Unsupervised Learning of Procedures from Demonstration Videos

Karan Goel
Machine Learning Department
Carnegie Mellon University
kgoel93@gmail.com

Emma Brunskill
Computer Science Department
Stanford University
ebrun@cs.stanford.edu

Abstract

Video demonstrations are a rich source of information about the procedural knowledge required to execute a task. In this work, we introduce the problem of identifying a common procedure underlying video demonstrations of a task, without using any supervision. The procedure corresponds to a sequence of primitive behaviors that must be executed to complete the task. Procedure identification is useful for a variety of applications including intelligent tutoring systems, representation learning, transfer learning and video understanding. We connect this procedure identification problem to that of temporal clustering and latent variable modeling, and develop a general pipeline for procedure extraction. We also identify issues with evaluation criteria used for temporal clustering, and propose new metrics that address them. Our methods and metrics are evaluated on a dataset of YouTube videos demonstrating Bach's Prelude in C-major.

1 Introduction

In the cognitive sciences, *procedural knowledge* is defined as knowledge that aids in the production of behaviors in order to accomplish some goal [1]. Along with *declarative knowledge*, it forms the two main types of knowledge, and is the one primarily responsible for the successful completion of tasks and goals. One of the building blocks for the acquisition of procedural knowledge is observation, especially of expert demonstrations [1]. Research from cognitive apprenticeship [2, 3] suggests that observing and assimilating information from expert demonstrations is a key step in building a conceptual model for the task to be performed. Concretely, demonstrations can be used to extract useful procedural information about how to accomplish a task such as the objects/behaviors required to be learned and the sequence of steps to be executed.

Traditionally, procedural knowledge acquisition has been guided by expert humans, who identify the primitives in the task being learned and then demonstrate them – *e.g.* a human math teacher describing the steps required to solve a system of linear equations. Intelligent tutoring systems have made a systematic effort to automate the knowledge acquisition process, with models of knowledge acquisition such as the Bayesian Knowledge Tracing model [4] being used to automatically teach a learner. However, these models and systems continue to leverage human-designed curricula to teach procedural knowledge.

More recently, the popularity of Massive Open Online Courses (MOOCs) has made videos a widely used resource in 'flipped' classroom settings. Studies [5, 6] suggest that videos can be an effective way to teach procedural knowledge, as long as the content being taught is appropriately chunked. Beyond the context of education, videos remain a rich source of data to model, understand and learn from human behavior. Websites such as YouTube have a vast collection of videos of humans participating in and demonstrating a multitude of activities. Datasets leveraging these video sources

CMU Machine Learning Data Analysis Project, 2018.

have recently been used in computer vision problems such as activity recognition and localization [7] and video understanding [8].

In this work we examine whether it is possible to generate procedural abstractions automatically from real-world video demonstrations, without additional human supervision. We identify the problem of automatically learning a procedural abstraction from a set of video demonstrations, and provide methods to learn this procedure. Outside of the demonstrations themselves, we eschew other forms of labeled supervision since it can be expensive to label and collect such data.

From a computational perspective, one of the main goals of Artificial Intelligence (AI) is the creation of conceptual representations that are useful for making decisions. Goal-directed intelligent agents must have an abstract understanding of the behaviors that they execute and the world around them, to allow decision making based on predicates rather than pixels. Finding rich, distributed representations for predictive tasks has been central to the success of neural networks [9]. However, creating interpretable and useful abstractions for decision-making agents remains an open and active area of research.

Automatic generation of the procedure encapsulated in a set of video demonstrations gives us the ability to create a succinct representation of the task. This low-dimensional task representation can be used to determine how different tasks relate to each other and what primitives are necessary for a task. Recent work has explored the possibility of learning agent policies directly from video demonstrations [10–12] using reinforcement learning. Learning a succinct task representation can aid these efforts in quickly establishing the viability of transfer learning between tasks, by comparing their task representations. This can further lead to more sample-efficient learning for cases where tasks are found to be related *e.g.* by learning a hierarchical policy over the task primitives, as done in [13].

Formally, given a set of video demonstrations of a task, our goal is to identify a procedure that can be used to decompose each video into a sequence of steps. The procedure can be thought of as a sequence of tokens, each token representing a primitive behavior or building block for the task. Aside from the lack of supervision in the form of labeled data, this problem is challenging due to several reasons. There exist several possible 'correct' procedures for any task, differing only in what they consider primitives. If the task is to play a musical piece, one procedure may consider individual notes as steps, while another may divide the piece up into groups of notes. This problem is closely related to past work in latent variable modeling [14–16] and temporal clustering [17, 18] and inherits their complexity.

What makes our setting more challenging is that we use high-dimensional video data, while most time-series work has operated on low-dimensional sequences [19]. Each video may be shot in a visually distinctive setting, with variations in the surroundings, viewpoint, lighting and even the objects used for the task. These demonstrations will almost certainly not be aligned in time, so different demonstrations may spend different amounts of time on the same behavior primitives. A simple example is of videos of two pianists playing the same piece – they will prefer different tempi and may utilize different amounts of rubato. One may be a concert pianist playing on a large Steinway grand piano in a concert hall, while the other an amateur on a small upright piano in their apartment. Some tasks may also contain 'repeated structure', where even within a video, behavior primitives are repeated in different contexts. Identifying this repeated structure can be very useful, since it yields a simpler program where a single primitive is reused multiple times over the course of the task.

We address some of these issues in this work and lay out the challenges that need to be overcome for those issues that remain unresolved. Our contributions are,

- We introduce the problem of learning a procedure from a set of video demonstrations without supervision and provide a general pipeline to solve this problem for video data. This problem has important applications in tutoring systems, representation learning, transfer learning and video understanding.
- We identify a gap in the evaluation criteria used for methods in temporal clustering, namely that they ignore temporal structure, and propose new metrics that address this.
- We evaluate our methods on a new dataset of videos collected from YouTube, for the task of learning Bach's Prelude in C-major. The evaluation verifies our new metrics and demonstrates that our methods can identify high-quality coarse procedures.

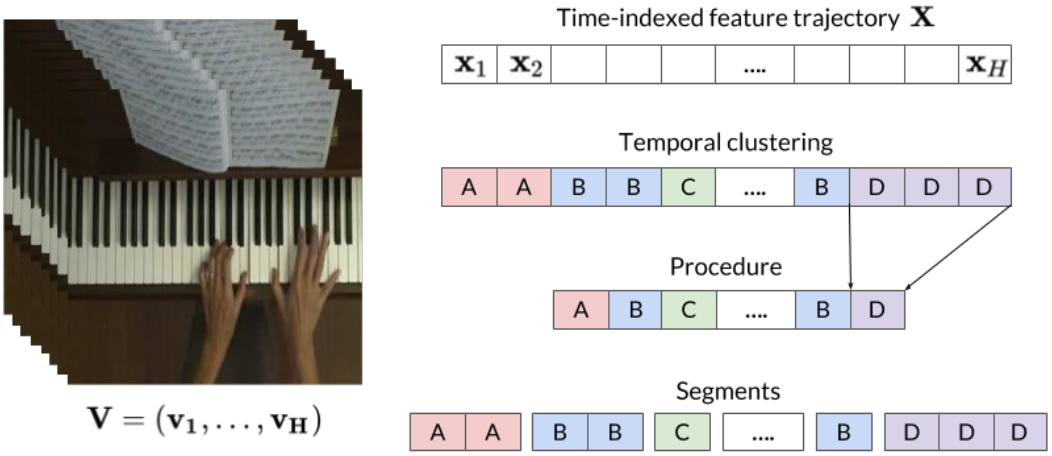


Figure 1: Illustration of definitions in Section 2. A temporal clustering and its corresponding procedure are shown for an example video.

2 Definitions

A video $\mathbf{V}^i = (\mathbf{v}_1^i, \mathbf{v}_2^i, \ldots, \mathbf{v}_{H_i}^i)$ is a sequence of RGB frames $\mathbf{v}_t^i \in \mathbb{R}^{h \times w \times 3}$ of height h, width w and total number of frames H_i. The dataset of videos is $\mathcal{D} = \{\mathbf{V}^1, \ldots, \mathbf{V}^n\}$ where $n \geq 1$.

We will represent each video as a time-indexed feature trajectory $\mathbf{X}^i = (\mathbf{x}_1^i, \mathbf{x}_2^i, \ldots, \mathbf{x}_{H_i}^i)$ with $\mathbf{x}_t^i \in \mathbb{R}^d$. How this representation is derived from $\mathbf{V}^i$ will be discussed in the next section.

