Searching Motion Graphs for Human Motion Synthesis

Chenchen Liu, Yadong Mu*
{liuchenchen,myd}@pku.edu.cn
Peking University
Beijing, P.R. China

ABSTRACT

This work proposes a graph search based method for human motion sequence synthesis, complementing the modern generative model (*e.g.*, variational auto-encoder or Gaussian process) based solutions that currently dominate this task and showing strong advantages at several aspects. The cornerstone of our method is a novel representation which we dub as *motion graph*. Each motion graph is scaffolded by a set of realistic human motion sequences (*e.g.*, all training data in the Human3.6M benchmark). We devise a scheme that adds transition edges across different motion sequences, enabling more longer and diverse routes in the motion graph. Crucially, the proposed motion graph bridges the problem of human motion synthesis with graph-oriented combinatorial optimization, by naturally treating pre-specified starting or ending pose in human pose synthesis as end-points of the retrieved graph path. Based on a jump-sensitive graph path search algorithm proposed in this paper, our model can efficiently solve human motion completion over the motion graphs. In contrast, existing methods are mainly effective for human motion prediction and inadequate to impute missing sequences while jointly satisfying the two constraints of pre-specified starting / ending poses. For the case of only specifying the starting pose (*i.e.*, human motion prediction), a forward graph walking from the starting node is first performed to sample a diverse set of ending nodes on the motion graph, each of which defines a motion completion problem. We conduct comprehensive experiments on two large-scale benchmarks (Human3.6M and HumanEva-I). The proposed method clearly proves to be superior in terms of several metrics, including the diversity of generated human motion sequences, affinity to real poses, and cross-scenario generalization etc.

CCS CONCEPTS

• **Information systems** → **Multimedia content creation**.

KEYWORDS

Human pose synthesis, motion graph, neural networks, graph search

*Corresponding author

MM '21, October 20–24, 2021, Virtual Event, China.

ACM ISBN 978-1-4503-8651-7/21/10…$15.00
https://doi.org/10.1145/3474085.3475264

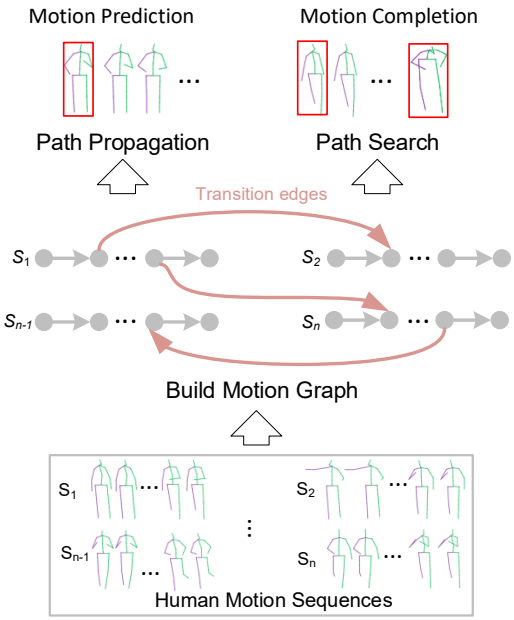

Figure 1: Overview of our method. The key idea is constructing a motion graph from data. The graph includes the adjacent node edges within the sequence and the transition edges between the sequences. We propose a generative model to render transition edges. Both motion completion and motion prediction can be conducted on the graph.

ACM Reference Format:
Chenchen Liu, Yadong Mu. 2021. Searching Motion Graphs for Human Motion Synthesis. In *Proceedings of the 29th ACM Int'l Conference on Multimedia (MM '21), Oct. 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474085.3475264

1 INTRODUCTION

The generation of realistic human motion sequences, including human motion completion and prediction, is an important task in video games and movies. It can be used to aid human-centric video generation [10, 47] and has wide applications in 3D character animations, human robot interaction [27], and human tracking [14] etc.

Existing works mainly focus on human motion prediction [4, 8, 9, 13, 35, 36], which aim at predicting near-future sequences given a few seen frames. Mainstream human motion prediction methods can be divided into two types: data-driven and generative model. Traditional data-driven approaches for human motion prediction, such as Hidden Markov Model [7], Gaussian Process latent variable models [43] and graph-based models [3, 28, 31], have proved effective for simple periodic motion and acyclic motions, such as walking and golf swing. With the development of deep learning [11, 21, 23,

30, 40], deep neural network models [8, 13, 18, 24, 33, 35, 36, 44] are used to tackle complicated motion sequences generation. Because of the temporal nature of the signal of interest, Recurrent Neural Networks (RNNs) methods [13, 18, 24, 36] are found to be effective for motion generation. However, existing works [18, 33] that often feed the estimation at specific RNN step as the input to the next prediction tend to have accumulated errors throughout the generated sequence, leading to unrealistic prediction at the inference time. Recently, several works [8, 33, 35] have proposed to adopt feed-forward networks for motion generation. They can generate more accurate sequences compared with RNN based methods, but are limited to short-term motion generation. The performance of long-term motion generation drops rapidly.

In this paper, we propose a unified method for tackling both human motion completion and prediction. The core of the proposed method is a novel directed graph based representation that encodes rich knowledge about human dynamics, which we term as *motion graph*. The conceptual pipeline is illustrated in Figure 1. Given a full set of human motion sequences (such as the training data in Human3.6M or HumanEva-I), tremendous key frames are first extracted from the sequences at some regular intervals, which are treated as the nodes in the motion graph. The temporal order of these frames naturally defines the directed structure of the graph. To enable the traverse among different sequences, we further propose to add *transitions edges* between the sequences, which is accomplished through transitive node selection (Section 3.1.3) and the generation of virtual sub-sequences that connect two transitive nodes (Section 3.2). Based on the constructed motion graph, for the human motion completion task, we develop a jump-sensitive graph search algorithm to find and rank multiple feasible paths from a starting node to an ending node, thereby generating a diverse set of motion completion results. The task of human motion prediction only specifies the starting poses, which are insufficient to initiate a graph-path search. We devise a forward graph walking from the starting nodes that returns multiple possible destinations. In this way, the human motion prediction boils down to a series of human motion completion with different ending nodes.

