Programmatic Concept Learning for Human Motion Description and Synthesis

Sumith Kulal*
Stanford University

Jiayuan Mao*
MIT

Alex Aiken†
Stanford University

Jiajun Wu†
Stanford University

Abstract

We introduce Programmatic Motion Concepts, a hierarchical motion representation for human actions that captures both low-level motion and high-level description as motion concepts. This representation enables human motion description, interactive editing, and controlled synthesis of novel video sequences within a single framework. We present an architecture that learns this concept representation from paired video and action sequences in a semi-supervised manner. The compactness of our representation also allows us to present a low-resource training recipe for data-efficient learning. By outperforming established baselines, especially in the small data regime, we demonstrate the efficiency and effectiveness of our framework for multiple applications.

1. Introduction

The advent of new datasets and progress in machine learning has pushed the boundaries of several video analysis tasks. In particular, there has been tremendous progress on description tasks such as action recognition [20, 22, 36, 46, 47] and localization [4, 11, 12, 44, 53, 57], and on synthesis tasks such as human motion synthesis [1, 14–16, 27, 37, 43] and video synthesis [5, 40, 45, 50]. However, most of these models focus solely on their respective tasks of video description or synthesis. We posit that these tasks are better learned jointly. We present a single framework for human motion description (recognizing and temporally localizing individual actions in the video), synthesis (generating videos from abstract descriptions) and editing (adding or removing actions and other fine-grained manipulations).

Our key insight is a hierarchical motion representation, *Programmatic Motion Concepts* (PMC). Beginning with standard low-level input (sequences of keypoints), we learn a representation of the distribution of high-level motion concepts (such as jumping jacks) first from a few examples and then from longer sequences of repetitions of the concept that

we automatically segment. More specifically, for each motion concept and each body keypoint, we learn a distribution of time-space trajectories expressed as cubic splines. The fact that we explicitly learn a distribution of motions at the level of the entire motion concept is what makes it possible to perform recognition and localization well, since our methods leverage information about the motion of individual keypoints over the entire span of the motion concept. Similarly, sampling from these distributions allows us to directly synthesize natural-looking instances of the motion concept.

PMC is based on prior work on hierarchical motion understanding with programmatic primitive-based representations [23]. Illustrated in Figure 1(I), a human motion sequence is represented as a sequence of *motion primitives*, which are compact, and human-interpretable, parametric curves. We further represent *motion concepts* as groupings of consecutive motion primitives that are named by humans, such as *jumping jack* and *squat* in workout videos, as shown in Figure 1(I). This representation choice has access to both low-level pose sequences and high-level description sequences as motion concepts.

We want to enable easy and efficient learning of our representation to quickly adapt this framework to different domains. Hence, we propose to learn this representation from *weakly-annotated* data. Our input data contains each concept annotated with only a few mouse clicks by the human annotator. For example, the annotation of each concept in our dataset consists of weak labels from < 15 workout videos. Illustrated in Figure 1(II), in each video, the human annotator provides a start and end point for a video segment that contains repetitions of jumping jacks as well as the start and end points of three individual jumping jack instances. Our method learns two models from this data, a *recognition* model that can be applied on any human motion sequences to detect occurrences of jumping jacks and a *generative* model that can synthesize jumping jack motions.

The key ideas behind our data efficient learning algorithm are three-fold. First, instead of using human skeleton sequences as input, our model operates on a representation of motion as time-space curves, which is a strong but generically applicable inductive bias for videos of human motion.

I. Programmatic Motion Concept

00:00	02:08	02:13	02:18	02:24	02:33

Motion Concepts   **21 ×Jumping Jacks**   **30 × High Knee Freeze**

Motion Primitives
(R Wrist)

```
concept(jumping_jack, motion=[
  spline(x=-0.07 * t^3 + 0.87 * t^2 + 1.74 * t + 275.45, y=......, duration=15)
  spline(x=-0.07 * t^3 + 0.25 * t^2 + 6.93 * t + 268.47, y=......, duration=13)
]) ... similar motion repeated for 20 times
```

```
concept(high_knee_freeze,
  motion=[...]
) ... similar motion
```