Temporal Clustering. A *temporal clustering* $\mathcal{C} = (c_1, \ldots, c_H)$ is a sequence of cluster labels where $c_t \in \Gamma$, where Γ is a set of cluster labels. Temporal clusterings map each item in the feature trajectory $\mathbf{X}$ to a cluster label in Γ.

We will use $\mathcal{C}^i$ as shorthand to denote the temporal clustering generated by some method for trajectory $\mathbf{X}^i$. The ground truth temporal clustering for $\mathbf{X}^i$ will be denoted by $\mathcal{G}^i$. For simplicity, we will define $\mathcal{C}_{a:b} = (c_a, c_{a+1}, \ldots, c_b)$ where $1 \leq a \leq b \leq H$ are time indices.

Lastly, for $\alpha \in \Gamma$ we let $\mathcal{C}[\alpha] = \{t | c_t = \alpha, 1 \leq t \leq H\}$ be the set of time indices in $\mathcal{C}$ whose cluster labels equal α. $\mathcal{C}[\alpha]$ is the cluster (in the traditional sense) corresponding to α.

Procedure. Let $\mathcal{P}$ be a function that removes running duplicates from a sequence of labels, yielding a sequence of *tokens*. For instance, $\mathcal{P}(\mathrm{A}, \mathrm{A}, \mathrm{A}, \mathrm{B}, \mathrm{B}, \mathrm{B}, \mathrm{C}, \mathrm{C}, \mathrm{C}, \mathrm{A}, \mathrm{A}, \mathrm{B}) = (\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{A}, \mathrm{B})$ yielding 5 tokens. For a temporal clustering $\mathcal{C}$, we define its corresponding *procedure* to be $\mathcal{P}(\mathcal{C})$. The procedure captures the cluster labels (as tokens) present in the temporal clustering and the sequence in which they occur.

Every token in the procedure signifies a primitive behavior required to carry out the task. We say that the procedure exhibits *repeated structure* if atleast one cluster label is reused in the procedure *i.e.* $|\mathcal{P}(\mathcal{C})| > |\Gamma|$ (the number of tokens exceeds the number of clusters). The example in the previous paragraph contains repeated structure since both cluster label A and cluster label B are reused in the procedure.

Segment. Each token in the procedure $\mathcal{P}(\mathcal{C})$ is associated with a *segment* in $\mathcal{C}$ – denoting the time boundaries of the token in the temporal clustering. We represent each segment as a pair of time indices (a, b), where a represents the start time-index of the segment while b the end time-index.

We let $\mathcal{S}(\mathcal{C}, \alpha) = \{(a, b) | \mathcal{C}_{a:b} = (c_a = \alpha, c_{a+1} = \alpha, \ldots, c_b = \alpha), 1 \leq a \leq b \leq H\}$ be a function that maps a temporal clustering $\mathcal{C}$ and cluster label $\alpha \in \Gamma$ to the set of segments associated with α in $\mathcal{C}$.

Figure 1 shows an example that illustrates these ideas. In the example, the temporal clustering is defined over the label set $\Gamma = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$. It associates each time step in the trajectory with a cluster label. The procedure extracted from this temporal clustering is shown below it. The arrows demonstrate how the purple D cluster labels are coalesced into a single token of the procedure. The

segment corresponding to this token (D) is then simply $(H-2, H)$. Notice that the procedure exhibits repeated structure since the blue B label has atleast 2 tokens/segments corresponding to it.

In our setting, we assume that a common procedure underlies each video demonstration. We characterize this more precisely below, before describing our problem statement,

Common Underlying Procedure. Each video $\mathbf{V}^i \in \mathcal{D}$ has an associated underlying procedure $\mathcal{P}^i$ such that $\mathcal{P}^1 = \mathcal{P}^2 = \cdots = \mathcal{P}^n$. This implies that each video is a demonstration that realizes a common underlying procedure.

Satisfying this assumption requires us to curate a dataset for which the above holds. This is typically referred to as a trimmed video dataset in the computer vision community. In practice, for large n (~ 10) this requirement is less stringent, and we can allow videos that are less precisely trimmed. Relaxing this assumption completely would permit the possibility that no two videos in the dataset share a procedure – this moves closer in spirit to work on temporal clustering for sequences. However, this body of work instead makes strong assumptions about how the sequences themselves are generated or relate to each other, which we avoid. We can now define our problem statement,

Problem Statement. Given a dataset of n video demonstrations, identify a temporal clustering $\mathcal{C}^i$ for each video, such that $\mathcal{P}(C^1) = \mathcal{P}(C^2) = \cdots = \mathcal{P}(C^n)$ *i.e.* these temporal clusterings share a single procedure.

Our goal is thus to identify the common procedure that underlies each of the demonstrations in the dataset, as well as a temporal clustering for each demonstration that respects the procedure. One of the advantages of extracting a procedure is that each token represents a small step that must be accomplished to do the task, and identifying the procedure gives us a recipe for the task. Each token will represent a behavior primitive or share some other commonality, which can be identified and associated with a semantic meaning if so desired.

Identifying the temporal clustering in addition to the procedure gives us an easy way to evaluate the quality of the identified procedure, by comparing to a ground truth temporal clustering for each video. It is also generally advantageous as the temporal clustering allows us to access the segments in each video corresponding to a token in the procedure. This is useful for building models to identify the token behavior in other settings.

Depending on how coarse or fine we want the procedure to be, we could end up with very different answers to this problem. This reflects the fact that the same task can be described in several ways, an inherent uncertainty in the problem that cannot be overcome without having additional information.

As an example, consider the case of experts demonstrating how to play a single bar of a piano piece. We could recover 2 distinct procedures, $\tilde{\mathcal{P}}_1 = (1), \tilde{\mathcal{P}}_2 = (1, 2, 3, 1)$ such that,

$\tilde{\mathcal{P}}_1$

1: Play the bar.

$\tilde{\mathcal{P}}_2$

1: Play notes CECGE in the right hand.
2: Play notes CFDFG in the right hand.
3: Play notes CDEFG in the right hand.

Both procedures are correct, but $\tilde{\mathcal{P}}_1$ is clearly very coarse and contains no new information about the task, whereas $\tilde{\mathcal{P}}_2$ is more fine-grained and exhibits some repeated structure. While $\tilde{\mathcal{P}}_2$ is clearly the superior choice here, choosing between competing procedures (or temporal clusterings) without access to a ground truth is a difficult problem, and will not be the focus of this work.[1] There are some general principles that we could follow, such as (i) prioritizing procedures that exhibit more repeated structure; (ii) preferring procedures whose segments are not too uneven in length; (iii) preferring procedures that don't have too many tokens. Making these notions precise is left to future work.

[1] A metric that is often used for this purpose in the standard clustering setting is the *silhouette score*, which measures the tightness of clusterings to decide the best one.

3 A General Pipeline

There are several problems that must be addressed by any method that does procedure extraction. Stated quite generally these are,

- *Feature extraction* or finding a good feature representation for the videos
- *Time warping* or aligning the videos to facilitate aggregation of information
- *Procedure identification* or finding a procedure common to the videos

We address each of these in turn.

3.1 Feature Extraction

The first step in our pipeline is to extract a lower dimensional representation of the video. Using the video directly is difficult, since each frame is extremely high dimensional. Video frames are also not translation/rotation invariant feature spaces and are generally considered unsuitable for use directly [20].

Since our pipeline is completely unsupervised, we opt to use a VGG-19 model [21] pre-trained on ImageNet as a feature extractor. Each frame $\mathbf{v}_t^i$ of video $\mathbf{V}^i$ is mapped to a 4096-dimensional representation $\mathbf{x}_t^i$ using the VGG-19 model's fc6 layer.

Note that it is possible to substitute other methods for feature extraction, including feature extractors that are sensitive to the motion in the video. However, there is no widely accepted feature extractor for video data, so we opted to extract features on a frame-by-frame basis instead.

3.2 Time Warping & Alignment

An important challenge that we must address is how to aggregate information across the feature trajectories $\mathbf{X}^i$ that we extracted for the videos in our dataset. This aggregation is non-trivial since each video may carry out the demonstration at very different rates. *E.g.* two pianists playing the same piece at different tempi. One of the ways we can do this alignment is by time warping each feature trajectory to a common length, so that they all align.

Time warping is a general method to align two (or more) sequences, such that an alignment cost is minimized. There are several variants of time warping such as dynamic time warping [22], canonical time warping [23, 24], etc. However, these methods have generally been used on low-dimensional time-series data, and we found their performance to be poor when used to align the $\mathbf{X}^i$s. Given this challenge, we will restrict ourselves to video datasets that we expect will approximately respect the following assumption.