Our key technical contributions can be summarized in two-folds:

1) We introduce a new data representation called motion graph and an accompanying graph search scheme for generating diverse and realistic human motion sequences, either from single-end input (*i.e.*, human motion prediction) or dual-end input (*i.e.*, human motion completion). The proposed method is essentially different from existing generative model based motion-synthesizing methods, and admits both algorithmic simplicity and generality for various human motion related tasks;

2) We provide in-depth empirical evaluations on two large-scale benchmarks (Human3.6M and HumanEva-I). On the rarely-explored human motion completion task, the proposed method surpasses other methods by large margins in terms of diversity and reality. For human motion prediction, our method produces comparable performances under various metrics to current state-of-the-art work (*e.g.*, DLow [48]). When evaluated under cross-scenario settings (for example, test a model trained on Human3.6M using the data from HumanEva-I), our method clearly demonstrates more excellent generalization ability under all metrics, which sheds light on the development of more robust human motion synthesis methods using data-driven search.

2 RELATED WORK

Motivated by their success in sequence-to-sequence prediction [26, 41], RNNs have become the commonly used model for human motion prediction [13, 24, 36]. For example, Fragkiadaki *et al.* [13] firstly proposed an *Encoder-Recurrent-Decoder (ERD)* model that incorporates a nonlinear encoder and decoder before and after recurrent layers. In [24], Jain *et al.* proposed to further encode the spatial and temporal structure of the pose prediction problem via a *Structure-RNN* model relying on high-level spatio-temporal graphs. While the two previous methods directly estimated absolute human poses, Matinez *et al.* [36] introduced a residual architecture to predict velocities. Interestingly, it was shown in this work that a simple zero-velocity baseline, *i.e.*, constantly predicting the last observed pose, led to better performance than [13, 24]. While [36] outperformed this baseline, the predictions produced by RNN still suffer from discontinuities between the observed poses and the predicted future ones.

Feed-forward networks, such as fully-connected and convolutional ones, were studied as an alternative solution to avoid the discontinuities produced by RNNs [8, 33]. In particular, in [8], Butepage *et al.* proposed to treat a recent pose history as the input to a fully-connected network, and introduced different strategies to encode additional temporal information via convolutions and spatial structure of the kinematic tree. In [35], Mao *et al.* proposed to use DCT to encode temporal information and use GCNs [25] to encode spatial structure. Their feed-forward network was proven highly effective for the human motion prediction task. For all these methods, they can only output one prediction for the same input, and lack the diversity of predictions. Our method can generate diverse results at different scales.

For human motion synthesis task, the Generative Adversarial Networks (GAN) [17] are also widely used in the literature. HP-CAN [4] combines the Seq2seq model and GAN for motion prediction, where the Seq2seq model is used as the generator and a fully connected network is used as a discriminator. Cai *et al.* [9] proposed a two-stage GAN for skeleton motion generation, where the first stage learns to generate spatial signals of pose, and the second stage generates temporal signals represented as latent vector sequences. Yan *et al.* [44] proposed a Convolutional Sequence Generation Network (CSGN) to generate the entire sequence altogether by transforming from a sequence of latent vectors sampled from a Gaussian process and model the structures in temporal and spatial dimensions. Motion-based graphs or motion-based database methods for motion synthesis were previously proposed in [3, 28, 31]. However, these methods are limited to generate some very simple actions, such as human walking and running. Distinguished from above-mentioned methods, we develop the motion graph with transition edges, generating more complex and realistic motion sequences with improved diversity.

3 THE PROPOSED METHOD

As shown in Figure 2, we decompose the proposed method into three stages: *motion graph construction*, *transition motion generation*

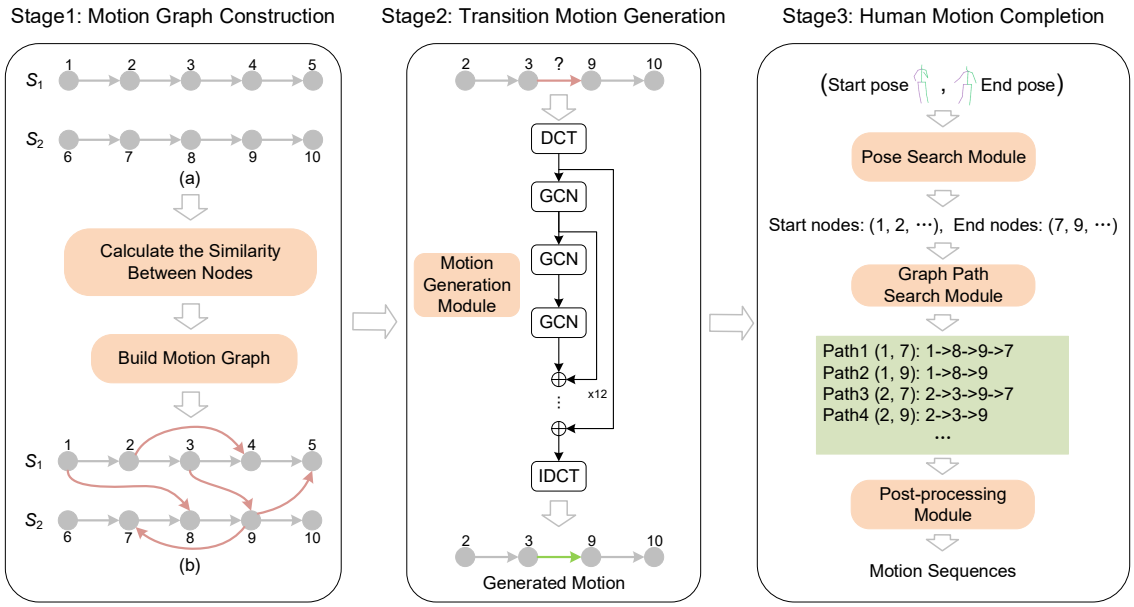

Figure 2: The computational pipeline of our proposed method. See main text for more explanation

and *human motion completion* (recall that in our proposed framework the human motion prediction tasks boils down to multiple executions of the motion completion routine). Stage one constructs a motion graph from the training dataset. This graph is used to encode the motion dynamics of various human actors within the same sequence and the smooth transfer between different sequences. The graph nodes involved in the inter-sequence transitions are chosen in this step, and the corresponding virtual routes are determined in the following step. In specific, we adopt a modified DCT-GCN [35] to generate virtual motions for the transition edges. Stage three then uses a graph path search algorithm to find numerous feasible paths from the starting node to the ending node in the graph, obtaining diverse results of motion completion. Finally, we use Gaussian filtering to post-process the motion sequences for taming some noise-corrupted human poses.

