 4.4 Detecting the Background Frames. We use the hand-object detection model from [67]. a) Frames not containing hand-object interaction. Second image in the first row contains a hand without an interaction with an object, hence, background. b) Frames containing hand-object interaction and contribute towards understanding the procedure.

manner, we propose to update them using the Node2Vec algorithm [27]. Once the embeddings are updated, we use the KMeans algorithm [53] to discover the key-steps.

4.2.2.2 Identifying the Order of Key-steps

Once we have the clusters of key-steps, we follow [3] to determine their order. For each clip, the normalized time is calculated [3, 44]. Based on the cluster clips' normalized time, the average time for the cluster is calculated. The clusters are then arranged in an increasing order of time to generate the order of key-steps. This approach has two advantages **(a)** it allows each video to have its own key-step order, and **(b)** it does not require providing the key-step ordering information.

4.3 Experiments

4.3.1 Evaluation

Unless otherwise mentioned, we evaluate the proposed GPL framework using the metrics from Chapter 3. We take the mean of the scores over all the key-steps and report F1 and IoU Scores. F1-Score is the harmonic mean of the precision and recall scores. For precision, we calculate the ratio between

Table 4.1 Hyper-parameter values for different components of the GPL framework. Here, "**FP**" refers to first-person, "**TP**" refers to third-person, and "Ablation table" refers to the table containing quantitative results for the respective hyper-parameter

Hyper-parameter	Notation	Value (**FP**)	Value (**TP**)	Ablation Table
Sampling Rate	σ	8	4	Table 4.4
Stride	ω	5	10	Table 4.4
UnityGraph's Window Size	ψ	10	10	Table 4.4
Similarity Metric	S_C	Cosine	Cosine	Table 4.5
Walk Count	α	100	50	Table 4.8
Walk Length	γ	100	50	Table 4.8
Node2Vec's Window Size	β	12	10	Table 4.8
Return Parameter	p	1.0	1.0	Table 4.6
In-out Parameter	q	1.0	1.0	Table 4.6
Clustering Technique	–	KMeans	KMeans	Table 4.7
No. of Key-steps	K	7	7	Table 4.9
No. of Videos	n	$\max(n)$	$\max(n)$	Table 4.11
Embedding Dimension	d	400	400	–

the number of frames having correct key-steps prediction and the number of frames assigned to the key-steps. For recall, the denominator is the number of ground truth key-step frames across all the key-steps of the video. Following [2,3,19,20,44,68], we obtain the one-to-one mapping between the ground truth and prediction using the Hungarian algorithm [43].

4.3.2 Implementation Details

We use features from the final layer of 3D ResNet-50 [31] pre-trained on Kinetics 400 [7] provided by PyTorch [60]. To keep feature extraction tractable, we reshape the short side of the video frame to 256, while maintaining the aspect ratio. For detecting the hand-object interactions, we use the 'handobj_100K+ego' model provided by [67]. We create and manipulate graphs using NetworkX [28]. Furthermore, Table 4.1 contains the hyper-parameter values obtained for egocentric and third-person view after an extensive ablation study.

4.3.3 Datasets

Contrary to previous works that use either first- or third-person datasets for procedure learning, we perform experiments on both views. For third-person procedure learning, we choose standard benchmark datasets, CrossTask [86] and ProceL [20]. Both datasets have been created from videos on YouTube. CrossTask consists of 213 hours of videos from 18 primary tasks (2763 videos). ProceL consists of 47.3 hours of videos from 12 diverse tasks (720 videos). To demonstrate the efficiency of the proposed GPL framework, we evaluate it on the first-person EgoProceL [3] dataset. It consists of 62 hours of egocentric videos of 130 subjects performing 16 tasks.

4.3.4 Baselines

(a) **Random**: Here, the labels are obtained by randomly sampling predictions from a uniform distribution with K values representing K key-steps.

(b) **CnC** [3]: This work generates frame-level embeddings by learning an embedding space that exploits temporal correspondences across a couple of videos.

(c) **GPL-I3D**: Here, we do not update the embeddings using Node2Vec. Instead, we utilise Unity-Graph consisting of nodes embeddings from I3D ResNet-50 [7, 31].

Table 4.2 Procedure Learning from Third-person Videos. Comparison between state-of-the-art methods and GPL on third-person datasets [20, 86].

	ProceL [20]			CrossTask [86]		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Uniform	12.4	9.4	10.3	8.7	9.8	9.0
Alayrc *et al.* [2]	12.3	3.7	5.5	6.8	3.4	4.5
Kukleva *et al.* [44]	11.7	30.2	16.4	9.8	35.9	15.3
Elhamifar *et al.* [19]	9.5	26.7	14.0	10.1	41.6	16.3
Fried *et al.* [24]	–	–	–	–	28.8	–
Shen *et al.* [68]	16.5	31.8	21.1	15.2	35.5	21.0
CnC [3]	20.7	22.6	21.6	22.8	22.5	22.6
GPL-I3D (*ours*)	21.3	23.0	22.1	23.4	23.0	23.2
GPL (*ours*)	**22.4**	**24.5**	**23.4**	**24.9**	**24.1**	**24.5**

4.4 Results

4.4.1 Third-person Videos

Table 4.2 compares state-of-the-art methods and GPL on two third-person datasets [20, 86]. We obtain the results for the previous works from [3, 68]. Here, for a fair comparison, the framework is evaluated using the metrics laid out in [19,44,68].