Uniform Time Warping Assumption. We assume that each video has an underlying temporal clustering $\mathcal{G}^i$ such that for any pair i, j of videos, $\mathcal{G}_t^i = \mathcal{G}_{t'}^j$ where $t' = t \cdot \frac{H_j}{H_i}$. This implies that all the trajectories can be aligned by simply uniformly stretching or shrinking them to the same length. In practice, it is sufficient for this assumption to hold approximately.

We let $\tau(\cdot, L)$ be the uniform time warping function, which takes an arbitrary sequence as its first argument and stretches/shrinks it uniformly to length L. τ can be used on feature trajectories $\mathbf{X}^i$ and temporal clusterings $\mathcal{C}^i$ or $\mathcal{G}^i$.

3.3 Procedure Identification

We now outline 2 general methods for procedure identification. The first, *local temporal clustering and aggregation* finds a temporal clustering for each $\mathbf{X}^i$ separately before aggregating these temporal clusterings to extract a procedure. The second, *global procedure extraction*, first constructs a single trajectory that aggregates information from all the videos, before directly extracting a procedure.

We assume access to some temporal clustering algorithm $\mathcal{A}$, which takes as input a feature trajectory $\mathbf{X}$, number of clusters k and outputs a temporal clustering $\mathcal{C} = \mathcal{A}(\mathbf{X}, k)$. $\mathcal{A}_{\text{hamming}}$ will be used to denote that the temporal clustering method uses the Hamming distance metric; otherwise we assume Euclidean distance is the default.

Algorithm 1 Local Temporal Clustering and Aggregation

Require: Temporal clustering method $\mathcal{A}$, feature trajectory dataset $\mathbf{X}^1, \ldots, \mathbf{X}^n$, warp length L, number of clusters k

1: **function** LTCA($\mathcal{A}, \mathbf{X}^1, \ldots, \mathbf{X}^n, L, k$)
2: **for all** $i \in [n]$ **do**
3: $\mathcal{C}^i \leftarrow \tau(\mathcal{A}(\mathbf{X}^i, k), L)$ ▷ Local temporal clustering and warping
4: **end for**
5: $\mathcal{T} \leftarrow \mathbf{S}(\mathcal{C}^1, \ldots, \mathcal{C}^n)$ ▷ Meta-trajectory construction
6: $\mathcal{C} \leftarrow \mathcal{A}_{\text{hamming}}(\mathcal{T}, k)$ ▷ Temporal clustering of meta-trajectory with Hamming distance
7: **for all** $i \in [n]$ **do**
8: $\tilde{\mathcal{C}}^i \leftarrow \tau(\mathcal{C}, H_i)$ ▷ Warp back for each temporal clustering
9: **end for**
10: **return** procedure $\mathcal{P}(\mathcal{C})$, temporal clusterings $\tilde{\mathcal{C}}^1, \ldots, \tilde{\mathcal{C}}^n$
11: **end function**

Both methods also require us to introduce the notion of a meta-trajectory, which we do first. We then go on to describe each method in turn.

Meta-Trajectory. Let $\mathbf{Z}^i = (\mathbf{z}_1^i, \mathbf{z}_2^i, \ldots, \mathbf{z}_L^i)$ be n trajectories of length L with $\mathbf{z}_t^i \in \mathbb{R}^p$. Define the meta-feature at $1 \leq t \leq L$ as,

$$\mathbf{s}_t = (\mathbf{z}_t^1, \mathbf{z}_t^2, \ldots, \mathbf{z}_t^n)$$

where $\mathbf{s}_t \in \mathbb{R}^{np}$. The meta-feature at t concatenates features from all trajectories at time point t. The meta-trajectory $\mathbf{S}$ over $(\mathbf{Z}^1, \ldots, \mathbf{Z}^n)$ is then defined as,

$$\mathbf{S}(\mathbf{Z}^1, \ldots, \mathbf{Z}^n) = (\mathbf{s}_1, \ldots, \mathbf{s}_L)$$

3.3.1 Local Temporal Clustering and Aggregation (LTCA)

Our first method is outlined in Algorithm 1. First, a temporal clustering is performed on each feature trajectory $\mathbf{X}^i$ using $\mathcal{A}$. Then this clustering $\mathcal{A}(\mathbf{X}^i, k)$ is uniformly warped using τ to a fixed length L, yielding a temporal clustering $\mathcal{C}^i$ of length L (lines $2-4$). Notice that it is not necessary here that $\mathcal{P}(\mathcal{C}^1) = \mathcal{P}(\mathcal{C}^2) = \cdots = \mathcal{P}(\mathcal{C}^n)$. In fact, it will almost certainly not be the case that this happens. This is due to a couple of reasons: (i) each video demonstration $\mathbf{V}^i$ is a noisy realization of the underlying procedure, with varied viewpoint, background and setting; (ii) the feature trajectories $\mathbf{X}^i$ determine the quality of the temporal clusterings extracted. In some cases, the features may not be informative enough to recognize repeated structure in the demonstration or to correctly delineate different parts of the procedure.

By aggregating over the temporal clusterings $\mathcal{C}^i$, we can reduce the noise that would be present if we were to pick any one of them. In the classical clustering literature, this is referred to as *consensus clustering* or clustering aggregation [25], and is used to aggregate multiple clusterings of the same dataset. In our setting, we instead have n separate temporal clusterings that we have aligned to aggregate.

We take all the temporal clusterings $\mathcal{C}^i$ and construct a single meta-trajectory $\mathcal{T}$ from them (line 5). We then perform a temporal clustering of this meta-trajectory $\mathcal{C} = \mathcal{A}_{\text{hamming}}(\mathcal{T}, k)$ under the Hamming distance metric (line 6). Using the Hamming distance allows us to measure the number of disagreements between any two meta-features *i.e.* how many temporal clusterings put them in different clusters. A similar method for standard non-temporal clusterings was described in [25].

Having found a single consensus clustering $\mathcal{C}$, we can now proceed to apply $\mathcal{C}$ to each video by simply warping it back to yield $\tilde{\mathcal{C}}^i = \tau(\mathcal{C}, H_i)$ (lines $7-9$). $\tilde{\mathcal{C}}^i$ is the identified temporal clustering for the i^{th} video. The overall procedure extracted is simply $\mathcal{P}(\mathcal{C})$.

3.3.2 Global Procedure Extraction (GPE)

Our second method GPE is outlined in Algorithm 2. The main difference from LTCA is that GPE directly creates a single feature trajectory and then clusters it. This avoids the consensus clustering step of LTCA.

Algorithm 2 Global Procedure Extraction

Require: Temporal clustering method $\mathcal{A}$, feature trajectory dataset $\mathbf{X}^1, \ldots, \mathbf{X}^n$, warp length L, number of clusters k

1: **function** GPE($\mathcal{A}, \mathbf{X}^1, \ldots, \mathbf{X}^n, L, k$)
2: **for all** $i \in [n]$ **do**
3: $\tilde{\mathbf{X}}^i \leftarrow \tau(\mathbf{X}^i, L)$ ▷ Warping feature trajectories
4: **end for**
5: $\mathcal{T} \leftarrow \mathbf{S}(\tilde{\mathbf{X}}^1, \ldots, \tilde{\mathbf{X}}^n)$ ▷ Meta-trajectory construction
6: $\mathcal{C} \leftarrow \mathcal{A}(\mathcal{T}, k)$ ▷ Temporal clustering of meta-trajectory
7: **for all** $i \in [n]$ **do**
8: $\tilde{\mathcal{C}}^i \leftarrow \tau(\mathcal{C}, H_i)$ ▷ Warp back for each temporal clustering
9: **end for**
10: **return** procedure $\mathcal{P}(\mathcal{C})$, temporal clusterings $\tilde{\mathcal{C}}^1, \ldots, \tilde{\mathcal{C}}^n$
11: **end function**

For GPE, we first uniformly warp each feature trajectory $\mathbf{X}^i$ to a common length L to yield $\tilde{\mathbf{X}}^i = \tau(\mathbf{X}^i, L)$ (lines $2-4$). Then we directly construct a meta-trajectory $\mathcal{T}$ using all the warped feature trajectories $\tilde{\mathbf{X}}^i$ (line 5) and find a temporal clustering $\mathcal{C} = \mathcal{A}(\mathcal{T}, k)$ for $\mathcal{T}$ (line 6). A consequence of directly clustering this feature meta-trajectory is that the Euclidean distance between any pair of items $\mathbf{s}_a, \mathbf{s}_b \in \mathcal{T}$ can be written as,

$$\|\mathbf{s}_a - \mathbf{s}_b\|_2^2 = \sum_{i=1}^{n} \|\mathbf{x}_a^i - \mathbf{x}_b^i\|_2^2$$

This makes the temporal clustering of the meta-trajectory $\mathcal{T}$ sensitive to distances in the individual $\mathbf{X}^i$'s for GPE. In LTCA on the other hand, the consensus clustering performed on the meta-trajectory was indifferent to distances in the individual $\mathbf{X}^i$'s. For GPE, it is important to ensure that the distance terms $\|\mathbf{x}_a^i - \mathbf{x}_b^i\|_2^2$ are similarly scaled so that no term dominates. Due to this, we standardize each $\mathbf{X}^i$ separately before running GPE.