3.1 Motion Graph Construction

3.1.1 Topology of the motion graph. In this section, we define the topological structure of the motion graph and the procedure of constructing motion graph from a database of human motion sequences. A human motion sequence is regarded as a regular sampling of the character's parameters, such as the angles or 3D coordinates of each skeleton joint. Typically, such a sequence is part of some daily actions, such as walking, talking on the phone, greeting, etc. Suppose we are given a human motion database $\mathcal{D} = \{S_1, S_2, S_3, ..., S_N\}$ that contains N human motion sequences. Each sequence $S_i = \{x_{i,1}, x_{i,2}, x_{i,3}, ..., x_{i,T}\}$ consists of T consecutive human poses. Let $x_{i,t} \in \mathbb{R}^K$, where K is the number of parameters describing each pose.

Our goal is to construct a motion graph $\mathcal{G}$ from $\mathcal{D}$. $\mathcal{G}$ is a directed graph where all edges correspond to some sub-sequences of human motion. For each sequence S_i, we sample one frame at a temporal stride of I frames[1] and assign a node for it in the motion graph. Importantly, there are two kinds of edges in the motion graph. If we only create an edge between two adjacent nodes within a same motion sequence, it leads to about $N(L/I)$ nodes and $N(L/I-1)$ edges where L is the average length of all sequences., as shown in Figure 2(a). Such a graph does not have cross-sequence connections, therefore it is trivial for our interested tasks. In order to diversity the graph routes across different motion sequences, we add more transition edges among motion sequences and seek for a motion graph with enhanced connectivity. The idea is illustrated in Figure 2(b). More details will be presented in Section 3.1.3.

3.1.2 Calculation of graph-node distance. For an arbitrary graph node pair $< S_{i,t}, S_{j,t'} >$, we combine two considerations as below (*i.e.*, pose and motion distance) when determining the similarity between them.

Pose distance. For the node pair $< S_{i,t}, S_{j,t'} >$, let $x_{i,t}$ and $x_{j,t'}$ be their vectorized pose representation, respectively. It is important to align two poses before comparing them. To this end, The Procrustes Analysis[2] is firstly used to align $x_{i,t}$ and $x_{j,t'}$. Without loss of generality, let us freeze $x_{i,t}$ and transform the other pose. The pose distance is computed as the Euclidean distance between the aligned poses, namely

$$x_{i,t}, x^{\star}_{j,t'} = \text{Procrustes\_Analysis}(x_{i,t}, x_{j,t'}),\ d_{pose} = \left\| x_{i,t} - x^{\star}_{j,t'} \right\|_2 . \tag{1}$$

Motion distance. Visually similar poses shall be distinguished according to the instantaneous moment-level human motion dynamics. To this end, we take 10 frames forward to get the adjacent sub-sequence of each node. For example, the sub-sequence to $S_{i,t}$

[1]Taking into account the scale of the motion graph and the requirements of the generative model for the known sequence length, we set I to 20.
[2]https://en.wikipedia.org/wiki/Procrustes_analysis

is $\{x_{i,t-9}, x_{i,t-8}, ..., x_{i,t}\}$. We adopt a simple strategy to estimate a local motion vector as below:

$$m_{i,t} = \{x_{i,t-8} - x_{i,t-9}, x_{i,t-7} - x_{i,t-8}, ..., x_{i,t} - x_{i,t-1}\}.$$

Likewise the motion vector $m_{i,t'}$ for $S_{i,t'}$ can be also computed. A Cosine-form distance is adopted to gauge the pair of motion vectors, namely

$$d_{mot} = 1 - \frac{m_{i,t}^T m_{j,t'}}{\|m_{i,t}\|_2 \cdot \|m_{j,t'}\|_2} \tag{2}$$

We fuse the above two metrics d_{pose} and d_{mot} to get the final distance between nodes $S_{i,t}$ and $S_{j,t'}$,

$$d = \alpha * \frac{d_{pose}}{D_{pose}} + (1 - \alpha) * \frac{d_{mot}}{D_{mot}}, \tag{3}$$

where D_{pose} and D_{mot} represent some maximal value of pose distance and motion distance for normalizing purpose, respectively. α is the harmonic parameter and we set it as 0.7 without further fine-tuning.

3.1.3 Submodular optimization for transition node selection. A transition node in $\mathcal{G}$ is designed to establish a virtual path between two motion sequences or a skip connection within a motion sequence. The specific choice of transition nodes crucially affects the quality of the final motion graph. Insufficiently-sampled transition nodes lead to low graph connectivity and excessive transitions make the graph paths fragmented. In addition, to ensure a transition to be smooth, the cost of linking a selected transition node with other nodes in the graph should be reasonably small. Formally, given a fully connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we aim to find a subset $\mathcal{A} \in \mathcal{V}$ such that the cost of all other nodes in the graph to this subset is as small as possible. This problem can be efficiently solved via submodular optimization [29].