4.4.2 Egocentric Videos

Table 4.3 compares state-of-the-art and GPL on the EgoProceL dataset consisting of 16 tasks. The results for tasks in CMU-MMAC [12] and EGTEA Gaze Plus [51] have been averaged and reported. Note that EgoProceL is a recent dataset for egocentric procedure learning, due to this, there is only one approach (CnC [3]) to fairly compare with. Furthermore, as other methods have been specifically designed around third-person datasets, we compare with them only on those datasets (Table 4.2).

Table 4.3 Results on egocentric view obtained on EgoProceL. Due to higher generalisation capability and effectively modeling the temporal and spatial relationships, the GPL framework performs the best. This highlight the effectiveness of the video representation generated using the proposed UnityGraph and Node2Vec for updating the embeddings based on the node neighborhoods. Note that EgoProceL is a recent dataset for egocentric procedure learning, due to this, there is only one approach (CnC [3]) to fairly compare with. Furthermore, as other methods have been specifically designed around third-person datasets, we compare with them on those datasets in Table 4.2

	EgoProceL											
	CMU-MMAC		EGTEA G.		MECCANO		EPIC-Tents		PC Assembly		PC Disassembly	
	F1	IoU	F1	IoU	F1	IoU	F1	IoU	F1	IoU	F1	IoU
Random	15.7	5.9	15.3	4.6	13.4	5.3	14.1	6.5	15.1	7.2	15.3	7.1
CnC [3]	22.7	11.1	21.7	9.5	18.1	7.8	17.2	8.3	25.1	12.8	**27.0**	14.8
GPL-I3D (*ours*)	28.4	15.6	25.3	14.7	18.3	8.0	16.8	8.2	22.0	11.7	24.2	13.8
GPL (*ours*)	**31.7**	**17.9**	**27.1**	**16.0**	**20.7**	**10.0**	**19.8**	**9.1**	**27.5**	**15.2**	26.7	**15.2**

The results obtained highlight **(a)** the generalisation capabilities of GPL. As GPL obtains high results on tasks using objects with high variability in size, different locations, and a variety of lighting conditions. **(b)** Effectiveness of using UnityGraph for modeling temporal and spatial relationships across the videos. Contrary to the previous works, UnityGraph enables creating a single representation for an arbitrary number of videos. **(c)** Efficacy of utilising Node2Vec for updating the node embeddings. The

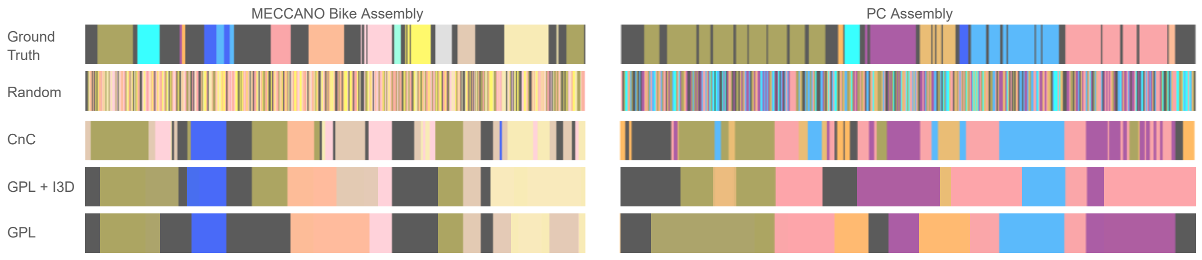


Figure 4.5 Qualitative Results for one video each of Bike and PC Assembly. Each color for a task denotes one key-step and gray sections are the background. The first row contains the ground truth label, the second row contains the results obtained by randomly predicting the key-steps, the third row shows results obtained using CnC [3], the fourth row highlights the results using UnityGraph's node generated using I3D ResNet-50, and the last row shows results obtained for the GPL framework. As can be seen, the segments obtained from GPL are more coherent upon using Node2Vec to update UnityGraph's embeddings. This highlights the efficacy of both, UnityGraph and Node2Vec.

results consistently increase upon using Node2Vec for updating the embeddings of UnityGraph-I3D. This further justifies our hypothesis, shown in Figure 4.2, that Node2Vec improves the embeddings and helps in inferring the procedure.

4.4.3 Qualitative Analysis

Figure 4.5 shows the qualitative results obtained using the baselines and the proposed GPL framework on two tasks from EgoProceL.

Figure 4.6 generated from UnityGraph's embeddings before and after applying the Node2Vec [27] algorithm. On the left-hand side of Figure 4.6, t-SNE visualisation for UnityGraph's clip-level embeddings before applying the Node2Vec algorithm is shown. Though UnityGraph's embeddings are able to capture the background clips well, they fall short of bringing the key-steps close in the embedding space. As can be seen, the ”apply jam” key-step is spread across the embedding space.