II. Human Annotations

00:00	02:08	02:13	02:18	02:24	02:33

Repetition Range
Instance Range

Label:
Jumping Jacks

III. Motion Description, Editing, and Synthesis

Input Video   **Motion Description** → **Concept-Annotated Motion Program**   **Motion Editing (Bi-Level)** → **Concept-Annotated Motion Program (Edited)**   **Motion Synthesis** → **Output Video**

```
concept(jumping_jack, motion=[...])
... similar motion repeated for 20 times
concept(high_knee_freeze, motion=[...])
... similar motion repeated for 29 times
```

```
concept(jumping_jack, motion=[...])
... similar motion repeated for 21 times
concept(squat, motion=<blank>)
... similar motion repeated for 19 times
concept(high_knee_freeze, motion=[...])
... similar motion repeated for 29 times
```
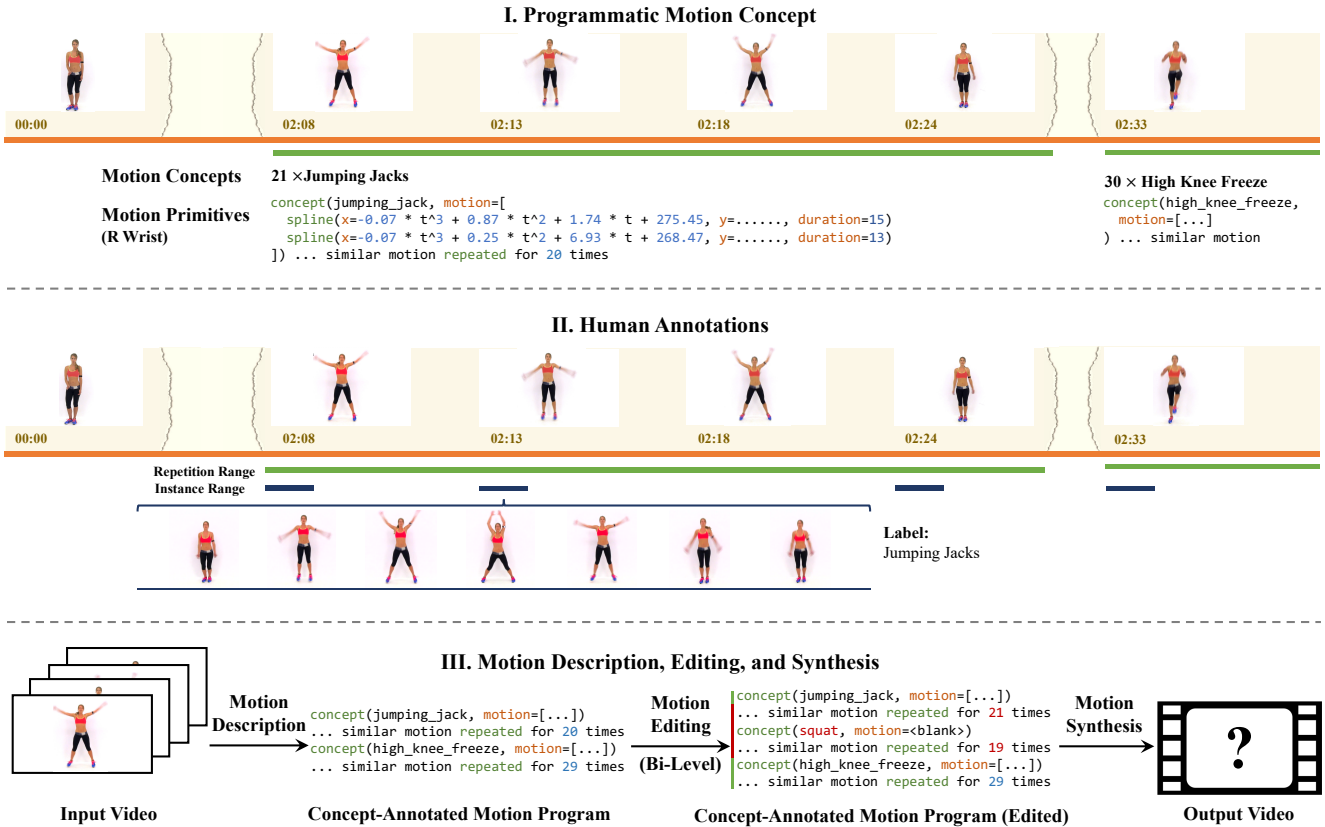
Figure 1. (I) We present a hierarchical human motion description framework. Each video is represented as a sequence of *motion concepts* and each *motion concept* is further grounded as a sequence of *motion primitives*. (II) Motion concepts can be learned from very small amount of human annotation: human annotators label a repetition range of a motion concept and just three instance ranges with in this group of repetitions. (III) Motion concept supports interactive editing and video synthesis. Human editors can flexibly edit a human motion video at both the concept level or the primitive level. We use neural generative models to render the output video.

Second, our model exploits the repetition of motion concepts in a single video. By annotating the beginning and the end of the repetition and a few occurrences of the action, our model automatically extrapolates to all instances in the repetition span. Third, the recognition and generation models are trained jointly, aiming at localizing individual occurrences of actions in the training data and learning the corresponding distribution of motion sequences, which significantly improves the quality of our motion synthesis model.

Programmatic Motion Concepts supports multiple human video analysis tasks, including recognition, localization, synthesis, and editing. Illustrated in Figure 1(III), given the input motion sequence, our model produces a human-interpretable label sequence of motion concepts. Each label is localized to a time span of the input. Users can tune the parameters for different motion primitives and even edit the label sequence itself to synthesize new motion sequences. Combined with techniques for human skeleton detection and skeleton-to-video synthesis, our data-efficient methods can be used to construct a full, practical pipeline for video-to-concept and concept-to-video workflows.

To summarize, our contributions are:

- We present a novel hierarchical representation of human motion that jointly supports description, synthesis, and editing of human motion videos.
- We present an data-efficient learning algorithm that leverages a motion primitive-based representation and repetitive structures in videos.
- Finally, we demonstrate the efficiency and effectiveness of our concept learning framework on three downstream tasks: motion description, action-conditioned motion synthesis, and controlled motion and video synthesis. We also present qualitative results on interactive editing.

2. Related Work

Action recognition and localization.   Large datasets [4, 22, 47] have led to deep learning architectures performing data-driven action recognition without hand-designed features [8, 46, 49, 51]. Recent works also tackle the closely related problem of localization of actions [3,6,19,55,58] that predict the temporal bounds of actions. Current state-of-the-art approaches for recognition [2,29,41] and localization [54] involve variants of the Transformer architecture.

Most relevant to us is work on weakly-supervised action

localization from video-level labels [18,35,52]. E-CTC [18] proposed an extended CTC framework enforcing frame similarity consistency. Other works [35,52] sample and classify key proposals from untrimmed videos to generate video-level labels. Our work is related to [18] in the learning architecture design but in addition to inferring video-level labels, we are able to leverage our learnt concept representation for synthesis and editing. Since most of these methods operate directly on frame-level input which is data-intensive, we use a variant of CTC [18] that operates on the pose-level inputs as our baseline for recognition and localization performance. **Human motion and video synthesis.** Methods performing human motion synthesis either use motion graphs [21,34], RNNs [13,25,26,32] or autoencoders [16]. Most relevant are works that perform a) action-conditioned motion synthesis and b) controlled motion synthesis from descriptions.

Action-conditioned motion synthesis is the problem of generating natural and diverse motion from a given action class. Action2Motion [15] proposes a VAE approach and uses Lie algebra to represent natural human motion. AC-TOR [37] generates SMPL human bodies using a Transformer VAE. In contrast, we learn a primitive distribution over curves as our concept representation for each action class. For action-condition motion synthesis, we sample this distribution to generate novel sequences.