Let us treat the graph nodes outside the subset $\mathcal{A}$ as queries, and formally model these querying nodes with a random process. With each node $s \in \mathcal{V}$ in the graph and cost $c \in \mathcal{C}$ ($\mathcal{C} \subseteq \mathbb{R}$ is a discrete or continuous ordered set of cost), we associate a nonnegative random variable $\mathcal{X}_{s,c}$, which is 0 if there is no path from query node to s with a cost c, and positive otherwise. Hence, the joint realization $\mathbf{x}_{\mathcal{V},\mathcal{C}}$ describes the state of the graph over the entire course of a query. For each possible realization, we evaluate a penalty function $\pi(\mathbf{x}_{\mathcal{V}}, \mathcal{A})$ which depends on the state of the graph $\mathbf{x}_{\mathcal{V},\mathcal{C}}$ at all nodes and all cost, as well as the subset $\mathcal{A}$. For each possible observation (selected graph node) $s \in \mathcal{V}$, we use

$$C(x_{\mathcal{V},\mathcal{C}}, s) = \min\{c \in \mathcal{C} : \mathbf{x}_{s,c} > 0\} \tag{4}$$

to denote the *cost of query*. Hereby, $C(\mathbf{x}_{\mathcal{V},\mathcal{C}}, s) = \infty$ if there is no path from query node to s, *i.e.*, $\mathbf{x}_{s,c} = 0$ for all $c \in \mathcal{C}$. For a set of observations $A \subseteq \mathcal{V}$, we define

$$C(X_{\mathcal{V},\mathcal{C}}, \mathcal{A}) = \min_{s \in \mathcal{A}} C(\mathbf{x}_{\mathcal{V},\mathcal{C}}, s), \tag{5}$$

i.e., the minimum cost from query node to any selected node in $\mathcal{A}$. Furthermore, we define a function $\pi(\mathbf{x}_{\mathcal{V},\mathcal{C}}, c) = \min(c, C_{\max})$, where $C_{\max}$ is the maximum cost of the dataset. π describing the penalty incurred if $\mathbf{x}_{\mathcal{V},\mathcal{C}}$ exists path for query node at cost c.

Based on above notations, we can formulate node querying formally as an observation selection problem. Define a quality function

$$u(\mathbf{x}_{\mathcal{V},\mathcal{C}}, \mathcal{A}) = \pi(\mathbf{x}_{\mathcal{V},\mathcal{C}}, \infty) - \pi(\mathbf{x}_{\mathcal{V},\mathcal{C}}, C(\mathbf{x}_{\mathcal{V},\mathcal{C}}, \mathcal{A})). \tag{6}$$

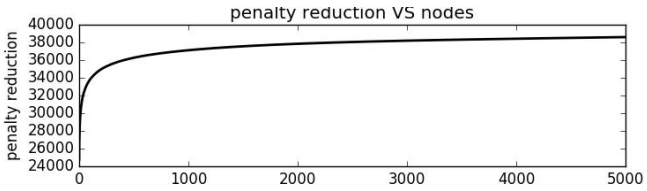

Figure 3: Convergence curve of the penalty reduction.

If we assume a probability distribution $P(\mathbf{x}_{\mathcal{V},\mathcal{C}})$ over query nodes, the corresponding expected quality function

$$F(\mathcal{A}) = \int P(\mathbf{x}_{\mathcal{V},\mathcal{T}})\, u(\mathbf{x}_{\mathcal{V},\mathcal{C}}, \mathcal{A})\, d\mathbf{x}_{\mathcal{V},\mathcal{C}}, \tag{7}$$

then models the expected penalty reduction obtained if observing subset $\mathcal{A} \subseteq \mathcal{V}$ of the nodes.

Using this notion of quality function, our goal is to solve the maximization problem: $\max_{\mathcal{A} \subseteq \mathcal{V}} F(\mathcal{A})$ s.t. $|\mathcal{A}| \leq k$, where k is the selected node number. The penalty reduction function $F(\mathcal{A})$ has several important and intuitive properties: Firstly, $F(\emptyset) = 0$, i.e., we do not reduce the penalty if not selecting any nodes. Secondly, F is non-decreasing, i.e., $F(\mathcal{A}) \leq F(\mathcal{B})$ for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$. Hence, adding nodes for $\mathcal{A}$ can only decrease the incurred penalty. Thirdly, and most importantly, it is submodular [29]. For all choice $\mathcal{A} \subseteq \mathcal{B} \subseteq V$ and nodes $s \in \mathcal{V} \backslash \mathcal{B}$, it holds that $F(\mathcal{A} \cup \{s\}) - F(\mathcal{A}) \leq F(\mathcal{B} \cup \{s\}) - F(\mathcal{B})$. Hence, if we add a node to a small set $\mathcal{A}$, we improve our score as least as much, as if we add it to a larger set $\mathcal{B} \supseteq \mathcal{A}$. In [29], it proved that the greedy algorithm applied to penalty reduction objective $F(\mathcal{A})$ is guaranteed to obtain a solution of the maximization problem which achieves at least a constant fraction of $(1 - 1/e)$ of the value obtained by the optimal solution. The pseudo code of the algorithm is in the supplementary materials.

The convergence curve of $\mathcal{F}(\mathcal{A})$ is shown as Figure 3. As the number of the selected nodes increases, the penalty reduction gradually stabilizes. We finally select 4000 nodes to form the set $\mathcal{A}$ as the transition nodes. Then, we use k-nearest neighbors to select nodes with the smallest distances for each transition node to establish edges. In experiments 5-NN is adopted.

3.2 Transition Motion Generation

The poses of two connected transition nodes can be subtly different and perceptible for human. The generation of transition sub-sequences aims to fill in this gap. It can be regarded as a tiny human motion completion problem: the sub-sequences of two transition nodes are known, which can be conceptually used as starting / ending poses. Transition sequence generation reads the known two sub-sequences and imputes the "missing" ones in-between.

Inspired by [35], we tackle transition motion generation by jointly encoding temporal information, via the use of the Discrete Cosine Transform (DCT), and spatial structure, via Graph Convolutional Networks (GCNs) with learnable connectivity. Let us denote by $\tilde{x_k} = \{x_{k,1}, x_{k,2}, x_{k,3}, ..., x_{k,T}\}$ the trajectory for the k^{th} joint across T frames. Given a trajectory $\tilde{x_k}$, the corresponding l^{th} DCT coefficient can be computed as

$$C_{k,l} = \sqrt{\frac{2}{T}} \sum_{t=1}^{T} x_{k,t} \frac{1}{\sqrt{1 + \delta_{l1}}} \cos(\frac{\pi}{2T}(2t - 1)(l - 1)), \tag{8}$$

where δ_{ij} denotes the *Kronecker* delta function with $\delta_{ij} = 1$ if i is equal to j, otherwise 0.