Now, the last step is similar to LTCA – we apply $\mathcal{C}$ to each video by simply warping it back to yield $\tilde{\mathcal{C}}^i = \tau(\mathcal{C}, H_i)$ (lines $7-9$). $\tilde{\mathcal{C}}^i$ is the identified temporal clustering for the i^{th} video and the overall procedure extracted is $\mathcal{P}(\mathcal{C})$.

3.3.3 Choosing $\mathcal{A}$

Since $\mathcal{A}$ is a general method for finding a temporal clustering, it can be chosen arbitrarily. In fact, it is possible to use any discrete latent variable model for time-series data, such as autoregressive models, hidden markov models, switching linear dynamic systems, etc as well.

For the purpose of this paper, we experiment with two choices for $\mathcal{A}$: (i) *Hierarchical Clustering (HC)* with the Ward linkage criterion. While hierarchical clustering is a standard non-temporal clustering method, we can directly apply it to the feature trajectories to yield a temporal clustering; (ii) *Gaussian Mixture Models (GMM)* which fit a mixture of Gaussians to the feature trajectories. We make a small modification to the feature trajectories $\mathbf{X}^i$ – instead we use $\mathbf{Y}^i = (\mathbf{y}_1^i, \ldots, \mathbf{y}_{H_i}^i)$ where $\mathbf{y}_t^i = (\mathbf{x}_{t-2}^i, \mathbf{x}_{t-1}^i, \mathbf{x}_t^i, \mathbf{x}_{t+1}^i, \mathbf{x}_{t+2}^i)$ is a stack of 5 time-steps. This is similar to the method of [14] and makes the GMM equivalent to fitting a switching linear dynamic system.

4 Evaluation of Temporal Clustering

An important problem to address is the evaluation of the identified procedure. In this work, we evaluate a procedure by comparing the temporal clustering $\mathcal{C}$ generated under it with respect to a ground truth temporal clustering $\mathcal{G}$. In particular, for any metric $\mathcal{M}$ we calculate $\frac{1}{n}\sum_i \mathcal{M}(\mathcal{G}^i, \mathcal{C}^i)$ to evaluate the temporal clusterings using $\mathcal{M}$ over the dataset of videos.

Using traditional clustering metrics for evaluation is the prevalent approach in temporal clustering [18, 17]. Unfortunately, as we will demonstrate shortly, traditional clustering metrics are ill-suited for

the evaluation. The lack of attention to temporal structure means that these metrics often give counter-intuitive results. To remedy this, we introduce 2 new metrics for temporal clustering, *temporal purity* and *temporal completeness*. We also introduce a new combined metric, the *temporal clustering score*, which can be used as a standalone metric to evaluate any temporal clustering.

4.1 Traditional Clustering Metrics

For a temporal clustering $\mathcal{C}$ the widely used *purity*[2][26] metric penalizes the presence of items from different ground truth labels in the same cluster. This is a desirable property that any good clustering evaluation criterion should consider. Purity is defined as,

$$\text{Purity} = \frac{1}{H} \sum_{\alpha \in \Gamma_{\mathcal{C}}} \max_{\beta \in \Gamma_{\mathcal{G}}} |\mathcal{G}[\beta] \cap \mathcal{C}[\alpha]|$$

A related metric is homogeneity [26], which captures the same idea in a different way,

$$\text{Homogeneity} = 1 - \frac{H(\Gamma_{\mathcal{G}}|\Gamma_{\mathcal{C}})}{H(\Gamma_{\mathcal{G}})}$$

where $H(\Gamma_{\mathcal{G}})$ is the entropy of the clustering $\mathcal{G}$, $H(\Gamma_{\mathcal{G}}|\Gamma_{\mathcal{C}})$ is the conditional entropy of $\mathcal{G}$ given $\mathcal{C}$. Intuitively the conditional entropy term peeks inside every cluster in $\mathcal{C}$, and checks the entropy of the items inside in terms of the ground truth labels. The conditional entropy term will be low if every cluster in $\mathcal{C}$ is pure and contains items of only a single ground truth label. This in turn makes the homogeneity high (close to 1.0).

Another important clustering metric is *completeness* [26], which prefers clusterings where all items from a ground-truth label lie in the same cluster *i.e.* the ground truth label is not split across clusters. It is defined as,

$$\text{Completeness} = 1 - \frac{H(\Gamma_{\mathcal{C}}|\Gamma_{\mathcal{G}})}{H(\Gamma_{\mathcal{C}})}$$

Both completeness and homogeneity are considered important criteria for clustering. Maximizing either at the expense of the other is bad – high homogeneity and low completeness imply a clustering that is too fine-grained (*e.g.* every item in its own cluster is perfectly pure and homogeneous), while low homogeneity and high completeness imply a clustering that is too coarse (*e.g.* a single cluster containing all items is perfectly complete). Therefore, several metrics that combine these 2 criterion have been proposed. The *normalized mutual information* (NMI) is a widely used metric, calculated as the geometric mean of homogeneity and completeness [27]. Another popular metric is the *V-measure*, which is calculated as the harmonic mean of homogeneity and completeness instead [26]. These criteria balance homogeneity and completeness for evaluating clusterings.

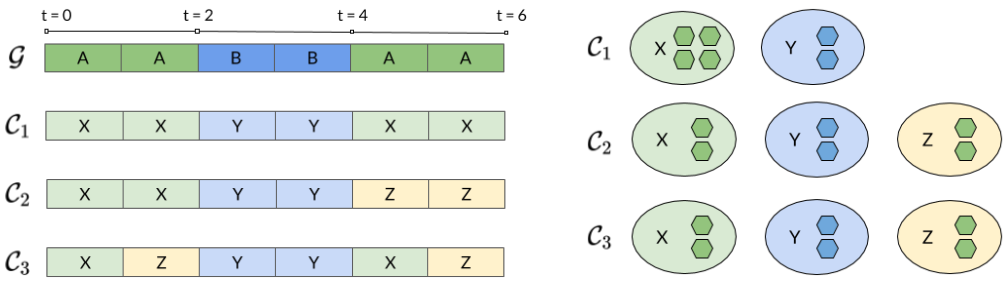
Issues. These metrics can give counter-intuitive results when used for evaluating temporal clusterings. The main issue that distinguishes the temporal clustering case is the presence of repeated structure – ground truth labels that are split across several segments in the trajectory. We desire in the temporal case that repeated segments from the same ground truth label are identified correctly. For instance, in Fig 2, identifying that the dark-green segments $\mathrm{A}_{0:2}, \mathrm{A}_{4:6}$ share the same label A is essential.

To illustrate what would happen if we used the metrics that we outlined above, we provide some examples in Fig 2. Fig 2a contains examples of possible temporal clusterings for a 6-step trajectory, as well as the ground truth. Fig 2b converts these temporal clusterings to their standard non-temporal clustering counterparts.

Firstly, notice that all three temporal clusterings $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ in Fig 2 are assigned a perfect purity or homogeneity of 1.0. The reason is apparent from Fig 2b; no cluster in any of the three clusterings contains items from more than one ground truth label.

However, while $\mathcal{C}_1$ perfectly matches the ground truth there are problems with both $\mathcal{C}_2$ and $\mathcal{C}_3$. $\mathcal{C}_2$ misses the repeated structure in A, splitting the items in A into 2 separate clusters $\mathcal{C}_2[\mathrm{X}]$ and $\mathcal{C}_2[\mathrm{Z}]$ instead. $\mathcal{C}_3$ also splits the items in A into 2 separate clusters $\mathcal{C}_3[\mathrm{X}]$ and $\mathcal{C}_3[\mathrm{Z}]$. However, $\mathcal{C}_3$ captures all the repeated structure in A, with $\mathcal{C}_3[\mathrm{X}_{0:1}\mathrm{Z}_{1:2}]$ corresponding to $\mathrm{A}_{0:2}$ and $\mathcal{C}_3[\mathrm{X}_{4:5}\mathrm{Z}_{5:6}]$ corresponding to $\mathrm{A}_{4:6}$.

[2]While we will define the traditional metrics for the temporal clustering case, they were originally intended for use with standard non-temporal clusterings.