On the other hand, on the right-hand side of Figure 4.6, t-SNE visualisation for UnityGraph's clip-level embeddings after applying the Node2Vec algorithm is shown. Here, we can see that the embeddings have arranged themselves in a particular pattern. In the majority of the cases, the embeddings for similar key-steps have come closer. For example, the cluster on the top consists of subjects applying peanut butter. The clips for “apply jam” have concentrated in a few selected clusters. And due to the nature of UnityGraph (connecting clips with similar semantic information), background clips with similar styles of information have come closer. For example, the cluster towards the center has background clips of subjects moving from one location to another.

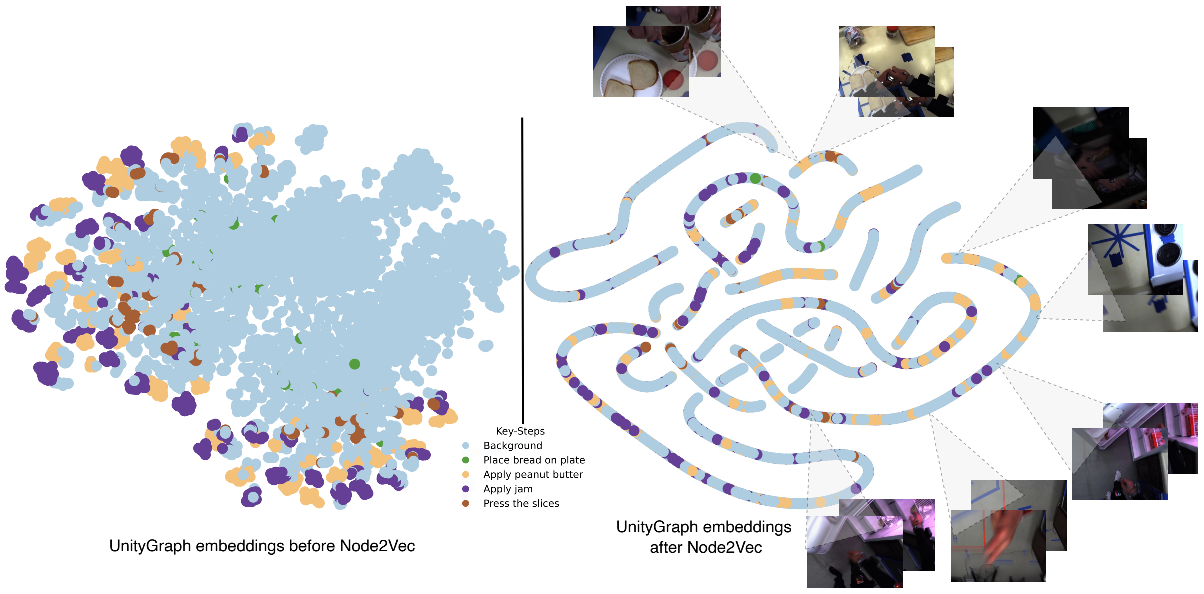


Figure 4.6 t-SNE [76] visualisation for the task of making a sandwich [3, 12] before and after updating UnityGraph's embeddings using the Node2Vec algorithm [27]. Here, each color represents a key-step's category, as noted in the legend. The left side of the figure consists of t-SNE visualisation obtained before using the Node2Vec algorithm. The right side of the figure consists of t-SNE visualisation obtained after updating UnityGraph's embeddings using Node2Vec. As can be seen, upon updating the embeddings using Node2Vec, clips with similar key-steps come close. For example, the cluster on the top consists of clips of subjects applying peanut butter, whereas the cluster towards the centre has background clips of subject moving themselves from one place to other.

4.5 Ablation Study

To strike a balance between finding the optimal hyper-parameters and minimising the environmental impact of the experiments, unless otherwise mentioned, we perform ablation on two challenging tasks: PC Assembly (egocentric) [3] and Change Tire (third-person) [20]. Furthermore, due to the different attributes of first- and third-person videos, we select one set of hyper-parameters for each of the views. Also, unless otherwise mentioned, UnityGraph is created using background frame detection.

4.5.1 Creating UnityGraph

In Table 4.4, we test various values for frame sampling rate (σ), stride (ω), and window size (ψ). As egocentric videos have high motion variability, the maximum scores are obtained for a high sampling rate of 8. Also, the stride (5) and window size (10) are lowest for egocentric videos enabling the creation of nodes with high variability in less time. For third-person videos, the maximum scores are obtained for a low sampling rate of 4. This is because third-person cameras are fixed and do not have high variability in scenes. Furthermore, the stride of 10 and window size of 10 works best as it allows us to sample sparsely and get the information.