In practice, $l \in \{1, 2, .., T\}$. In short, DCT encoding allows us to model the temporal information of each joint using DCT coefficients. Given such coefficients, the original human pose representation can be obtained via the Inverse Discrete Cosine Transform (IDCT) as

$$x_{k,t} = \sqrt{\frac{2}{T}} \sum_{l=1}^{T} C_{k,l} \frac{1}{1+\delta_{l1}} \cos(\frac{\pi}{2T}(2t-1)(l-1)), \tag{9}$$

where $t \in \{1, 2, 3, ..., T\}$.

After getting the DCT encoding of the sequence, we use GCN to encode the spatial structure of human pose. Let us assume that the human body is modeled as a fully-connected graph with K nodes. The strength of the edges in this graph can then be represented by a weighted adjacency matrix $\mathbf{A} \in \mathbb{R}^{K \times K}$. A graph convolutional layer p then takes as input a matrix $\mathbf{H}^{(p)} \in \mathbb{R}^{K \times F}$, with F the number of features output by the previous layer. Given this information and a set of trainable weights $\mathbf{W}^{(p)} \in \mathbb{R}^{F \times F}$, a graph convolutional layer outputs a matrix of the form $\mathbf{H}^{p+1} = \sigma(\mathbf{A}^{(p)}\mathbf{H}^{(p)}\mathbf{W}^{(p)})$, where $\mathbf{A}^{(p)}$ is the trainable weighted adjacency matrix for layer p and $\sigma(\cdot)$ is an activation function.

3.3 Jump-Sensitive Graph Path Search for Human Motion Completion

3.3.1 Pose search. For the human motion completion or prediction problems, the computation initiates from finding a few nodes in the motion graph $\mathcal{G}$ that are similar to a given starting or ending pose. We call this process *Pose Search*. Pose search are accomplished by two operations: pose alignment (*i.e.*, find a geometric transform that optimally matches two poses) and pose distance calculation.

The motion graph $\mathcal{G}$ is often comprised of tremendous vertices owing to the dense sampling over motion sequences. As stated before, it is important to precisely align two human poses before calculating their similarity. A naive linear scan scheme that separately aligns each human pose in $\mathcal{G}$ with respect to the querying poses will trigger intolerably high complexity. To expedite the pose search process, this work adopts a divide-and-conquer strategy for pre-indexing all information in the motion graph $\mathcal{G}$. The key insight is that human poses tend to be locally clustered with each other in some feature space. Finding the most similar poses for a query can thus be approximately conducted only within the cluster where the query resides.

Given a motion graph $\mathcal{G}$ with a total of N nodes, we calculate the Euclidean distance after each two nodes are aligned to obtain a distance matrix D. In D, 0 means identical elements, and high values mean very dissimilar elements. It can be transformed in a similarity matrix by applying the Gaussian kernel,

$$S = \exp(-D^2/(2 * \delta^2), \tag{10}$$

where δ is a free parameter representing the width of the Gaussian kernel and can be empirically estimated from the training data. Next, we perform the spectral clustering algorithm on the similarity matrix S, getting C clusters. For each cluster, a center pose x_i of cluster i is computed by

$$x_i = \text{argmin}_{x_i \in C_i} \sum_{x \in C_i} d_{pose}(x_i, x), \tag{11}$$

where C_i is the set of all poses of cluster i. $d_{pose}(\cdot)$ is defined in Equation 1. After choosing x_i, we align all poses in the cluster i with respect to x_i.

To further speed up the pose search process, we conduct vector-based hashing separately in each cluster. We follow the ITQ algorithm [15] to perform binary quantization. For specific cluster, assume there are n poses $\{x_1, x_2, ..., x_n\}, x_i \in \mathbb{R}^d$, forming a data matrix $X \in \mathbb{R}^{n \times d}$. Our goal is to learn a binary code matrix $B \in \{0, 1\}^{n \times h}$, where h denotes the binary code length. For the first step, PCA (principal component analysis) is adopted to project the pose data X to some $V \in \mathbb{R}^{n \times c}$. Following [15, 16], the binary hashing code matrix B can be estimated by optimizing the quantization loss:

$$Q(B, R) = \|B - VR\|_F^2, \tag{12}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and R is an orthogonal $c \times c$ matrix. After an optimal R^* is pursued, the binary code matrix $B = \text{sgn}(VR^*)$, where $sng(\cdot)$ is a binary encoding function. $\text{sgn}(v) = 1$ if $v \geq 0$ and 0 otherwise.

For C clusters, we get C binary code matrix $\{B_1, B_2, ..., B_C\}$. Each node in the motion graph $\mathcal{G}$ now has a uniuqe cluster index and a corresponding compact hash code. For a given query pose p, we first calculate the distance between the query pose and the center of each cluster, which identifies the closest cluster. Afterwards, the query's hash code within the closest cluster is calculated. Using the code, it is able to quickly retrieve the most similar poses from the cluster based on Hamming distance (*i.e.*, the number of discrepancy in two binary codes) ranking.

3.3.2 Graph search. A human motion completion instance is defined by a starting (or source) pose x_s and an ending (or target) pose x_t. The goal is to faithfully guess the intermediate motion between x_s and x_t. Using the aforementioned *pose search*, we can find a corresponding node set $\mathcal{S}$ in the motion graph for x_s, and $\mathcal{T}$ for x_t respectively. The motion completion task can be divided into two part: first, we search the possible graph paths from all source-target pair $(s, t) \in \mathcal{S} \times \mathcal{T}$, and then restore the missing intermediate motion from x_s to x_t according to the paths.

The transition nodes, despite enhancing the graph connectivity, have the side effect of rendering non-smooth sub-sequences. We use the term *jump* to refer to the transition from one motion sequence to another. Fewer jumps on a graph path tends to correspond to more realistic human motion sequences, yet have limited diversity. It is easy to know that for a directed acyclic graph, the time complexity of finding all paths from source node to target node is $O(2^N)$, where N is the nodes number in the graph. And for the path search with the specific number of jumps, the time complexity is $O(N^J)$, J is the number of jumps. The search for all s-t paths in the graph is an NP-Hard problem, which can not be solved in polynomial time. Since we do not need to find all possible paths in most practical applications, we propose a heuristic approximate search algorithm, which has polynomial time complexity while ensuring the search effect.