Recent studies [1,14,27,38,39] have looked at controlled motion synthesis from textual descriptions. Most approaches use an RNN-based encoder-decoder model to map descriptions to poses with learning techniques proposed to improve generation quality. In contrast, we perform controlled motion synthesis in a modular way by synthesizing poses for individual concepts and stitching them together, which enables scaling to larger and novel descriptions. Finally, we close the loop by synthesizing realistic videos from the generated motion using prior work in skeleton-to-video synthesis [5,40]. **Primitive-based representations for vision.** Several recent works have combined deep neural networks with symbolic structures such as programs to capture higher-level structures. This has been demonstrated on hand-drawn sketches [9], natural images [31,56] and videos [23]. We build on previous work [23] and extend it to capture more expressive motion.

Learning of visual concepts in deep networks has been demonstrated in other domains [24,30,42,48]. NS-CL [30] maps image patches and words to a joint embedding space for improved performance and data-efficiency in VQA. Similarly, several works in 3D understanding [24,42,48] exploit structure and repetition in 3D shapes to infer concepts such as *table top* and *chair leg*. We extend this idea to human motion, by learning concepts for exercises by exploiting their repetitive nature.

3. Motion Concept Learning Framework

Our goal is to learn a vocabulary of concept representations that supports video description and synthesis tasks.

We define the description L of a video V as a *hierarchical* abstraction at three different levels: primitive level, concept level, and program level. Depicted in Figure 1(I), a *motion primitive* is a parameterized trajectory of the subject's joints. A *motion concept* is composed of one or more motion primitives, associated with a linguistic name such as *jumping jacks* and *squats*. A video can be described at the *program* level as a sequence of motion concepts, each of which is further concretized as a subsequence of motion primitives.

Our representation not only facilitates human interpretation and analysis of the input video, but also supports interactive editing tasks. Shown in Figure 1(II), our framework supports interactive editing of the input video at all three levels: users can tweak the parameters for individual primitives, change the concept labels of existing motion concept segments, or even add and remove motion concepts in the video.

Although these tasks have been studied previously, we focus on creating a unified system for human motion description and synthesis using a small amount of human annotation. For example, the annotation of each concept in our dataset uses weak labels from < 15 videos on average.

In the following section, we introduce our framework, *Programmatic Motion Concepts*, a data-efficient learning framework for human motion description and synthesis. PMC addresses the challenging data-efficient learning problem by leveraging the hierarchical structure of human motions as well as the repetitive structure of concepts in videos. We start from reviewing primitive-based representations for human motion.

3.1. Primitive-based Motion Representation

Programmatic motion primitives, first presented in [23], are a primitive-based motion representation that capture a level of abstraction higher than pose sequences. The key idea is representing motion trajectories as a sequence of sub-trajectories, each of which can be described with few parameters. Kulal et al. [23] focus on three families of primitives: stationary, linear, and circular. The goal is to fit the motion parameters to well-approximate the input trajectory. The inference of motion primitive sequences involves a dynamic programming algorithm that segments the input pose trajectories at the best possible locations. The recurrence relation for the best fit sequence of primitives for the first n frames is given as $\text{Prims}_n = \min_{k<n} [\text{Prims}_k + \text{synt}(\text{poses}[k:n]) + \lambda]$.

Here, Prims_n is the best fit sequence of primitives for the first n frames and $\text{synt}(\text{poses}[k:n])$ is the best-fit single primitive synthesized for frames k to n. The parameter λ controls the granularity of the synthesized primitives.

3.2. Refined Spline-based Primitives

Motion primitives as defined above do well at representing human motions in a wide variety of videos. However, there are multiple failure modes that make it inapplicable for
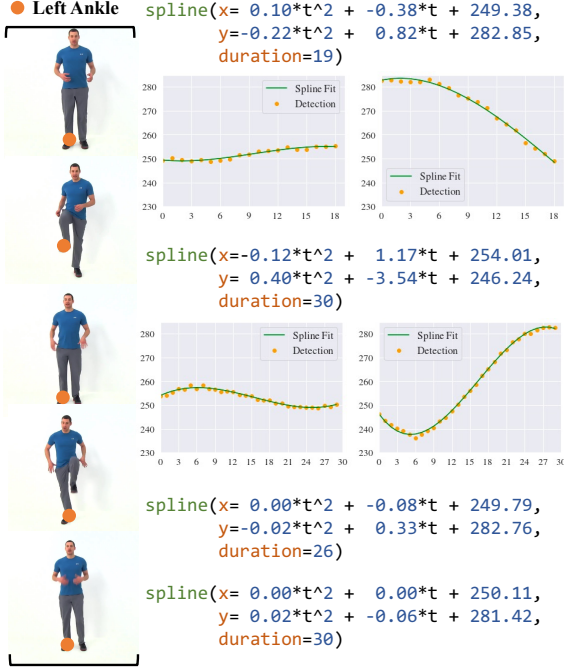
```
● Left Ankle    spline(x= 0.10*t^2 + -0.38*t + 249.38,
                       y=-0.22*t^2 +  0.82*t + 282.85,
                       duration=19)
```

```
                spline(x=-0.12*t^2 +  1.17*t + 254.01,
                       y= 0.40*t^2 + -3.54*t + 246.24,
                       duration=30)
```

```
                spline(x= 0.00*t^2 + -0.08*t + 249.79,
                       y=-0.02*t^2 +  0.33*t + 282.76,
                       duration=26)
```

```
                spline(x= 0.00*t^2 +  0.00*t + 250.11,
                       y= 0.02*t^2 + -0.06*t + 281.42,
                       duration=30)
```

Figure 2. Four detected motion primitives in a High Knees clip. The third order coefficient is small in all four primitives (absolute value is smaller than 0.01) so we omitted them in the visualization.
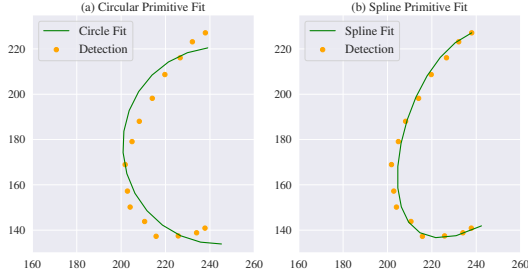
Figure 3. Refined spline primitives can better capture the right-wrist motion in the Jumping Jacks class compared to circular primitives.

our use case of generic human exercise videos. Specifically, some human joint trajectories do not fall into any of the linear/circular/stationary categories. We introduce spline-based primitives which are more general curves than lines and circles while still having only a small number of parameters to learn. We define our spline primitive as

$$\text{spline} = \begin{bmatrix} X = a_x t^3 + b_x t^2 + c_x t + d \\ Y = a_y t^3 + b_y t^2 + c_y t + d \\ T = t \end{bmatrix}. \quad (1)$$

Figure 2 shows the synthesized motion primitives in a high knees exercise. We present more evaluations of these primitives in Section 4.2. We generalize the motion primitive language of [23] to spline-based primitives and use their fast parallel inference algorithm. Figure 3 demonstrates a concrete example of the improved expressiveness. The raw input trajectory is a path traced by the right-wrist keypoint in

a jumping jacks video. We observe that the spline primitive on the right approximates this trajectory much better than the best-fit circular primitive.