(a) $\mathcal{G}$ is the ground truth temporal clustering and $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ are possible temporal clusterings.

(b) $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ when converted to standard non-temporal clusterings for use with traditional metrics.

Figure 2: Problematic cases for traditional metrics, on a simple 6-step trajectory. $\mathcal{C}_1$ is equivalent to ground truth and preferable to both $\mathcal{C}_2$ and $\mathcal{C}_3$ but all three receive a perfect score on purity and homogeneity. $\mathcal{C}_2$ and $\mathcal{C}_3$ have an identical completeness score but $\mathcal{C}_3$ captures repeated structure and should be preferable to $\mathcal{C}_2$. All traditional clustering metrics based on a contingency matrix rely on the representation in (b) and give $\mathcal{C}_2$ and $\mathcal{C}_3$ an equal score.

While homogeneity failed to distinguish this case, the problem is that completeness also does not help. Both $\mathcal{C}_2$ and $\mathcal{C}_3$ have an identical completeness score, since the clusters in both have exactly the same composition as seen in Fig 2b (in fact no traditional metric such as NMI or V-Measure can distinguish these cases). However, the temporal clusterings for $\mathcal{C}_2$ and $\mathcal{C}_3$ contain useful information which should allow us to pick $\mathcal{C}_3$ over $\mathcal{C}_2$.

The technical problem with all of these metrics is that they rely on proportions, and can be computed using only a contingency matrix. This loses the temporal structure that is important to accurately assess the temporal clusterings. There are several other metrics which we did not discuss here, including the Rand index, the Adjusted Rand index [28], and so on, but they all suffer from the same problem.

We now describe metrics that alleviate these problems in a natural fashion.

4.2 Temporal Purity (TP)

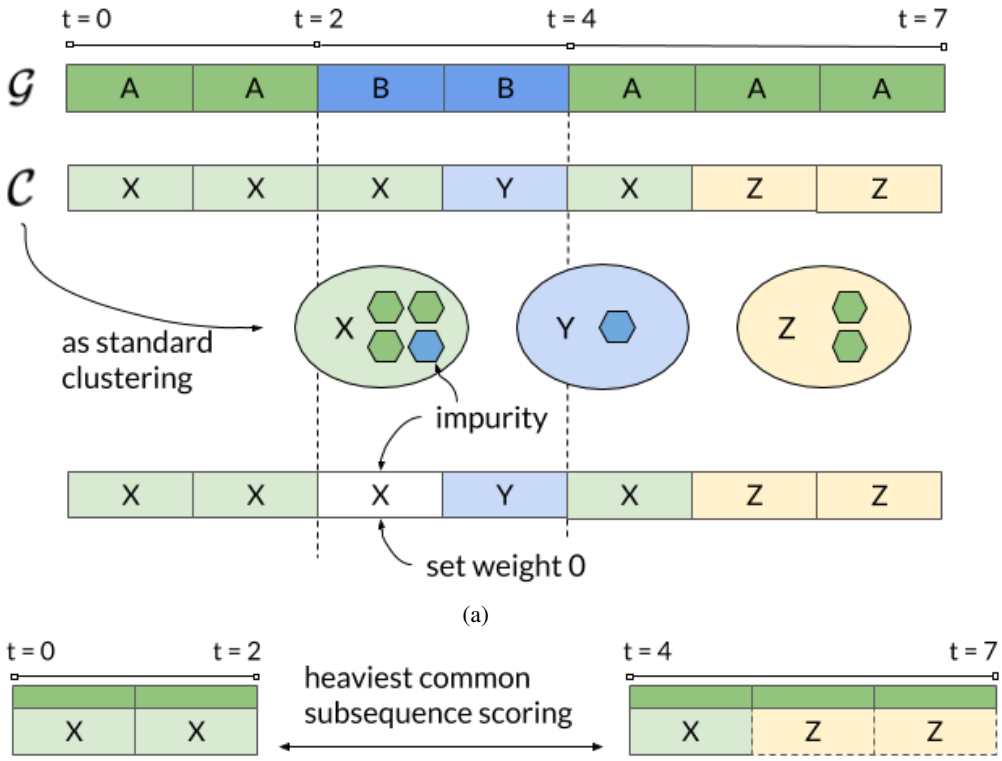
We propose *temporal purity*, a metric that generalizes purity to temporal clusterings by being sensitive to the correct identification of repeated segments. Temporal purity neatly reduces to purity when no ground truth label occurs in multiple segments *i.e.* there is no repeated structure. Similar to purity, temporal purity is a normalized metric that lies in $(0, 1]$.

We give a few definitions before describing how to compute temporal purity. Let $\mathcal{W}$ be a function that counts the number of running duplicates in a sequence *e.g.* $\mathcal{W}(\mathrm{A}, \mathrm{A}, \mathrm{A}, \mathrm{B}, \mathrm{B}, \mathrm{B}, \mathrm{C}, \mathrm{C}, \mathrm{C}, \mathrm{A}, \mathrm{A}, \mathrm{B}) = (3, 3, 3, 2, 1)$. Notice that $\mathcal{W}$ computes the weight of the segment corresponding to each cluster label in the sequence. Recall that $\mathcal{P}$ maps a sequence to its procedure, and for the case above, $\mathcal{P}(\mathrm{A}, \mathrm{A}, \mathrm{A}, \mathrm{B}, \mathrm{B}, \mathrm{B}, \mathrm{C}, \mathrm{C}, \mathrm{C}, \mathrm{A}, \mathrm{A}, \mathrm{B}) = (\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{A}, \mathrm{B})$. Thus, $\mathcal{W}$ is a weighing function which computes the length of each token in the procedure. Also recall that for any $\alpha \in \Gamma$ and a temporal clustering $\mathcal{C}$, $\mathcal{S}(\mathcal{C}, \alpha)$ is the set of segments corresponding to the label α in $\mathcal{C}$.

Let $\mathcal{H}(\mathcal{P}_1, \mathcal{P}_2, \mathcal{W}_1, \mathcal{W}_2)$ be a program that solves the heaviest common sub-sequence problem [29] – it returns the total weight of the heaviest common sub-sequence in $\mathcal{P}_1$ and $\mathcal{P}_2$ (which have associated weights $\mathcal{W}_1$ and $\mathcal{W}_2$ respectively). $\mathcal{H}$ can be computed efficiently using a dynamic program in $O(|\mathcal{P}_1||\mathcal{P}_2|)$. For instance, $\mathcal{H}((\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{A}), (\mathrm{E}, \mathrm{B}, \mathrm{C}, \mathrm{E}), (1, 1, 1, 1), (2, 2, 2, 2))$ identifies the heaviest sub-sequence (B, C) and returns a total weight of $1 + 1 + 2 + 2 = 6$.

The computation of temporal purity is outlined in Algorithm 3. There are 2 steps in the computation of the metric – a preprocessing step and a scoring step. We will describe the scoring step first, since it is the main step in the computation. We will use Fig 3 as an aid in the next few paragraphs as we lay out intuition on what temporal purity is trying to achieve.

For scoring, we first consider a ground truth label β (line 5) – say A in $\mathcal{G}$ in Fig 3. We then pick a pair of segments (a_1, b_1) and (a_2, b_2) from $\mathcal{S}(\mathcal{G}, \beta)$ (lines 11, 12) – in Fig 3, we could pick the pair of segments $(0, 2)$ and $(4, 7)$. These segments can be picked since $\mathcal{G}_{0:2}$ and $\mathcal{G}_{4:7}$ both contain only



(b) A pair of segments being compared and scored. The segments correspond to $\mathcal{C}_{0:2}$ and $\mathcal{C}_{4:7}$. First find $\mathcal{P}(\mathcal{C}_{0:2}) = (\mathrm{X}), \mathcal{W}(\mathcal{C}_{0:2}) = (2)$ and $\mathcal{P}(\mathcal{C}_{4:7}) = (\mathrm{X}, \mathrm{Z}), \mathcal{W}(\mathcal{C}_{4:7}) = (1, 2)$. Then, running $\mathcal{H}((\mathrm{X}), (\mathrm{X}, \mathrm{Z}), (2), (1, 2))$ yields the heaviest common subsequence (X) and a score of 3. The dotted lines around $\mathcal{C}_{5:7}$ show that it does not contribute to the score.

Figure 3: An example to illustrate the computation of temporal purity. (a) demonstrates how the proposed temporal clustering $\mathcal{C}$ is adjusted to purify each cluster (lines 2-10 in Algorithm 3). (b) shows how a pair of segments is compared (lines 11-16 in Algorithm 3).