For the path search problem, one of the most widely-used algorithms is breadth-first search (BFS). However, its search space expands exponentially as the number of jumps increases. Here we propose a heuristic-guided strategy to filter out unreasonable nodes in the search space. The Floyd-Warshall algorithm is first performed to calculate a minimum-jump matrix D between all nodes. Suppose the goal is to find the paths with a given number of jumps j^* from some graph node $s \in \mathcal{S}$ to another node $t \in \mathcal{T}$. Assume we have already obtained a path $P = \{p_1, p_2, \ldots, p_o\}$ and its jump number is j_{cur}. Let us try to determine whether the successor node p of p_o can be added to the search space. Denote the number of jumps from p_o to p as j_p, and the minimum number of jumps from p to t is pre-stored in $D_{p,t}$. If $j_{cur} + j_p + D_{p,t} > j^*$, then p is an unreasonable node and will not be added to the search space. Since we do not need all possible $s - t$ paths, we keep only Q paths in the current search space for each additional jump, so that the search space be reduced from the exponential level to the polynomial level. The time complexity of using the Floyd-Warshall algorithm to calculate the jump matrix D is $O(N^3)$, but this time cost is one-time and does not need to be calculated for every query. And the time complexity of each query of our heuristic graph search algorithm is $O(JQN)$. So the overall time complexity of the algorithm is polynomial. In experiments Q is set to 1000. The pseudo code of the algorithm is shown in supplementary materials.

After getting the graph paths from the nodes $\mathcal{S}$ to $\mathcal{T}$, we restore the corresponding motion sequences for the paths. For edges in an original human motion sequence in the database, we directly take out its corresponding motion sub-sequence from the original sequence. For transition edges between sequences, we use a generative model to generate the corresponding motion sequences. This way returns the completion result from the start pose to the end pose.

3.4 Extension to Human Motion Prediction

For human motion prediction, we only know the starting pose x_s. The goal is to forecast human motions in the future. Using *pose search*, we can find a corresponding node set $\mathcal{S}$ in the motion graph for x_s. Then we implement motion prediction in two steps. Firstly, we conduct a jump-sensitive forward propagation in the motion graph for each node $s \in \mathcal{S}$. Jump-sensitive forward propagation walks forward from the node s to get the candidate paths of the specified number of jumps J. We can generate paths of different lengths by controlling J and rank them according to the cost on the graph path. Top-ranked graph paths are kept as the final results. Secondly, we can restore the predicted motion based on the paths. In this way, human motion prediction boils down to multiple runs of our proposed human motion completion routines.

After getting the synthesized sequences of human motion completion or prediction, we use Gaussian filtering as a post-processing to remove noises. The diffusion parameter σ of the filter is set to 2 and the size of the sliding window is 10.

4 EXPERIMENTS

4.1 Dataset Description

Human3.6M. Human3.6M [38] is a large-scale dataset with 11 subjects (7 with ground truth) and 3.6 million video frames in total. Each subject performs 15 actions (walking, eating, discussion, sitting, and phoning etc.) and the human motion is recorded at 50 HZ. Following previous work [34, 36, 37, 48], we adopt a 17-joint skeleton, train the model on five subjects (S1, S5, S6, S7, S8) and test it on two rest subjects (S9 and S11).

HumanEva-I [39]. HumanEva-I contains three subjects recorded at 60 HZ. We adopt a 15-joint skeleton [37] and use the same train / test split provided by the dataset organizer.

4.2 Evaluation Protocol

4.2.1 Human motion completion. We conduct human motion completion experiments on Human3.6M dataset. Following [44], we use *Inception Score* (IS) [6] and *Fréchet Inception Distance* (FID) [22] to evaluate our proposed method. These two metrics are widely used in the image generation task. *Inception Score* feeds generated samples to a classification model and analyzes their output probabilities over all classes. *Fréchet Inception Distance* measures the distance between the statistics of real and synthesized data in some feature space. We choose ST-GCN [45] as the action classifier for calculating IS and FID. The dataset of Human3.6M contains only limited number of actions and subjects, the trained models are thus easy to overfit. The top-1 and top-5 classification accuracies of ST-GCN are 50.8% and 88.6%, respectively. Following [44], we use an extension of IS and FID. In order to calculate them, we generate $N = 1000$ sequences for each sequence length T, and cut each sequence into M short snippets of length 50. The overlap of two adjacent short snippets is 25 frames.

Mean time FID/IS scores characterize the generation quality within each generated sequence. It first obtains the time FID/IS scores for each generated sequence by computing the basic metrics on all snippets within each sequence and then averages the scores across all sequences. Take the mean time FID (denoted as FID^t) as an example,

$$\text{FID}^t = \frac{1}{N} \sum_n \text{FID}(\{\mathcal{F}^{mn}\}_{m=1,\ldots,M}). \tag{13}$$

To evaluate the effect of our method on generating sequences of different lengths, we set T to five different lengths of 1, 5, 10, 20, and 30 seconds.

4.2.2 Human motion prediction. We conduct human motion prediction experiments on both Human3.6M and HumanEva-I datasets. For Human3.6M, we predict future motion for 2 seconds based on observed motion of 0.5 seconds. For HumanEva-I, we forecast future motion for 1 second given observed motion of 0.25 seconds.

Following [48], we use the following metrics to measure both sample *diversity* and *accuracy*.

Average Pairwise Distance (APD): average $L2$ distance between all pairs of motion samples to measure diversity within samples, which is computed as

$$APD = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j \neq i}^{N} \|x_i - x_j\|, \tag{14}$$

where x is the generated sequences, N is the sequences numbers.