3.3. Human Motion Description

The pipeline in PMC for inferring human motion descriptions L from videos V is illustrated in Figure 4. Given the input video as a sequence of 2D images, of length T, we first run a human pose estimation process [10] to extract the trajectories of human joints $\bar{P} = \{p_1, p_2, \cdots p_T\}$. We then use the motion primitive segmentation algorithm described in 3.1 to transform the skeleton sequence into primitive sequences $\bar{S} = \{s_1, s_2, \cdots, s_K\}$. The total number of primitives, K, is automatically determined by the motion primitive inference algorithm. Our learning task is completely based on this higher-level representation of motion trajectories.

Next, we encode each motion primitive with a multi-layer perceptron (m-MLP), which is shared across all primitives. For each pair of consecutive primitives, we encode their features with a separate multi-layer perceptron (t-MLP), which is also shared across all consecutive pairs. The input to t-MLP is the concatenation of two neighboring primitive parameters. Encoding the transition between two primitives helps us capture the boundaries of motion concepts. We use the same number of output channels for m-MLPs and t-MLPs.

This step gives us in total $2K - 1$ latent features as a sequence: K steps for motion primitives and $K - 1$ steps for consecutive primitive pairs. We further aggregate the contextual information of neighboring latent vectors by running a recurrent cell (we investigated both LSTM [17] and GRU [7]) in a local temporal window (the window size is set to 13 in our experiments). Finally, we decode a concept label distribution for each step $\bar{C} = \{c_1, c_2, \cdots, c_{2K-1}\}$. Each class label is either a category for the motion concept that this primitive belongs to or a "blank" token, indicating that it's not part of a motion concept. We refer to $c_1, c_3, \cdots, c_{2K-1}$ as *primitive concept* predictions and $c_2, c_4, \cdots c_{2K}$ *transition concept* predictions.

Based on the concept label prediction sequence $\bar{C}$, we generate the final prediction of the motion concepts in the video $\bar{L}$ by first iteratively merging consecutive predictions c_i and c_{i+1} with the same label, and then removing all "blank" labels. For example, if $\bar{C} = \{JJ, JJ, JJ, Blank, JJ, JJ, JJ\}$ (JJ is short for "jumping jack"), the final output with be $\bar{L} = \{JJ, JJ\}$. We name this process as $\bar{L} = compress(\bar{C})$. We further denote $uncompress(\bar{L}) = \{\bar{C}|\bar{L} = compress(\bar{C})\}$ as the preimage of *compress*, i.e., the set of concept label sequences $\bar{C}$ that leads to the same output sequence $\bar{L}$.

Now we can formally define the probability distribution over all possible output sequences $\bar{L}$. We define $p(\bar{L}|\bar{S}) = \sum_{\bar{C} \in uncompress(\bar{L})} p(\bar{C}) =$
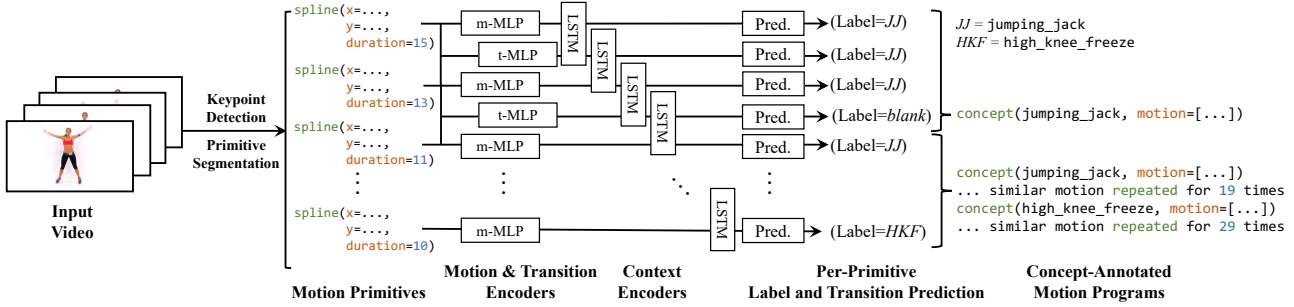
Figure 4. The model for generating motion descriptions. Given the input video, we first generate motion primitives and feed in the primitives to a simple neural architecture containing feed-forward and recurrent layers to compute concept label predictions for each primitive. We use a connectionist temporal classification-style (CTC) objective to extract the most probable motion description.

$\sum_{\bar{C} \in uncompress(\bar{L})} \prod_{t=1}^{2K-1} p\left(c_t | \bar{S}\right)$, where $p\left(c_t | \bar{S}\right)$ is the concept label sequence predicted by the neural network.

The inference of the labeling sequence $\bar{L}$ can be mathematically formed as finding the $\arg \max_{\bar{L}} p\left(\bar{L} | \bar{S}\right)$, which can be solved with a dynamic programming procedure. We discuss this procedure in our supplementary material.

Remark. Our inference algorithm, as well as one of the loss functions we use to train the neural network model, is similar to the connectionist temporal classification (CTC) model. The essential difference between our use case and a standard CTC model is the inclusion of concept label prediction for each pair of consecutive primitives (t-MLPs). This adaptation is important for our domain: We deal with videos where a single action repeats multiple times, so we need a mechanism to insert correct "blank" tokens between two consecutive repetitions of the same motion concept.