Table 4.4 Hyper-parameters for creating UnityGraph. Here, the results are obtained upon changing various parameters for creating UnityGraph. **R**, and **F** represent recall, and F-score, respectively

Sampling Rate	Stride	Window Size	PC Assembly			Change Tire		
			R	**F**	**IoU**	**R**	**F**	**IoU**
4	5	10	23.0	22.6	12.5	23.1	20.7	12.6
8	5	10	**28.8**	**27.2**	**15.1**	23.8	21.0	12.8
4	10	10	20.0	19.9	11.0	**26.2**	**23.2**	**13.9**
8	10	10	28.1	26.8	15.0	23.7	21.1	12.7
4	15	10	25.0	24.3	13.5	23.8	21.4	13.1
8	15	10	25.7	26.6	14.3	23.2	21.0	12.9
4	5	15	22.2	21.7	11.9	24.1	21.0	12.4
8	5	15	21.1	20.6	10.7	21.2	19.6	11.9
4	10	15	25.2	24.0	13.2	23.9	21.6	13.2
8	10	15	23.4	22.3	12.3	25.1	22.8	13.6
4	15	15	23.7	22.9	12.8	22.7	20.1	12.2
8	15	15	22.7	21.8	11.9	24.2	21.8	13.3

Table 4.5 Similarity metrics to create UnityGraph's edges. Here, the results are obtained upon generating edges by various similarity metrics. **R**, and **F** represent recall, and F-score, respectively

Similarity Metric	PC Assembly			Change Tire		
	R	**F**	**IoU**	**R**	**F**	**IoU**
Eucledian	26.0	25.3	13.4	**24.8**	**21.7**	**13.5**
Cosine	**28.8**	**27.2**	**15.1**	23.8	21.0	12.8

Table 4.5 contains the results obtained after using various similarity measurement approaches for creating edges of UnityGraph.

4.5.2 Learning and Clustering the Embeddings

Table 4.6 analyses return and in-out parameters for Node2Vec. Return parameter controls the likelihood of immediately revisiting a node in the walk, and in-out parameter allows the search to differentiate between inward and outward nodes [27]. For both the views, we obtain highest results for p and q as 1.0 and 1.0, respectively.

Table 4.6 Hyper-parameters for walks over UnityGraph. Here, we perform multiple walks to analyse the hyper-parameters for Node2Vec. **R**, and **F** represent recall, and F-score, respectively

Return Parameter	In-out Parameter	PC Assembly			Change Tire		
		R	**F**	**IoU**	**R**	**F**	**IoU**
0.1	0.5	28.7	26.9	14.8	24.5	21.8	13.2
0.1	1.0	28.4	26.7	14.7	24.5	21.8	13.2
0.5	0.1	24.2	23.0	12.3	20.5	18.9	11.4
0.5	1.0	28.5	26.8	14.7	20.6	19.0	11.5
1.0	0.1	24.6	24.6	14.5	**25.6**	**21.9**	13.0
1.0	0.5	28.8	27.3	15.1	20.4	18.9	11.4
1.0	1.0	**29.0**	**27.5**	**15.2**	**25.6**	**21.9**	**13.1**

In Table 4.7, we experiment with two clustering techniques **(a)** KMeans and **(b)** Spectral Clustering. Here, the motivation is to analyse the shape in which the data is lying in the embedding space. Here, KMeans performs the best, showing that the data is mostly spherical in shape and does not require manifold-based clustering techniques.

Table 4.7 Clustering the embeddings. Here, the results are obtained upon using multiple clustering methods on the learned embeddings. **R**, and **F** represent recall, and F-score, respectively

Clustering Technique	PC Assembly			Change Tire		
	R	**F**	**IoU**	**R**	**F**	**IoU**
KMeans	**29.0**	**27.5**	**15.2**	**25.6**	**21.9**	**13.1**
Spectral	14.4	22.0	4.5	12.2	18.6	5.3

Table 4.8 Hyper-parameters for learning the embeddings. Here, the results are obtained upon varying Node2Vec's parameters. **R**, and **F** represent recall, and F-score, respectively

Walk Count	Walk Length	Window Size	PC Assembly			Change Tire		
			R	**F**	**IoU**	**R**	**F**	**IoU**
50	50	8	28.7	27.2	15.1	20.6	19.2	11.6
100	50	8	23.0	22.3	12.1	20.6	19.2	11.6
50	100	8	28.3	26.6	14.7	23.8	21.1	12.9
100	100	8	28.1	26.1	14.3	25.1	21.9	13.1
50	150	8	28.2	26.6	14.7	23.9	21.1	12.9
100	150	8	28.0	26.0	14.3	25.0	21.7	13.0
50	50	10	28.8	27.3	15.1	**26.1**	**22.6**	**13.4**
100	50	10	27.9	26.3	14.5	24.3	21.9	13.2
50	100	10	28.8	27.2	15.1	23.8	21.0	12.8
100	100	10	23.1	22.5	12.1	24.4	21.7	13.2
50	150	10	23.0	22.4	12.0	24.4	22.0	13.3
100	150	10	27.7	26.2	14.4	23.9	21.1	12.9
50	50	12	28.9	27.5	15.2	24.8	21.7	12.9
100	50	12	24.6	23.4	12.5	24.4	21.8	13.2
50	100	12	27.8	26.3	14.3	24.4	21.1	12.7
100	100	12	**29.0**	**27.5**	**15.2**	25.6	21.9	13.1
50	150	12	28.7	27.1	14.8	24.4	21.1	12.7
100	150	12	28.8	27.3	15.0	24.6	21.8	13.2