Average Displacement Error (ADE): average $L2$ distance over all time steps between the ground truth motion $\hat{x}$ and the closest

Method	$IS^t \uparrow$					$FID^t \downarrow$				
	1s	5s	10s	20s	30s	1s	5s	10s	20s	30s
Linear Interpolation	6.75	1.82	1.84	1.89	1.89	21.2	176.0	174.1	172.1	172.8
DCT-GCN [35]	6.82	1.87	1.87	1.88	1.96	19.7	174.2	173.8	174.4	169.3
Ours	8.00	1.45	1.88	2.46	3.25	18.2	188.8	157.0	130.3	107.4
Ours*	8.35	1.51	1.84	2.54	3.28	16.4	191.7	150.3	124.3	104.4

Table 1: Experimental results for human motion completion on Human3.6M. See main text for more explanation.

	Human3.6M [38]					HumanEva-I [39]				
Method	APD ↑	ADE ↓	FDE ↓	MMADE ↓	MMFDE ↓	APD ↑	ADE ↓	FDE ↓	MMADE ↓	MMFDE ↓
ERD [13]	0	0.722	0.969	0.776	0.995	0	0.382	0.461	0.521	0.595
acLSTM [50]	0	0.789	1.126	0.849	1.139	0	0.429	0.541	0.530	0.608
Pose-Knows [42]	6.723	0.461	0.560	0.522	0.569	2.308	0.269	0.296	0.384	0.375
MT-VAE [46]	0.403	0.457	0.595	0.716	0.883	0.021	0.345	0.403	0.518	0.577
HP-GAN [4]	7.214	0.858	0.867	0.847	0.858	1.139	0.772	0.749	0.776	0.769
Best-of-Many [5]	6.265	0.448	0.533	0.514	0.544	2.846	0.271	0.279	0.373	0.351
GMVAE [12]	6.769	0.461	0.555	0.524	0.566	2.443	0.305	0.345	0.408	0.410
DeLiGAN [20]	6.509	0.483	0.534	0.520	0.545	2.177	0.306	0.322	0.385	0.371
DSF [49]	9.330	0.493	0.592	0.550	0.599	4.538	0.273	0.290	0.364	0.340
DLow [48]	11.741	0.425	0.518	0.495	0.531	4.855	0.251	0.268	0.362	0.339
Ours	8.006	0.476	0.582	0.540	0.590	3.822	0.282	0.285	0.368	0.334
	HumanEva-I → Human3.6M					Human3.6M → HumanEva-I				
DLow [48]	5.929	0.848	0.907	0.870	0.913	7.031	0.563	0.623	0.614	0.643
Ours	6.127	0.611	0.831	0.665	0.849	7.206	0.540	0.571	0.591	0.596

Table 2: Quantitative results on Human3.6M and HumanEva-I.

sample, which is computed as

$$ADE = \frac{1}{T}\min_{x\in\mathcal{X}}\|\hat{x} - x\|, \tag{15}$$

where $\mathcal{X}$ is the set of generated sequences.

Fianl Displacement Error (FDE): *L2* distance between the final ground truth pose x^T and the closest sample's final pose, which is computed as

$$FDE = \min_{x\in\mathcal{X}}\|\hat{x}^T - x^T\|. \tag{16}$$

Multi-Modal ADE (MMADE): the multi-modal version of ADE that obtains multi-modal ground truth future motions by grouping similar past motions.

Multi-Modal FDE (MMFDE): the multi-modal version of FDE.

Among these metrics, APD has been used to measure sample diversity [2]. ADE and FDE are common metrics for evaluating sample accuracy in trajectory forecasting literature [1, 19, 32]. MMADE and MMFDE [49] are metrics used to measure a method's ability to produce multi-modal predictions.

4.3 Experimental Results and Analysis

We compare the performances of our proposed method and other baseline models on Human3.6M and HumanEva-I datasets on the two tasks of human motion completion and prediction. The experimental results and analysis are as follows.

Human motion completion. To the best of our knowledge, there is currently no published model that can directly solve the problem of human motion completion. To verify the effectiveness of our model, we construct two baseline model **Linear Interpolation** and **DCT-GCN** for human motion completion. For the linear interpolation method, we interpolate the starting and end poses to complete the motion sequence of the specified length in between. As mentioned earlier, the DCT-GCN method was originally used for human motion prediction. In Section 3.2, we modify the DCT-GCN model and use it for short-term transition sequence generation. Here we extend the model used in section 3.2 to make it possible to generate motion completion results of varying lengths.

The experimental results of above methods are shown in Table 1. Larger IS scores or smaller FID are better. For the human motion completion task, we mainly focus on the quality and diversity within the generated sequences, namely IS^t and FID^t. Since the input snippet length of our human action classification model is fixed to 1 second, when the sequence length is 1 second, the values of FID^t and IS^t can not be calculated.

As shown in Table 1, the simplest linear interpolation method has the worst performance, and the DCT-GCN method is slightly better. As the length of the generated sequence increases, more stronger advantage of our method over the two baselines can be observed. This is because our data-driven method directly finds the appropriate motion from the database for sequence generation. As the sequence length increases, the proportion of the transition sequence in the entire generated sequence is smaller, therefore the generation result is more real. Our proposed approach is more effective in long sequence completion.

Human motion prediction. We summarize the quantitative results on Human3.6M and HumanEva-I in Table 2. The metrics are computed with a sample set size of 50. On both datasets, our method outstrips all other previous methods, except for two recent variational modeling based methods DSF [49] and DLow [48]. We attribute the slight inferiority to the lack of fine-grained pose-processing (only standard Gaussian smoothing is used in our current implementation) after obtaining the synthesized motion sequences, unlike [48, 49].