3.4. Low-Resource Training of Motion Concepts

Training neural networks for recognizing motion concepts can be data and annotation-intensive, because we need frame-level concept annotations that are laborious to obtain for videos with multiple repetitions. Our goal is to minimize the amount of annotation effort of the user of our system. While we have included a full description and a demo of our annotation tool in the supplementary material, here we briefly introduces the design choices of our labeling tools and how we leverage different types of labels provided by the user to train our motion description model.

Our labeling process leverages the fact that a single motion concept usually repeats multiple times in an exercise video, e.g., a person does 20 jumping jacks as a group. Our annotation of the motion concepts is *hierarchical*: the annotator first specifies the start frame and the end frame of a "repetition range" of the motion concept and assigns a label to it (e.g., *jumping jacks*). Next, the person specifies the start and end frames of three "instance ranges" of the occurrences in this "repetition range".

Our labelling pipeline is efficient and quick to annotate. We include a demo in the supplementary material showing how to annotate the start and end boundaries and three repetition intervals for every action class instance, requiring eight

mouse-clicks per video.

Second, we aid the labeling process by supplementing diagrams showing the trajectory of human joints. Typically, a motion concept will start and end at a "local extremum" of a trajectory. The annotators can click on the local extremum of the trajectory to accurately locate them.

Next, we leverage the hierarchical annotation of motion concept sequences and our primitive based representation of human motion to train the entire model. The first objective we used to train the model is a CTC-style loss. Based on the three instance ranges in a repetition range, we compute the average length of the motion concept occurring in the range and estimate the total number n of occurrences. Then, we compute a pseudo concept label sequence by repeating the concept label n times. We denote the generated label sequence as $\bar{L}^{\dagger}$. Our first loss function is: $\mathcal{L}_{\text{CTC}} = -\log p\left(\bar{L} = \bar{L}^{\dagger} | \bar{S}\right)$.

We also enforce that, all concept label predictions corresponding to a motion primitive in a repetition range (i.e., the primitive concept predictions, in contrast to the ones associated with each pair of consecutive primitives) should output the concept label associated with the repetition range, C. Let K be the number of motion primitives; these predictions are $c_1, c_3, \cdots, c_{2K-1}$. Our second loss function is: $\mathcal{L}_{\text{P}} = -\log \prod_{i=1,3,\cdots,(2K-1)} p\left(c_i = C | \bar{S}\right)$.

We use a similar idea to generate pseudo-labels for all concept label predictions corresponding to each pair of consecutive primitives (i.e., the transition concept predictions). Within each repetition range, based on the average length of occurrence of motion concept C, we segment the motion primitive sequences at the locations that are close to any multiples of the average length. Next, we generate the pseudo-label sequence for transition concept predictions: $c_2^{\dagger}, c_4^{\dagger}, \cdots, c_{2K}^{\dagger}$. $c_{2i}^{\dagger} = C$ if s_i and s_{i+1} belongs to the same segment, and $c_{2i}^{\dagger} = blank$ otherwise. The final loss function is: $\mathcal{L}_{\text{T}} = -\log \prod_{i=2,4,\cdots,2K} p\left(c_i = c_i^{\dagger} | \bar{S}\right)$.

The final loss function is $\mathcal{L} = \mathcal{L}_{\text{CTC}} + \lambda_1 \mathcal{L}_{\text{P}} + \lambda_2 \mathcal{L}_{\text{T}}$. Initially, $\lambda_1 = \lambda_2 = 1$. The purposes of $\mathcal{L}_{\text{P}}$ and $\mathcal{L}_{\text{T}}$ are to bias the network towards a preferred solution. Since they are
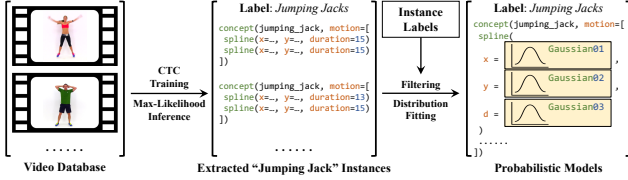
Figure 5. Learning a generative model of motion concepts. We leverage the learnt alignments to extract all single repetition examples for each action class. By taking few ground-truth repetitions as reference, we filter them to resolve ambiguities and then learn a primitive distribution as our concept representation.

approximate, we set $\lambda_1 = \lambda_2 = 0$ after certain epochs.

3.5. Human Motion Synthesis

Once we have a learnt model for video description, we can also infer the most probable labels and corresponding alignments for all videos in the dataset, which is done in two steps. Following the notation in Section 3.3, first, given the motion primitive sequence $\bar{S}$, we compute $\bar{L}^* = \arg\max_{\bar{L}} p\left(\bar{L}|\bar{S}\right)$, which is the optimal label sequence associated with the input primitive sequence. Next, we compute $\bar{C}^* = \arg\max_{\bar{C} \in uncompress(\bar{L})^*} p(\bar{C})$, which is the most probable (uncompressed) concept label sequence that yields the label sequence $\bar{L}^*$. This essentially gives us a label associated with each primitive and thus the segmentation of the input primitive sequence $\bar{S}$ into segments, where each segment forms a motion concept.

We use these alignments to extract individual repetitions for all motion concepts from the training dataset. Thus, for each motion concept c, we obtain a dataset of occurrences $D_c = \{d_1, d_2, \cdots\}$ where each element d_i corresponds to a small segment composed of a sequence of spline curves $d_i = \{s_i^1, s_i^2, \cdots s_i^{l_i}\}$. We then perform two steps of filtering of this dataset D_c. First, we compute mode of the number of splines used in instances of this concept $l_c^* = mode\left(\{l_i\}\right)$ and then filter out all d_i's whose number of splines is not l_c^*. Furthermore, we threshold and choose primitives that agree well with the single repetition examples already annotated for concept c. Denote the filtered set of occurrences for concept c as D_c'. Each element in D_c' has the same number of spline curves. We learn a distribution for each spline curve by fitting a simple Gaussian model over the parameters. We can then sample these distributions to generate novel instances of concept c. This pipeline is visualized in Figure 5. The details of the model fitting and data filtering are included in the supplementary material.