In Table 4.8, we explore various values for Walk Count (α), Walk Length (γ), and Window Size (β). For egocentric videos, we achieve the best results for α as 100, γ as 100, and β as 12. Due to high variability of scenes in egocentric videos, the walk's count required to update the embeddings are high. This also leads to having a high walk length and a high number of frames in a window. Instead, for third-person videos, we require comparatively less walk count (50), walk length (50), and window size (10). As a majority of the videos in third-person datasets are from the internet, they skip a large number of repetitive portions of the task [3]. Due to this, the number of walks and the length are less. Furthermore, these datasets contain multiple non-relevant frames (explanation/animation) that should be circumvented.

4.5.3 Number of Key-steps and Background Frames

Table 4.9 contains results obtained upon varying the values of K (number of key-steps) for the GPL framework. The results follow the trend in the previous work [3] and are highest for $K = 7$. The results decrease significantly as the values of K increase.

Table 4.9 Tuning K. Here, the results are obtained for various values of K. **R**, and **F** represent recall, and F-score, respectively

K	PC Assembly			Change Tire		
	R	**F**	**IoU**	**R**	**F**	**IoU**
7	**29.0**	**27.5**	**15.2**	**25.6**	**21.9**	**13.1**
10	19.5	20.4	10.4	17.6	16.9	10.2
12	18.8	20.5	10.2	14.4	14.2	8.5
15	19.7	22.5	10.7	13.5	13.6	7.4

As discussed in Section 4.2.1.2, procedure learning datasets have high background frames [86]. To filter the background frames, we detect hand-object interaction in first-person videos. The frames with hand-object interaction are considered foreground, and the rest are background. Table 4.10 shows the results obtained with and without the background frame's filtration. The results improve upon filtering the background frames for the categories that involve working in an open space. For example, in Greek Salad [51], the subjects are working in an unrestricted kitchen, as compared to PC Assembly [3], where they are working in a restricted space with hands being visible a majority of the time.

Table 4.10 Detecting the background frames. Here, the results are obtained upon filtering the frames that do not contain hand-object interaction. Results improve for categories with subjects working in an unrestricted space. **R**, and **F** represent recall, and F-score, respectively

Hand-Object Interaction	PC Assembly			Greek Salad		
	R	**F**	**IoU**	**R**	**F**	**IoU**
Not Checked	**29.5**	**27.6**	14.4	25.4	22.3	12.7
Checked	29.0	27.5	**15.2**	**34.9**	**26.5**	**21.4**

4.5.4 Number of Videos for Creating UnityGraph

Table 4.11 contains the results obtained upon increasing the number of videos used to create UnityGraph. The motivation for these experiments is to explore the effectiveness of the GPL framework as a function of the number of videos. For performing these experiments, we select the tasks that have the number of videos in the powers of two. We create n graph and concatenate the results to keep the evaluation the same as the other experiments. As seen in Table 4.11, the highest F-score and IoU are obtained when the number of videos is the highest. This supports our major claim that using UnityGraph to create a unified representation for all the videos of a task allows us to capture **(a)** temporal relationships in the same video and **(b)** semantic relationships across the videos. Furthermore, as the size of the datasets increase, GPL, along with UnityGraph, will obtain high results.

Table 4.11 Number of Videos. Here, the results are obtained upon systematically increasing the number of videos for creating UnityGraph. **R**, and **F** represent recall, and F-score, respectively

Number of Videos	Bacon and Eggs [51]			Tie-Tie [20]		
	R	**F**	**IoU**	**R**	**F**	**IoU**
4	23.6	20.0	11.4	20.9	18.4	11.2
8	25.0	22.1	12.1	21.3	18.8	11.2
16	**27.8**	**23.1**	**12.6**	20.1	17.6	10.7
32	–	–	–	20.2	18.0	10.8
64	–	–	–	**23.5**	**19.7**	**11.4**

4.6 Summary

Procedure learning is an important direction toward creating systems capable of assisting humans. Contrary to current approaches, we propose the Graph-based Procedure Learning (GPL) framework. GPL consists of UnityGraph which creates a unified representation for multiple videos of the same task. UnityGraph allows us to model both temporal and spatial information. The results obtained and the ablation performed demonstrates the capability of a graph-based approach for procedure learning.

Chapter 5

Conclusion

In this thesis, we proposed and thoroughly investigated two approaches for unsupervised procedure learning. We looked at the literature, identified the gaps, and proposed solutions to fill those gaps. Here is a summary of the thesis.