$\beta = \mathrm{A}$ *i.e.* $\mathcal{G}_{0:2}, \mathcal{G}_{4:7} \subseteq \mathcal{G}[\mathrm{A}]$. Notice that this pair of segments in $\mathcal{G}$ exhibits repeated structure – in fact any pair we would have picked using this method would exhibit repeated structure in $\mathcal{G}$.

We now look at what these identified segments look like in $\mathcal{C}$ *i.e.* we find $\mathcal{C}_{a_1:b_1}$ and $\mathcal{C}_{a_2:b_2}$. In Fig 3 this corresponds to $\mathcal{C}_{0:2} = (\mathrm{X}, \mathrm{X})$ and $\mathcal{C}_{4:7} = (\mathrm{X}, \mathrm{Z}, \mathrm{Z})$.

At this point, we pause and recap what we have done so far: (i) picked a pair of segments that exhibits repeated structure in $\mathcal{G}$; (ii) identified this pair of segments in $\mathcal{C}$. The main idea will now be to score how faithfully the pair of segments in $\mathcal{C}$ capture the repeated structure in $\mathcal{G}$. To do this, we run the heaviest common sub-sequence problem on this pair (line 14). Fig 3b shows details of this computation for our running example – we run $\mathcal{H}((\mathrm{X}), (\mathrm{X}, \mathrm{Z}), (2), (1, 2))$ and it returns a score of 3.

By using the heaviest common sub-sequence function to score the segment pair in $\mathcal{C}$, we can determine whether the repeated structure in $\mathcal{G}$ was discovered by $\mathcal{C}$ and if so, to what extent. At one extreme, if no common sub-sequence is found by $\mathcal{H}$, then there are *no* overlapping tokens in the procedures corresponding to $\mathcal{C}_{a_1:b_1}$ and $\mathcal{C}_{a_2:b_2}$. This directly implies that $\mathcal{C}$ has failed to identify repeated structure in the two segments, and the score awarded for this segment pair is 0. At the other extreme, if both segments being compared are identical, they will overlap perfectly according to $\mathcal{H}$. This implies that $\mathcal{C}$ was able to identify the repeated structure perfectly for these segments and will get the maximum possible score for this segment pair. For segment pairs that lie in-between these extremes, $\mathcal{H}$ finds the best possible overlap to score the segment. This method can now be repeatedly applied for every possible segment pair in $\mathcal{C}$ that exhibits repeated structure in $\mathcal{G}$ (lines $5, 12, 13, 14$ handle this).

The other important step in the algorithm is a preprocessing step that trims each cluster based on its purity. Fig 3a demonstrates how this is carried out. $\mathcal{C}$ is first viewed as a standard, non-temporal

Algorithm 3 Calculation of Temporal Purity

Require: Ground truth $\mathcal{G}$ and temporal clustering $\mathcal{C}$

function TEMPORALPURITY($\mathcal{G}, \mathcal{C}$)
  for all $\alpha \in \Gamma_{\mathcal{C}}$ **do** ▷ for every cluster label α
    $\beta^*_\alpha \leftarrow \arg\max_{\beta \in \Gamma_{\mathcal{G}}} |\mathcal{G}[\beta] \cap \mathcal{C}[\alpha]|$ ▷ find ground truth label β^*_α which most overlaps α
  end for
  for all $\beta \in \Gamma_{\mathcal{G}}$ **do** ▷ for every ground truth label β
    for all $(a, b) \in \mathcal{S}(\mathcal{G}, \beta)$ **do** ▷ for every segment corresponding to β in $\mathcal{G}$
      $\mathcal{P}(\mathcal{C}_{a:b}) := (p_1, p_2, \ldots, p_r)$ ▷ find procedure for this segment
      $\mathcal{W}(\mathcal{C}_{a:b}) := (w_1, w_2, \ldots, w_r)$ ▷ find weights for this segment in $\mathcal{C}$
      $\tilde{w}_i \leftarrow w_i \times \mathbb{I}[\beta = \beta^*_{p_i}]$ ▷ trim weights based on purity
      $\tilde{\mathcal{W}}(\mathcal{C}_{a:b}) \leftarrow (\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_r)$ ▷ construct new weights
    end for
    for all $(a_1, b_1) \in \mathcal{S}(\mathcal{G}, \beta)$ **do**
      for all $(a_2, b_2) \in \mathcal{S}(\mathcal{G}, \beta)$ **do**
        $h(a_1, b_1, a_2, b_2) \leftarrow \mathcal{H}(\mathcal{P}(\mathcal{C}_{a_1:b_1}), \mathcal{P}(\mathcal{C}_{a_2:b_2}), \tilde{\mathcal{W}}(\mathcal{C}_{a_1:b_1}), \tilde{\mathcal{W}}(\mathcal{C}_{a_2:b_2}))$
        ▷ compute score for every pair of segments with ground truth label β
      end for
    end for
  end for
  $\text{score} \leftarrow \sum_{\beta \in \Gamma_{\mathcal{G}}} \sum_{(a_1,b_1) \in \mathcal{S}(\mathcal{G},\beta)} \sum_{(a_2,b_2) \in \mathcal{S}(\mathcal{G},\beta)} h(a_1, b_1, a_2, b_2)$ ▷ sum up scores
  $\text{max\_score} \leftarrow 2 \sum_{\beta \in \Gamma_{\mathcal{G}}} |\mathcal{S}(\mathcal{G}, \beta)| \sum_{(a,b) \in \mathcal{S}(\mathcal{G},\beta)} (b - a + 1)$ ▷ $|\cdot|$ is cardinality
  $\text{temporal purity} \leftarrow \frac{\text{score}}{\text{max\_score}}$
  return temporal purity
end function

clustering. Each cluster $\mathcal{C}[\alpha]$ is purified by marking out impure items. These are items that do not belong to the majority ground truth label β^*_α that constitutes the cluster $\mathcal{C}[\alpha]$. In Fig 3 only the $\mathcal{C}[\text{X}]$ cluster contains a single item that is impure. The impure items are then identified in the original temporal clustering and given 0 weight in all subsequent calculations. This preprocessing step ensures two things: (i) $\mathcal{C}$ is penalized for constructing impure clusters which put items from different ground truth labels together; and (ii) temporal purity reduces to purity in the standard clustering case.

4.3 Temporal Completeness (TC)

Similar to the issue with purity, temporal purity cannot penalize temporal clusterings which contain clusters that are too fine-grained (*e.g.* every item in its own cluster). Just as with completeness, we would like a metric that can identify when the temporal clustering is too fine-grained, and over-segments the trajectory.

When is a temporal clustering $\mathcal{C}$ not over-segmented? We contend that $\mathcal{C}$ is complete when the items in each *segment* (rather than for each ground truth label) of $\mathcal{G}$ are clustered together. By using segments rather than entire labels in the definition of completeness for temporal clustering, we avoid the problems laid out in Fig 2. Stated differently, the best granularity for $\mathcal{C}$ is when its set of transition points between segments align perfectly with those in $\mathcal{G}$.

A major issue in using the standard definition of completeness is that if the repeated structure of some ground truth label is *not* identified by $\mathcal{C}$, completeness penalizes it, even if the granularity of the clustering is correct. For instance, $\mathcal{C}_2$ in Fig 2 is penalized as much as $\mathcal{C}_3$ even though it is less fine-grained (in fact $\mathcal{C}_2$'s transition points perfectly align with $\mathcal{G}$ so it is just right). This is undesirable since (i) we already have a precise metric in temporal purity for penalizing a lack of identified repeated structure in $\mathcal{C}$. Using completeness together with temporal purity would lead to a 'double' penalty in these cases; and (ii) temporal clusterings where transition points between segments align with the ground truth should have maximum score.

To remedy this, we consider the conditional entropy $H\left(\Gamma_{\mathcal{C}} | \bigcup_{\beta \in \Gamma_{\mathcal{G}}} \mathcal{S}(\mathcal{G}, \beta)\right)$ – the conditioning here is over the set of all segments $\bigcup_{\beta \in \Gamma_{\mathcal{G}}} \mathcal{S}(\mathcal{G}, \beta)$ rather than the set of ground truth labels $\Gamma_{\mathcal{G}}$, as was the case in completeness. However, normalizing this quantity by $H(\Gamma_{\mathcal{C}})$ (as done in completeness) is a problem since $H(\Gamma_{\mathcal{C}})$ is sensitive to the number of clusters $|\Gamma_{\mathcal{C}}|$, even if the transition points remain unchanged.