Based on the learned generative model for motion concepts, we can perform both action-conditioned motion synthesis and controlled motion synthesis from descriptions. To perform action-conditioned motion synthesis, we sample the learnt primitive distributions for the corresponding action.

To perform controlled motion synthesis given our descriptions, for example *four jumping jacks followed by three*

Table 1. MotiCon dataset statistics. For each class, we list **# vids** (total number of videos), **# mins** (total number of minutes) and **# reps** (total number of repetitions).

Class	# vids	# mins	# reps
Arm Cross Swings	12	6.68	127
Buttkickers	9	5.94	425
High Knee March	8	3.70	81
Jumping Jack	23	13.03	853
Toe Touch Sweeps	11	5.00	35
Torso Twists	8	3.90	79

squats, we sample the learnt distributions for the required number of distributions in the given order of actions. We stitch these segments together by manually setting the start pose of a segment to the last pose of the previous segment for smooth transitions. Hence, our approach is modular and scales better to longer descriptions than previous methods.

4. Experiments

We conduct multiple experiments to evaluate the performance of our framework. First, we compare the spline-based primitives with the primitives proposed in previous work [23] in Section 4.2. We evaluate PMC on three downstream applications: i) human motion description in Section 4.3, ii) action-conditioned motion synthesis in Section 4.4, and iii) controlled motion and video synthesis from descriptions in Section 4.5. Qualitative results on interactive editing can be found in the supplementary material. All downstream applications are evaluated on the MotiCon dataset.

4.1. MotiCon Dataset

We have gathered a dataset of human cardio videos to evaluate the concept learning framework and the downstream tasks. We scraped YouTube for single human exercise videos and filtered these videos to those with a stable single camera, 1920×1080 resolution and 30 FPS. We label 6 action classes with the interval and single rep annotations as mentioned earlier. The statistics of the final dataset are in Table 1. We refer to this dataset as MotiCon in our experiments.*

4.2. Evaluation of Spline Primitives

We use two metrics. First, we are interested in how well the primitive-based representation encodes the ground-truth poses. We execute our primitive-based representation to generate poses at each time step and compute the keypoint difference (KD) of the generated and ground-truth poses. We report KD as L2 error per joint per frame averaged across all videos (lower is better), and the average time for primitive synthesis in seconds (Time, lower is better).

Analysis. We evaluate on two datasets: GolfDB [33]† and

---

*Our dataset contains videos with Standard YouTube license. It is available on our website as a list of URLs with timestamps and annotations.

†GolfDB is released with a CC BY-NC 4.0 license.

Table 2. Evaluating expressive power of motion primitives. On two datasets, we compute **KD** (keypoint difference w/ ground-truth), and **Time** (synthesis time in seconds).

Method	GolfDB [33]		MotiCon	
	KD↓	Time↓	KD↓	Time↓
Motion Programs [23]	1.171%	48.82	0.575%	57.65
PMC (Ours)	**0.529%**	**3.22**	**0.229%**	**15.92**
Ground-Truth	0%	✗	0%	✗

MotiCon. The results are in Table 2. We observe that our spline-based representation decreases KD by $\sim 57\%$. We also see significant qualitative improvement in the results as our representation can faithfully encode a large range of motions. We supply these results in our supplementary material. Our synthesis times are also lower than Motion Programs because synthesizing best-fit circular primitives is more expensive than synthesizing splines. However, spline-based representations can require more parameters to represent the primitives, at times 1.5-2x of Motion Programs.

4.3. Human Motion Description

We evaluate the task of human motion description, which is to infer the correct order of concept sequences along with their localization in unseen videos.

We use two metrics. First, we compute the average NormED (normalized edit distance), which is the edit distance of the inferred concept sequence with the ground-truth concept sequence normalized by the maximum length of the two sequences (lower the better). We define SeqAcc (sequence accuracy) as $(1 - \text{NormED}) \times 100\%$ to translate the NormED to an accuracy metric. Finally, to evaluate the localization accuracy, we compute the repetition mAP (mAP). We annotate the ground-truth action localization for the test set and evaluate the average mAP of the inferred localization for varying IoU thresholds (mAP@[.5:.95] [3], higher is better). SeqAcc reflects recognition performance while mAP reflects the localization performance.

Baselines. We compare multiple architectures with our approach. First, we compare to vanilla encoder-decoder models (prefixed seq2seq) to show that CTC-based approaches are superior for this task. For CTC-based architectures, we compare the effects of using different kinds of recurrent layers: GRU and LSTM. For all models, we implement two variants, one taking as input the pose sequences (suffixed -Pose) and second taking as input the primitive sequences (suffixed -Prim). We also study the effects of various loss terms by ablating the primitive loss $\mathcal{L}_P$ and transition loss $\mathcal{L}_T$.

Analysis. The results of the evaluation are in Table 3. We present the average over three independent runs for each model. We observe that CTC models perform better than seq2seq models for recognition. CTC models are also able to perform localization by design while seq2seq models cannot but we see that performing localization from pose level

Table 3. Evaluating human motion description. We compute the **NormED** (normalized edit distance) and **SeqAcc** (sequence accuracy) for recognition and **mAP** (repetition mAP) for localization.

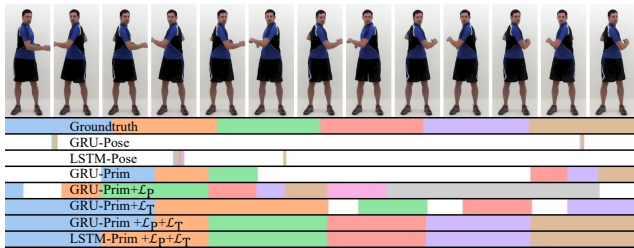| Method	NormED↓	SeqAcc↑	mAP↑
seq2seq-LSTM-Pose	0.64840	35.16%	✗
seq2seq-GRU-Pose	0.31353	68.65%	✗
seq2seq-LSTM-Prim	0.60810	39.19%	✗
seq2seq-GRU-Prim	0.59770	40.23%	✗
LSTM-Pose	0.22990	77.01%	0.00%
GRU-Pose	0.18057	81.94%	0.00%
GRU-Prim w/o $\mathcal{L}_P$ & $\mathcal{L}_T$	0.11072	88.93%	0.10%
GRU-Prim w/o $\mathcal{L}_P$	0.10303	89.70%	20.58%
GRU-Prim w/o $\mathcal{L}_T$	0.14763	85.24%	17.87%
LSTM-Prim (Ours)	0.08831	91.17%	**45.43%**
GRU-Prim (Ours)	**0.08469**	**91.53%**	38.74%