In Chapter 2, we look at existing third-person datasets for procedure learning and highlight their shortcomings. We highlight the efficacy of first-person videos and propose the EgoProceL dataset to circumvent the shortcomings. Furthermore, we describe various protocols and annotation techniques followed to create the dataset. Finally, we provide various statistics to provide a deeper insight into the dataset and highlight its utility for procedure learning and other computer vision tasks.

In Chapter 3, we highlight the challenges posed by first-person videos for procedure learning and why previous approaches will fall short of performing them. To overcome the challenges, we propose to use the signal provided by the temporal correspondences between key-steps across videos. To this end, we propose the Correspond and Cut (CnC) framework consisting of TC3I loss to learn the embedding space and the ProCut Module (PCM) to identify and cluster the key-steps. Our results on the ProceL and CrossTask datasets highlight the efficacy of using correspondences over state-of-the-art techniques. Furthermore, we demonstrate the efficacy of using the egocentric view for procedure learning.

In Chapter 4, contrary to previous works that generate the learning signal from one or two videos, we focus on creating a representation capable of uniting all the videos of the same task. To this end, we propose the Graph-based Procedure Learning (GPL) framework. GPL consists of UnityGraph, which creates a unified representation for multiple videos of the same task. Due to this, UnityGraph allows the modelling of both temporal and spatial information. Furthermore, we perform an extensive ablation study on various hyper-parameters for UnityGraph and report state-of-the-art results on ProceL, CrossTask, and EgoProceL.

To conclude, in this thesis, we highlight multiple shortcomings of existing works and propose various approaches to overcome them. We hope this work will motivate future egocentric vision-based procedure learning research. Also, we are excited to see more un-/self-supervised approaches to tackle procedure learning.

Related Publications

- **My View is The Best View: Procedure Learning from Egocentric Videos**, Siddhant Bansal, Chetan Arora, C.V. Jawahar. *In IEEE/CVF European Conference on Computer Vision (ECCV) 2022.*

- **United We Stand, Divided We Fall: UnityGraph for Unsupervised Procedure Learning from Videos**, Siddhant Bansal, Chetan Arora, C.V. Jawahar. *In IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2024.*

Other publications:

- **Ego4D: Around the World in 3,000 Hours of Egocentric Video**, 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 Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong 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 Gonzalez, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jachym Kolar, 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 Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Yunyi Zhu, Pablo Arbelaez, David Crandall, Dima Damen, Giovanni Maria Farinella, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, Jitendra Malik. *In Conference on Computer Vision and Pattern Recognition (CVPR) 2022.*

- **Improving Word Recognition using Multiple Hypotheses and Deep Embeddings**, Siddhant Bansal, Praveen Krishnan, C.V. Jawahar. *In International Conference on Pattern Recognition (ICPR), 2021.*

- **Fused Text Recogniser and Deep Embeddings Improve Word Recognition and Retrieval**, *Siddhant Bansal*, Praveen Krishnan, C.V. Jawahar. *In IAPR International Workshop on Document Analysis and System (DAS), 2020.*

Bibliography

[1] U. Ahsan, C. Sun, and I. Essa. DiscrimNet: Semi-Supervised Action Recognition from Videos using Generative Adversarial Networks. In *Computer Vision and Pattern Recognition Workshops (CVPRW) 'Women in Computer Vision (WiCV)'*, 2018.

[2] J.-B. Alayrac, P. Bojanowski, N. Agrawal, I. Laptev, J. Sivic, and S. Lacoste-Julien. Unsupervised learning from Narrated Instruction Videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] S. Bansal, C. Arora, and C. Jawahar. My View is the Best View: Procedure Learning from Egocentric Videos. In *European Conference on Computer Vision (ECCV)*, 2022.

[4] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly Supervised Action Labeling in Videos under Ordering Constraints. In *European Conference on Computer Vision (ECCV)*, 2014.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[6] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi. Domain Generalization by Solving Jigsaw Puzzles. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[7] J. Carreira and A. Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] C.-Y. Chang, D.-A. Huang, Y. Sui, L. Fei-Fei, and J. C. Niebles. D3TW: Discriminative Differentiable Dynamic Time Warping for Weakly Supervised Action Alignment and Segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[9] R. W. Conners and C. A. Harlow. A Theoretical Comparison of Texture Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1980.

[10] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling Egocentric Vision: The EPIC-KITCHENS Dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[11] D. Damen, T. Leelasawassuk, O. Haines, A. Calway, and W. Mayol-Cuevas. You-Do, I-Learn: Discovering Task Relevant Objects and their Modes of Interaction from Multi-User Egocentric Video. In *British Machine Vision Conference (BMVC)*, 2014.

[12] F. De La Torre, J. Hodgins, A. Bargteil, X. Martin, J. Macey, A. Collado, and P. Beltran. Guide to the Carnegie Mellon University Multimodal Activity (CMU-MMAC) database. In *Robotics Institute*, 2008.

[13] A. Diba, V. Sharma, L. Gool, and R. Stiefelhagen. DynamoNet: Dynamic Action and Motion Network. In *International Conference on Computer Vision (ICCV)*, 2019.