Instead, we normalize the conditional entropy in each segment by the *maximum* possible entropy of that segment. For a particular segment $(a, b) \in \bigcup_{\beta \in \Gamma_{\mathcal{G}}} \mathcal{S}(\mathcal{G}, \beta)$, the conditional entropy of that segment is simply $H(\Gamma_{\mathcal{C}}|(a, b))$. The maximum possible entropy in (a, b) is $\log(b - a + 1)$, and occurs when each item in the segment lies in a different cluster. The normalized conditional entropy becomes $\frac{H(\Gamma_{\mathcal{C}}|(a,b))}{\log(b-a+1)}$. This normalization is independent of the choice of temporal clustering and depends only on $\mathcal{G}$, and specifically on the size of segments in $\mathcal{G}$.

Thus, we have

$$\text{Temporal Completeness} = 1 - \sum_{(a,b) \in \bigcup_{\beta \in \Gamma_{\mathcal{G}}} \mathcal{S}(\mathcal{G},\beta)} \left(\frac{b-a+1}{H}\right) \cdot \frac{H(\Gamma_{\mathcal{C}}|(a,b))}{\log(b-a+1)}$$

In practice, the temporal completeness tends to take high values between $0.8 - 1.0$. Since the normalized conditional entropy $\frac{H(\Gamma_{\mathcal{C}}|(a,b))}{\log(b-a+1)}$ is normalized with respect to the maximum possible entropy, it skews close to small values, and thus the metric takes high values. This is not an issue, but it means that small changes in the metric are generally significant in practice.

4.4 Temporal Clustering Score (TCS)

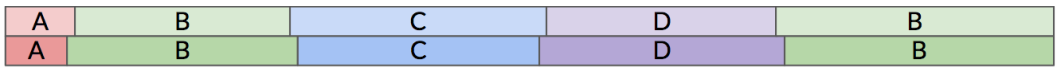
Given the two new metrics, temporal purity and temporal completeness, we now define a single combined metric – the temporal clustering score. It is defined as,

$$\text{Temporal Clustering Score} = \frac{(1+\beta) \cdot \text{TC} \cdot \text{TP}}{(\beta \cdot \text{TP}) + \text{TC}}$$

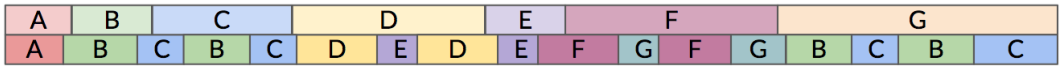
Similar to the V-Measure in standard clustering, the temporal clustering score balances its two constitutent criteria using a weighted harmonic mean. In practice, we use a value of $\beta = 0.1$ since the temporal completeness tends to skew to high values.

5 Experiments

We carry out experiments on a dataset that we collected from YouTube. The dataset consists of 17 videos of pianists playing Bach's Prelude in C major. Videos were picked only if they were captured from a camera that had a top-down view (approximately) of the piano and the pianist's hands. Each video was trimmed to only contain the first 4 bars of the piece, and was manually annotated with two ground truth temporal clusterings – a fine-grained temporal clustering $\mathcal{G}^i_{\text{fine}}$ consisting of 7 clusters and 17 segments (Fig 4b lower) and a coarse temporal clustering $\mathcal{G}^i_{\text{coarse}}$ consisting of 4 clusters and 5 segments (Fig 4a lower). We will attempt to discover these using our methods.



(a) Upper: $\tilde{\mathcal{C}}^1$, the temporal clustering extracted using LTCA (HC) for the coarse 4 cluster case on video 1. Lower: $\mathcal{G}^1_{\text{coarse}}$, the ground truth temporal clustering for video 1.



(b) Upper: $\tilde{\mathcal{C}}^1$, the temporal clustering extracted using LTCA (HC) for the fine 7 cluster case on video 1. Lower: $\mathcal{G}^1_{\text{fine}}$, the ground truth temporal clustering for video 1.

Figure 4: Examples of temporal clusterings identified for video 1 (170 time steps) compared to ground truth for (a) the coarse 4 cluster case and (b) the fine 7cluster case.

	LTCA (HC)	GPE (HC)	LTCA (GMM)	GPE (GMM)	Individual	Canonical
NMI	0.77	0.64	0.65	0.39	0.48	0.70
TCS	**0.89**	0.78	0.61	0.46	0.66	0.89

Table 1: Performance compared to $\mathcal{G}_{\text{coarse}}$ when extracting a coarse 4 primitive procedure.

How different are the videos? As a sanity check of sorts, we show what happens when all n feature trajectories $\mathbf{X}^i$ are clustered together in one go. All time-steps across all feature trajectories $\mathbf{x}_t^i$ are gathered into a single dataset and then clustered using hierarchical clustering (ward linkage). Our hope is that even across demonstrations, the same primitives will have very similar feature representations.

We find that each video is contained inside its own cluster. This indicates that the features extracted have very little commonality *across* videos – each video demonstration's feature trajectory $\mathbf{X}^i$ lies on a different manifold in feature space. This is not surprising – frames within a video share strong visual similarity, and are quite dissimilar to those from another video. However, it seems that using common feature extractors does not remedy this issue.

Experiments with LTCA and GPE. We run LTCA and GPE with both hierarchical clustering (HC) and Gaussian mixture models (GMM) in two separate runs. Python implementations from the package `sklearn` were used for both. The two runs set the number of clusters at $k = 4$ (coarse) and $k = 7$ (fine) respectively. We compare to 2 baselines,

Individual. For each $\mathbf{X}^i$, we compute its temporal clustering $\mathcal{A}(\mathbf{X}^i, k)$ and directly compare it to the ground truth temporal clustering $\mathcal{G}^i$. Note that this baseline cannot extract a procedure but allows us to verify if aggregating using LTCA or GPE helps.

Canonical. For a single canonical $\mathbf{X}^i$ we find its temporal clustering $\mathcal{A}(\mathbf{X}^i, k)$. We then uniformly warp this canonical temporal clustering to apply it to all other videos. To apply it to the j^{th} video, we find $\tau(\mathcal{A}(\mathbf{X}^i, k), H_j)$ and then compare this to the ground truth temporal clustering $\mathcal{G}^j$. For determining which video to pick as the canonical, we simply pick the one which gives the best evaluation on the TCS metric.

The results of all methods are shown in Table 1 and Table 2. For the coarse procedure extraction, we get high quality results that perform favorably compared to the baselines. Fig 4a shows an example of a temporal clustering extracted by LTCA (HC) for video 1. Our identified temporal clustering is quite close to the ground truth qualitatively, and this is reflected in the high value of the TCS metric in Table 1.

However, for the fine procedure extraction, results are poor for all competing methods. While the canonical baseline performs best, no method is able to identify the fine-grained repeated structure in the ground truth. Our hypothesis is that the feature representation used is unable to identify important features that can be used to identify the repeated structure. Identifying a good, general feature representation for video demonstrations is an important direction for future work.

Performance of TCS metric. The TCS metric is able to do a much better job of distinguishing the performance of various methods. A good example is the difference in performance of GPE (HC) and LTCA (GMM) in Table 1. While NMI considers LTCA (GMM) to be similar to GPE (HC) (0.65 v/s 0.64), TCS prefers GPE (HC) by a wide margin. The main reason is that LTCA (GMM) is unable to identify any repeated structure, while GPE (HC) does identify repeated structure. This verifies the issues with traditional metrics laid out earlier.

	LTCA (HC)	GPE (HC)	LTCA (GMM)	GPE (GMM)	Individual	Canonical
NMI	0.59	0.61	0.61	0.57	0.52	0.56
TCS	0.36	**0.40**	0.38	0.32	0.43	**0.46**

Table 2: Performance compared to $\mathcal{G}_{\text{fine}}$ when extracting a fine 7 primitive procedure.

6 Related Work

Our work is related to several areas of research spanning unsupervised learning, computer vision and reinforcement learning.

Temporal clustering. Our work is closely related to research in temporal clustering. Recent work [18, 17] proposed the aligned cluster analysis algorithm, extending kernel k-means and spectral clustering to find temporal clusterings of time-series data. [30] propose a method for temporal clustering that relies on assuming data points lie in an a union of low-dimensional subspaces. Other work from changepoint detection [31, 32] can also be used to generate temporal segmentations, but not find the assignment of segments to clusters. Classical clustering methods [33] can also be used to generate temporal clusterings.

Prior work in temporal clustering has focused on low-dimensional time series such as the bee dances dataset [34] and the CMU motion-capture dataset [19]. The Weizmann dataset [35] is one of the few video datasets used, but it contains extremely simple scenes with a person walking or running against a static background. In contrast, our work focuses on the setting where videos may be extremely varied, complex and noisy.

Latent variable models. A related line of work is in learning latent variable models which model time-series as switching between a set of linear dynamics models. These include switching linear dynamical systems (SLDSs), hidden Markov models (HMMs) [36] and vector autoregressive models (VARs) [37].