Figure 6. Visualising the localized "torso twists" concept in a video. Different colors represent different segments of the localized by models. Even if they all predict the correct number of repetitions of "torso twists", models using our primitive-based representations and $\mathcal{L}_P + \mathcal{L}_T$ finds the segments that maximally align with the ground-truth.

inputs is harder than primitive level. We also observe that our models outperform all baselines on both the recognition and localization metrics. A visualization of the localized intervals for all models is presented in Figure 6, where different colors denote different intervals located by the models. The localized intervals from our model aligns best with the ground truth, which yields more accurately localized motion descriptions and also facilitates the learning of the motion generative model. In both Table 3 and Figure 6, we study effects of ablating our loss terms. We see that both losses contribute to improved performance, performing the best when used in combination.

4.4. Action-Conditioned Motion Synthesis

We evaluate our concept representation on action-conditioned motion synthesis. We use the localized segments from our learnt description model to learn our concept representation. We sample this representation to generate novel instances from individual action classes.

As in Guo et al. [15], we evaluate on four quantitative metrics that together try to capture the correctness and diversity of the synthesized motions. We compute the i) FID score between the generated motions and real motions, ii)

Table 4. Evaluating action-conditioned motion synthesis. We compare the **FID** score, **Acc** (recognition accuracy), **Div** (variance across action classes) and **MM** (variance within action classes).

Method	FID↓	Acc↑	Div→	MM→
Real motions	0.220	98.72%	8.363	1.891
Act2Mot-2D [15]	23.465	84.39%	7.748	**2.001**
Act2Mot-2D-Prim	16.971	83.49%	7.872	2.794
PMC (Ours)	**2.406**	**97.48%**	**8.182**	1.141

Table 5. Evaluating controlled motion synthesis. We compare the **APE** (average position error) and **AVE** (average variance error).

Method	APE↓	AVE↓
Lin et al. [27]	0.2470	0.0457
JL2P [1]	0.2504	0.0440
Lin et al.-Prim	0.2075	0.0359
JL2P-Prim	0.2052	0.0358
PMC (Ours)	**0.1531**	**0.0174**

recognition accuracy (Acc) of the generated motions, iii) diversity (Div) defined as the variance of generated motion across action classes (closer to real motions is better), and iv) multimodality (MM) defined as the variance of generated motions within each action class (closer to real motions is better). We use a RNN-based action classifier as our feature extractor to compute these metrics. More details on the metrics can be found in the supplementary materials.

Baselines. We compare our method with Act2Mot-2D which is a 2D variant of Action2Motion [15] without their Lie algebra variant (as it is 3D-specific). Both the baselines and our method take the same localized segments as input data. For the baseline we implement two variants, one predicting the pose sequences and second predicting the primitive sequences (suffixed -Prim).

Analysis. The results of the evaluation are in Table 4. We observe that our method has lower FID score and higher Acc value conveying that the synthesized motions closely resemble the real motion and can be well-classified to the correct classes. Our method also produces diverse motion across classes (Div) but produces less diversity within an action class (MM) compared to the baseline, due to our hyperparameter choice that controls variance in model fitting. One can trade-off a lower value of Acc for improved variance. More details are provided in the model fitting section of the supplementary material.

4.5. Controlled Motion and Video Synthesis

We evaluate controlled motion and video synthesis given descriptions. For the test set, we take in video-level labels and synthesize the pose and video sequence.

As in prior works [1,14], we evaluate the motion synthesis on two key metrics: i) Average Position Error (APE), which measures the positional difference between the generated pose sequence with the ground-truth pose sequence (lower

Table 6. Evaluating controlled video synthesis. We compare the **PSNR**, **SSIM**, **LPIPS** and **MSE** metrics for the generated videos.

Method	PSNR↑	SSIM↑	LPIPS↓	MSE↓
Lin et al. [27]	17.791	0.925	0.112	1230.732
JL2P [1]	17.898	0.926	0.109	1192.657
Lin et al.-Prim	18.574	0.930	0.094	1067.636
JL2P-Prim	18.637	0.931	0.093	1067.194
PMC (Ours)	**19.355**	**0.934**	**0.085**	**916.898**

the better), and ii) Average Variance Error (AVE), which measures the difference in variance of the generation with the ground-truth (lower the better).

We train two skeleton-to-human GANs [5] to synthesize realistic videos from the generated pose sequence for all models. We compute the PNSR, SSIM, LPIPS and MSE metrics to judge the resulting video quality.

Baselines. We compare against two methods: i) Lin et al. [28], which first trains an autoencoder to learn a human motion model followed by an encoder-decoder model to translate description to pose sequences, ii) JL2P [1], which trains an encoder-decoder model with a joint embedding space for descriptions and poses. Additionally, we also modify both methods to predict primitive sequences (denoted with -Prim) for a stronger baseline.

Analysis. Motion synthesis evaluation results are in Table 5. We observe that our model outperforms the baselines on both APE and AVE by a fair margin, showing that the generated motion sequence matches the prompted description more faithfully. We also note that augmenting the baselines with our spline primitives improves performance. The results of video synthesis evaluations are in Table 6. Similar to motion synthesis, we observe that video synthesized from our method outperforms the baselines on all metrics. Qualitative visualizations are provided in the supplemental material.

5. Conclusion

We present a single framework for human motion description and synthesis via programmatic motion concepts. We show that our concept representation can be learnt in a data-efficient manner and is effective on multiple applications. We present a brief discussion on limitations and societal impacts of our work in the supplementary material.