[14] L. Ding and C. Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[15] L. Ding and C. Xu. Weakly-Supervised Action Segmentation with Iterative Soft Boundary Assignment. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[16] H. Doughty, I. Laptev, W. Mayol-Cuevas, and D. Damen. Action Modifiers: Learning From Adverbs in Instructional Videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[17] J. C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, 1973.

[18] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. Temporal Cycle-Consistency Learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[19] E. Elhamifar and D. Huynh. Self-supervised Multi-task Procedure Learning from Instructional Videos. In *European Conference on Computer Vision (ECCV)*, 2020.

[20] E. Elhamifar and Z. Naing. Unsupervised Procedure Learning via Joint Dynamic Summarization. In *International Conference on Computer Vision (ICCV)*, 2019.

[21] K. G. *et al*. Ego4D: Around the World in 3,000 Hours of Egocentric Video, 2021.

[22] Z. Feng, C. Xu, and D. Tao. Self-Supervised Representation Learning by Rotation Feature Decoupling. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[23] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-Supervised Video Representation Learning with Odd-One-Out Networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] D. Fried, J.-B. Alayrac, P. Blunsom, C. Dyer, S. Clark, and A. Nematzadeh. Learning to Segment Actions from Observation and Narration. In *Association for Computational Linguistics (ACL)*, 2020.

[25] A. Furnari and G. Farinella. Rolling-Unrolling LSTMs for Action Anticipation from First-Person Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[26] D. Greig, B. Porteous, and A. Seheult. Exact Maximum A Posteriori Estimation for Binary Images. *Journal of the Royal Statistical Society Series B-Methodology*, 1989.

[27] A. Grover and J. Leskovec. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[28] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, 2008.

[29] T. Han, W. Xie, and A. Zisserman. Video Representation Learning by Dense Predictive Coding. In *Workshop on Large Scale Holistic Video Understanding, ICCV*, 2019.

[30] S. Haresh, S. Kumar, H. Coskun, S. N. Syed, A. Konin, Z. Zia, and Q.-H. Tran. Learning by Aligning Videos in Time. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[31] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[32] G. E. Hinton and R. S. Zemel. Autoencoders, Minimum Description Length and Helmholtz Free Energy. In *Neural Information Processing Systems*, 1993.

[33] D.-A. Huang, L. Fei-Fei, and J. C. Niebles. Connectionist Temporal Modeling for Weakly Supervised Action Labeling. In *European Conference on Computer Vision (ECCV)*, 2016.

[34] Y. Huang, M. Cai, Z. Li, and Y. Sato. Predicting gaze in egocentric video by learning task-dependent attention transition. In *European Conference on Computer Vision (ECCV)*, 2018.

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

[36] Y. Jang, B. Sullivan, C. Ludwig, I. Gilchrist, D. Damen, and W. Mayol-Cuevas. EPIC-Tent: An Egocentric Video Dataset for Camping Tent Assembly. In *International Conference on Computer Vision (ICCV) Workshops*, 2019.

[37] L. Ji, C. Wu, D. Zhou, K. Yan, E. Cui, X. Chen, and N. Duan. Learning Temporal Video Procedure Segmentation From an Automatically Collected Large Dataset. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[38] D. Kim, D. Cho, and I.-S. Kweon. Self-Supervised Video Representation Learning with Space-Time Cubic Puzzles. In *AAAI Conference on Artificial Intelligence*, 2019.

[39] D. Kim, D. Cho, D. Yoo, and I.-S. Kweon. Learning Image Representations by Completing Damaged Jigsaw Puzzles. In *Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[40] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations, (ICLR)*, 2015.

[41] N. Komodakis and S. Gidaris. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations (ICLR)*, 2018.

[42] H. Kuehne, A. B. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[43] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 1955.

[44] A. Kukleva, H. Kuehne, F. Sener, and J. Gall. Unsupervised learning of action classes with continuous temporal embedding. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[45] G. Larsson, M. Maire, and G. Shakhnarovich. Learning Representations for Automatic Colorization. In *European Conference on Computer Vision (ECCV)*, 2016.

[46] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a Proxy Task for Visual Understanding. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[47] H.-Y. Lee, J.-B. Huang, M. K. Singh, and M.-H. Yang. Unsupervised Representation Learning by Sorting Sequences. In *International Conference on Computer Vision (ICCV)*, 2017.

[48] J. Li, P. Lei, and S. Todorovic. Weakly Supervised Energy-Based Learning for Action Segmentation. In *International Conference on Computer Vision (ICCV)*, 2019.

[49] J. Li and S. Todorovic. Set-Constrained Viterbi for Set-Supervised Action Segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[50] Y. Li, A. Fathi, and J. M. Rehg. Learning to Predict Gaze in Egocentric Video. In *International Conference on Computer Vision (ICCV)*, 2013.

[51] Y. Li, M. Liu, and J. M. Rehg. In the Eye of Beholder: Joint Learning of Gaze and Actions in First Person Video. In *European Conference on Computer Vision (ECCV)*, 2018.