Several papers [16, 15] address the challenge of learning SLDSs with a variable number of switching modes by using non-parametric Bayesian methods such as a hierarchical Dirichlet process prior [15] or beta process prior [16]. Recent work [14] used a Dirichlet process Gaussian mixture model as an SLDS to find transition points in surgical video data. Other work includes the infinite Hidden Markov Model [38] which extends the basic HMM model using a Dirichlet process to define an infinite number of hidden components. Other variants of SLDSs [39, 40] introduce recurrence or time-delayed transitions.

Closest to our setting is work on the beta process autoregressive HMM (BP-AR-HMM) [15, 16], which learns models for multiple time-series that share a set of dynamical behaviors. However, similar to work in temporal clustering, they operate on low-dimensional motion capture data. While our setting (n demonstrations realizing a common underlying procedure) is a specific instance of the general problem considered in this work, the complexity of our dataset makes it a non-trivial problem to solve.

Activity recognition and localization. Papers from computer vision in activity recognition typically operate on complex video datasets such as the Bach dataset collected by us. However, they typically rely on supervision in the form of activity labels to learn a classification model for activities. Work in action detection identifies a single action present in a video clip [41–44] after learning a model on training data. A more complex setting is where videos may contain multiple actions which must be identified and temporally segmented [45–48, 7, 49], given a densely annotated training set of localized actions.

There is also some work in semi-supervised and unsupervised action localization such as [50–52]. [50] use information about the actions but not their localization, [51] focus on activities where it is possible to do skeletal modeling of human actions while [52] propose a pipeline for building an unsupervised dictionary of actions.

Learning from demonstration. Learning from demonstration in the reinforcement learning setting involves mimicking a demonstrator's policy using expert trajectories collected from them [53, 54]. The setting bears some similarity to ours since trajectories often correspond to performing the same procedure. However, access to both control inputs and the state of the demonstrator are assumed which is not true in our case.

7 Conclusion

In this work, we introduced the problem of identifying the common procedure underlying a dataset of task video demonstrations without any supervision. This problem has important applications in

learning task representations and creating general tutoring systems. We identified the main challenges that need to be addressed in order to solve the problem, and described two general-purpose methods for learning a common procedure.

We also identify a gap in the set of evaluation criteria used by work in temporal clustering and activity recognition. We instead propose several metrics that remedy these gaps and can be used for general purpose evaluation. Lastly, we validate our methods and evaluation criteria on a dataset of YouTube video demonstrations of a piano piece.

8 Acknowledgments

The authors would like to acknowledge the inputs of Daniel Guo, Tong Mu, Mina Lee at Stanford as well as the useful discussions with Christoph Dann and Geoff Gordon at CMU.

References

[1] Duane F Shell, David W Brooks, Guy Trainin, Kathleen M Wilson, Douglas F Kauffman, and Lynne M Herr. The unified learning model. In *The Unified Learning Model*, pages 1–4. Springer, 2010.

[2] John Seely Brown, Allan Collins, and Paul Duguid. Situated cognition and the culture of learning. *Educational researcher*, 18(1):32–42, 1989.

[3] Vanessa P Dennen and Kerry J Burner. The cognitive apprenticeship model in educational practice. *Handbook of research on educational communications and technology*, 3:425–439, 2008.

[4] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[5] Philip J Guo, Juho Kim, and Rob Rubin. How video production affects student engagement: An empirical study of mooc videos. In *Proceedings of the first ACM conference on Learning@ scale conference*, pages 41–50. ACM, 2014.

[6] Natalie B Milman. The flipped classroom strategy: What is it and how can it best be used? *Distance Learning*, 9(3):85, 2012.

[7] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. Cdc: convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1417–1426. IEEE, 2017.

[8] Vignesh Ramanathan, Percy Liang, and Li Fei-Fei. Video event understanding using natural language descriptions. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 905–912. IEEE, 2013.

[9] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[10] Pierre Sermanet, Corey Lynch, Jasmine Hsu, and Sergey Levine. Time-contrastive networks: Self-supervised learning from multi-view observation. *arXiv preprint arXiv:1704.06888*, 2017.

[11] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. *arXiv preprint arXiv:1703.01703*, 2017.

[12] Josh Merel, Yuval Tassa, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

[13] Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*, 2017.

[14] Sanjay Krishnan, Animesh Garg, Sachin Patil, Colin Lea, Gregory D. Hager, Pieter Abbeel, and Kenneth Y. Goldberg. Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning. *I. J. Robotics Res.*, 36:1595–1618, 2015.

[15] Emily B. Fox, Erik B. Sudderth, Michael I. Jordan, and Alan S. Willsky. Nonparametric bayesian learning of switching linear dynamical systems. In *NIPS*, 2008.

[16] Emily B. Fox, Erik B. Sudderth, Michael I. Jordan, and Alan S. Willsky. Sharing features among dynamical systems with beta processes. In *NIPS*, 2009.

[17] Feng Zhou, Fernando De la Torre, and Jessica K. Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:582–596, 2013.

[18] Feng Zhou, Fernando De la Torre, and Jessica K. Hodgins. Aligned cluster analysis for temporal segmentation of human motion. *2008 8th IEEE International Conference on Automatic Face Gesture Recognition*, pages 1–7, 2008.

[19] Leonid Sigal, Alexandru O Balan, and Michael J Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International journal of computer vision*, 87(1-2):4, 2010.

[20] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[22] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, 1994.

[23] Feng Zhou and Fernando De la Torre. Canonical time warping for alignment of human behavior. In *NIPS*, 2009.

[24] Feng Zhou and Fernando De la Torre. Generalized canonical time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:279–294, 2016.

[25] Nam Nguyen and Rich Caruana. Consensus clusterings. *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 607–612, 2007.

[26] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.

[27] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[28] Douglas Steinley. Properties of the hubert-arable adjusted rand index. *Psychological methods*, 9(3):386, 2004.

[29] Guy Jacobson and Kiem-Phong Vo. Heaviest increasing/common subsequence problems. In *Annual Symposium on Combinatorial Pattern Matching*, pages 52–66. Springer, 1992.

[30] Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:2765–2781, 2013.

[31] Zaïd Harchaoui, Francis R. Bach, and Eric Moulines. Kernel change-point analysis. In *NIPS*, 2008.

[32] Steven M Kay. *Fundamentals of statistical signal processing: Practical algorithm development*, volume 3. Pearson Education, 2013.

[33] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[34] Sang Min Oh, James M Rehg, Tucker Balch, and Frank Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, 77(1-3):103–124, 2008.

[35] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, December 2007.

[36] Sean R Eddy. Hidden markov models. *Current opinion in structural biology*, 6(3):361–365, 1996.

[37] Helmut Lütkepohl. Vector autoregressive models. In *International Encyclopedia of Statistical Science*, pages 1645–1647. Springer, 2011.

[38] Matthew J. Beal, Zoubin Ghahramani, and Carl E. Rasmussen. The infinite hidden markov model. In *NIPS*, 2001.

[39] Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Artificial Intelligence and Statistics*, pages 914–922, 2017.

[40] Sang Min Oh, James M Rehg, and Frank Dellaert. Segmental switching linear dynamic systems. Technical report, Georgia Institute of Technology, 2005.

[41] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 4489–4497. IEEE, 2015.

[42] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Actom sequence models for efficient action detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3201–3208. IEEE, 2011.

[43] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Activity representation with motion hierarchies. *International journal of computer vision*, 107(3):219–238, 2014.

[44] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. Going deeper into action recognition: A survey. *Image and vision computing*, 60:4–21, 2017.

[45] Alberto Montes, Amaia Salvador, Santiago Pascual, and Xavier Giro-i Nieto. Temporal activity detection in untrimmed videos with recurrent neural networks. *arXiv preprint arXiv:1608.08128*, 2016.

[46] Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1914–1923, 2016.

[47] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Daps: Deep action proposals for action understanding. In *European Conference on Computer Vision*, pages 768–784. Springer, 2016.

[48] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. Sst: Single-stream temporal action proposals. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6373–6382. IEEE, 2017.

[49] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2678–2687, 2016.

[50] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, pages 137–153. Springer, 2016.

[51] Chenxia Wu, Jiemi Zhang, Silvio Savarese, and Ashutosh Saxena. Watch-n-patch: Unsupervised understanding of actions and relations. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4362–4370. IEEE, 2015.

[52] Yang Yang, Imran Saleemi, and Mubarak Shah. Discovering motion primitives for unsupervised grouping and one-shot learning of human actions, gestures, and expressions. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1635–1648, 2013.

[53] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[54] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.