References

[1] Chaitanya Ahuja and Louis-Philippe Morency. Language2pose: Natural language grounded pose forecasting. In *2019 International Conference on 3D Vision (3DV)*, pages 719–728. IEEE, 2019. 1, 3, 8

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. *arXiv preprint arXiv:2103.15691*, 2021. 2

[3] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 961–970, 2015. 2, 7

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 1, 2

[5] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5933–5942, 2019. 1, 3, 8

[6] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1130–1139, 2018. 2

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*, 2014. 4

[8] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. In *CVPR*, 2015. 2

[9] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Josh Tenenbaum. Learning to Infer Graphics Programs from Hand-Drawn Images. In *NeurIPS*, 2018. 3

[10] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: Regional multi-person pose estimation. In *ICCV*, 2017. 4

[11] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 203–213, 2020. 1

[12] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019. 1

[13] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015. 3

[14] Anindita Ghosh, Noshaba Cheema, Cennet Oguz, Christian Theobalt, and Philipp Slusallek. Synthesis of compositional animations from textual descriptions. *arXiv preprint arXiv:2103.14675*, 2021. 1, 3, 8

[15] Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong, and Li Cheng. Action2motion: Conditioned generation of 3d human motions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2021–2029, 2020. 1, 3, 7, 8

[16] Ikhsanul Habibie, Daniel Holden, Jonathan Schwarz, Joe Yearsley, and Taku Komura. A recurrent variational autoencoder for human motion synthesis. In *28th British Machine Vision Conference*, 2017. 1, 3

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, 1997. 4

[18] 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. 3

[19] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, 2017. 2

[20] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-Scale Video Classification with Convolutional Neural Networks. In *CVPR*, 2014. 1

[21] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *SIGGRAPH*, 2008. 3

[22] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE, 2011. 1, 2

[23] Sumith Kulal, Jiayuan Mao, Alex Aiken, and Jiajun Wu. Hierarchical motion understanding via motion programs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6568–6576, June 2021. 1, 3, 4, 6, 7

[24] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. GRASS: Generative Recursive Autoencoders for Shape Structures. *ACM TOG*, 2017. 3

[25] Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. Learn to dance with aist++: Music conditioned 3d dance generation, 2021. 3

[26] Zimo Li, Yi Zhou, Shuangjiu Xiao, Chong He, Zeng Huang, and Hao Li. Auto-conditioned recurrent networks for extended complex human motion synthesis. *arXiv preprint arXiv:1707.05363*, 2017. 3

[27] Angela S Lin, Lemeng Wu, Rodolfo Corona, Kevin Tai, Qixing Huang, and Raymond J Mooney. Generating animated videos of human activities from natural language descriptions. *Learning*, 2018:1, 2018. 1, 3, 8

[28] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction. In *AAAI*, 2018. 8

[29] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 2

[30] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences from Natural Supervision. In *ICLR*, 2019. 3

[31] Jiayuan Mao, Xiuming Zhang, Yikai Li, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. Program-Guided Image Manipulators. In *ICCV*, 2019. 3

[32] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2891–2900, 2017. 3

[33] William McNally, Kanav Vats, Tyler Pinto, Chris Dulhanty, John McPhee, and Alexander Wong. Golfdb: A video database for golf swing sequencing. In *CVPR Workshop*, June 2019. 6, 7

[34] Jianyuan Min and Jinxiang Chai. Motion graphs++ a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics (TOG)*, 31(6):1–12, 2012. 3

[35] Phuc Nguyen, Ting Liu, Gautam Prasad, and Bohyung Han. Weakly supervised action localization by sparse temporal pooling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6752–6761, 2018. 3

[36] Juan Carlos Niebles and Li Fei-Fei. A Hierarchical Model of Shape and Appearance for Human Action Classification. In *CVPR*, 2007. 1

[37] Mathis Petrovich, Michael J Black, and Gül Varol. Action-conditioned 3d human motion synthesis with transformer vae. *arXiv preprint arXiv:2104.05670*, 2021. 1, 3

[38] Matthias Plappert, Christian Mandery, and Tamim Asfour. The kit motion-language dataset. *Big data*, 4(4):236–252, 2016. 3

[39] Matthias Plappert, Christian Mandery, and Tamim Asfour. Learning a bidirectional mapping between human whole-body motion and natural language using deep recurrent neural networks. *Robotics and Autonomous Systems*, 2018. 3

[40] Jian Ren, Menglei Chai, Sergey Tulyakov, Chen Fang, Xiaohui Shen, and Jianchao Yang. Human motion transfer from poses in the wild. In *European Conference on Computer Vision*, pages 262–279. Springer, 2020. 1, 3

[41] Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? *arXiv preprint arXiv:2106.11297*, 2021. 2

[42] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. CSGNet: Neural Shape Parser for Constructive Solid Geometry. In *CVPR*, 2018. 3

[43] Eli Shlizerman, Lucio Dery, Hayden Schoen, and Ira Kemelmacher-Shlizerman. Audio to body dynamics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7574–7583, 2018. 1

[44] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1049–1058, 2016. 1

[45] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in Neural Information Processing Systems*, 32:7137–7147, 2019. 1

[46] Karen Simonyan and Andrew Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *NeurIPS*, 2014. 1, 2

[47] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A Dataset of 101 Human Actions Classes from Videos in the Wild. *arXiv:1212.0402*, 2012. 1, 2

[48] Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. Learning to Infer and Execute 3D Shape Programs. In *ICLR*, 2019. 3

[49] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 2

[50] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing Motion and Content for Video Generation. In *CVPR*, 2018. 1

[51] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4305–4314, 2015. 2

[52] Limin Wang, Yuanjun Xiong, Dahua Lin, and Luc Van Gool. Untrimmednets for weakly supervised action recognition and detection. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4325–4334, 2017. 3

[53] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 1

[54] Xiang Wang, Zhiwu Qing, Ziyuan Huang, Yutong Feng, Shiwei Zhang, Jianwen Jiang, Mingqian Tang, Changxin Gao, and Nong Sang. Proposal relation network for temporal action detection. *arXiv preprint arXiv:2106.11812*, 2021. 2

[55] 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. 2

[56] Halley Young, Osbert Bastani, and Mayur Naik. Learning Neurosymbolic Generative Models via Program Synthesis. In *ICML*, 2019. 3

[57] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. Graph convolutional networks for temporal action localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7094–7103, 2019. 1

[58] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2914–2923, 2017. 2