[52] X. Liu, J. van de Weijer, and A. D. Bagdanov. Leveraging Unlabeled Data for Crowd Counting by Learning to Rank. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[53] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 1982.

[54] J. Malmaud, J. Huang, V. Rathod, N. Johnston, A. Rabinovich, and K. Murphy. What's Cookin'? Interpreting Cooking Videos using Text, Speech and Vision. In *HLT-NAACL*, 2015.

[55] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In *International Conference on Computer Vision (ICCV)*, 2019.

[56] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and Learn: Unsupervised Learning Using Temporal Order Verification. In *European Conference on Computer Vision (ECCV)*, 2016.

[57] Z. Naing and E. Elhamifar. Procedure Completion by Learning from Partial Summaries. In *British Machine Vision Conference (BMVC)*, 2020.

[58] E. Ng, D. Xiang, H. Joo, and K. Grauman. You2Me: Inferring Body Pose in Egocentric Video via First and Second Person Interactions. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[59] M. Noroozi, H. Pirsiavash, and P. Favaro. Representation Learning by Learning to Count. In *International Conference on Computer Vision (ICCV)*, 2017.

[60] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Neural Information Processing Systems*, 2019.

[61] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[62] F. Ragusa, A. Furnari, S. Livatino, and G. M. Farinella. The MECCANO Dataset: Understanding Human-Object Interactions From Egocentric Videos in an Industrial-Like Domain. In *Winter Conference on Applications of Computer Vision (WACV)*, pages 1569–1578, 2021.

[63] A. Richard, H. Kuehne, and J. Gall. Action Sets: Weakly Supervised Action Segmentation Without Ordering Constraints. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[64] A. Richard, H. Kuehne, A. Iqbal, and J. Gall. NeuralNetwork-Viterbi: A Framework for Weakly Supervised Video Learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[65] F. Sener and A. Yao. Zero-Shot Anticipation for Instructional Activities. In *International Conference on Computer Vision (ICCV)*, 2019.

[66] O. Sener, A. R. Zamir, S. Savarese, and A. Saxena. Unsupervised Semantic Parsing of Video Collections. In *International Conference on Computer Vision (ICCV)*, 2015.

[67] D. Shan, J. Geng, M. Shu, and D. Fouhey. Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[68] Y. Shen, L. Wang, and E. Elhamifar. Learning To Segment Actions From Visual and Language Instructions via Differentiable Weak Sequence Alignment. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[69] G. A. Sigurdsson, A. Gupta, C. Schmid, A. Farhadi, and K. Alahari. Actor and Observer: Joint Modeling of First and Third-Person Videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[70] S. Singh, C. Arora, and C. V. Jawahar. First Person Action Recognition Using Deep Learned Descriptors. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[71] E. V. . C. software]. (2020). *Nijmegen: Max Planck Institute for Psycholinguistics, The Language Archive.* Retrieved from https://archive.mpi.nl/tla/elan.

[72] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised Learning of Video Representations Using LSTMs. In *International Conference on Machine Learning (ICML)*, 2015.

[73] Y. Tang, D. Ding, Y. Rao, Y. Zheng, D. Zhang, L. Zhao, J. Lu, and J. Zhou. COIN: A Large-Scale Dataset for Comprehensive Instructional Video Analysis. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[74] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *International Conference on Computer Vision (ICCV)*, 2015.

[75] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[76] L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 2008.

[77] R. G. VidalMata, W. J. Scheirer, A. Kukleva, D. Cox, and H. Kuehne. Joint Visual-Temporal Embedding for Unsupervised Learning of Actions in Untrimmed Sequences. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[78] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *International Conference on Machine Learning (ICML)*, 2008.

[79] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating Videos with Scene Dynamics. In *Neural Information Processing Systems*, 2016.

[80] X. Wang, R. B. Girshick, A. Gupta, and K. He. Non-local Neural Networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[81] D. Wei, J. Lim, A. Zisserman, and W. T. Freeman. Learning and Using the Arrow of Time. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[82] J. woo Choi, G. Sharma, S. Schulter, and J.-B. Huang. Shuffle and Attend: Video Domain Adaptation. In *European Conference on Computer Vision (ECCV)*, 2020.

[83] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, and Y. Zhuang. Self-Supervised Spatiotemporal Learning via Video Clip Order Prediction. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[84] S.-I. Yu, L. Jiang, and A. Hauptmann. Instructional Videos for Unsupervised Harvesting and Learning of Action Examples. In *ACM International Conference on Multimedia*, 2014.

[85] L. Zhou, C. Xu, and J. J. Corso. Towards Automatic Learning of Procedures From Web Instructional Videos. In *AAAI Conference on Artificial Intelligence*, 2018.

[86] D. Zhukov, J.-B. Alayrac, R. G. Cinbis, D. Fouhey, I. Laptev, and J. Sivic. Cross-task weakly supervised learning from instructional videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[87] D. Zhukov, J.-B. Alayrac, I. Laptev, and J. Sivic. Learning Actionness via Long-range Temporal Order Verification. In *European Conference on Computer Vision (ECCV)*, 2020.