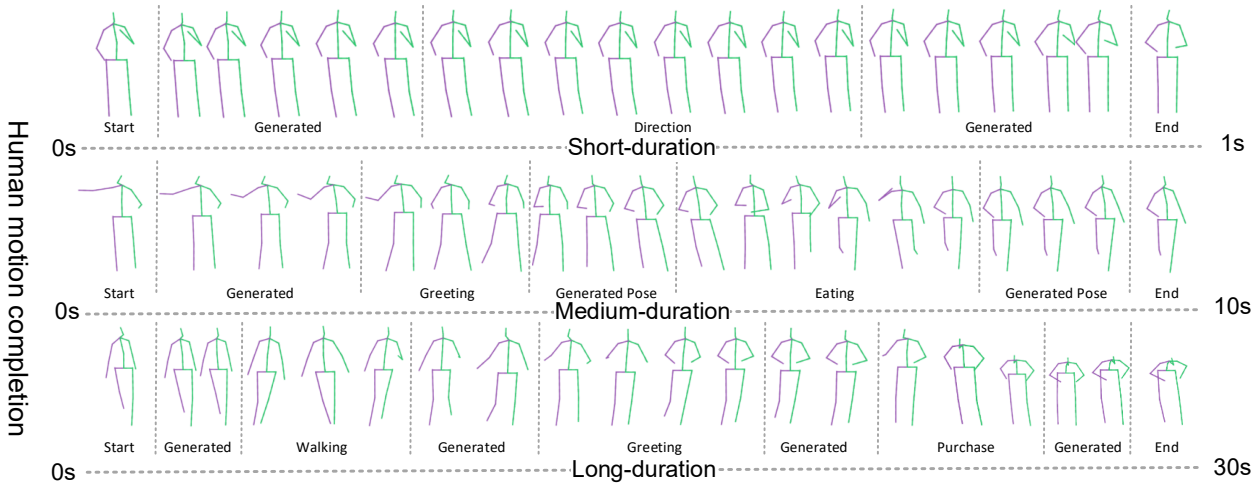

Figure 4: Some examples for human motion completion.

Bits Length	12	24	48	72	96
mAP	5.14	16.10	29.40	37.48	41.59
Precision@5	20.84	39.84	54.80	63.20	65.20
Precision@10	20.62	38.09	52.45	59.52	62.35
QPS	387.1	306.3	179.2	134.8	104.5

Table 3: Experimental results for global alignment.

Cluster Numbers	20	40	60	80	100
mAP	27.33	28.23	27.67	28.99	26.63
Precision@5	53.68	53.98	54.10	56.28	55.50
Precision@10	50.65	51.16	50.69	52.61	52.32
QPS	1126.5	867.6	596.5	540.6	465.4

Table 4: Experimental results for group alignment.

In order to evaluate the generalization performance of different methods, we also conduct cross-scenario experiments. As shown in Table 2, HumanEva-I → Human3.6M refers to building a motion graph or training model on the training set of the HumanEva-I dataset, and testing on the test set of the Human3.6M dataset. The experimental results show that the performances of our method on all metrics are consistently more robust than the state-of-the-art model DLow [48]. The difference of generalization performances suggests that current generative models, exemplified by DLow [48], suffers from more severe over-fitting issue than data-driven methods like ours in this paper.

Effectiveness of post-processing. We conducted several ablation studies to verify the effectiveness of our post-processing module. In Table 1, **Ours** represents the method before using the generated network for post-processing, and **Ours*** represents our complete method after post-processing. As shown in the above tables, our post-processing module can play a positive role in most cases.

Pose search. In this part, we conduct experiments to study the impact of pose hashing on pose search performance. In the experiment, we randomly select 1000 query poses from the testing dataset and calculate the pose distance between them and all poses in the training set. If the distance is less than the threshold, it is set as positive, otherwise it is negative, which is the groundtruth used in the evaluation later. We use mAP, Precision@5, Precision@10 and QPS (Queries Per Second) to evaluate the performance of the methods. Precision@5 refers to the proportion of positive samples among the 5 samples given by the model. If we do not do pose hashing, that is, we search according to the method of calculating groundtruth, the query speed is 5.8 QPS. Table 3 shows the retrieval effect after we align all the data in the training set with the same pose, and then perform hashing. It can be seen that the retrieval speed is significantly improved compared to 5.8 QPS. Based on the consideration of speed and accuracy, we choose 48 bits for group alignment experiment. Table 4 shows the effect of the number of clusters of spectral clustering on the retrieval performance. Normally, the more the number of clusters, the faster the retrieval speed, but because our database is relatively small, the time to find a suitable cluster takes up a higher proportion, so the search speed slows down as the cluster increases. According to the above experimental results, we choose 80 clusters and 48 bits length as the final pose search module parameters.

Some examples. In Figure 4, we illustrate the results of human motion completion at three different scales (1s, 10s,, and 30s). In order to better show the composition of the generated sequence, the pose here is not sampled at equal intervals. More video examples can be seen in the supplementary materials.

5 CONCLUSIONS

In this paper, we introduce a framework for generating human diversity motion sequences by data-driven and generative models. The whole pipeline can be decomposed into three stages: Motion Graph Construction, Transition Motion Generation and Human Motion Sequences Generation. In the first stage, we construct a motion graph on the training set. This graph can represent the change of motion within the same sequence and the transfer between different sequences. In the second stage, a modified DCT-GCN model is used to generate transition sequences for the transition edges. In the last stage, we apply graph path search algorithm and graph walks to handle human motion completion and prediction tasks, respectively. The experimental results on dataset Human3.6M and HumanEva-I demonstrate the effectiveness of our proposed method.

Acknowledgement: This work is supported by National Natural Science Foundation of China (61772037) and Beijing Natural Science Foundation (Z190001).

REFERENCES

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Fei-Fei Li, and Silvio Savarese. 2016. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *CVPR*. IEEE Computer Society.

[2] Mohammad Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Lars Petersson, and Stephen Gould. 2020. A Stochastic Conditioning Scheme for Diverse Human Motion Prediction. In *CVPR*.

[3] Okan Arikan and David A. Forsyth. 2002. Interactive motion generation from examples. *ACM Trans. Graph.* 21, 3 (2002), 483–490.

[4] Emad Barsoum, John Kender, and Zicheng Liu. 2018. HP-GAN: Probabilistic 3D Human Motion Prediction via GAN. In *CVPR Workshops*.

[5] Apratim Bhattacharyya, Bernt Schiele, and Mario Fritz. 2018. Accurate and Diverse Sampling of Sequences Based on a "Best of Many" Sample Objective. In *CVPR*.

[6] Ali Borji. 2019. Pros and cons of GAN evaluation measures. *Comput. Vis. Image Underst.* 179 (2019), 41–65.

[7] Matthew Brand and Aaron Hertzmann. 2000. Style machines. In *SIGGRAPH*.

[8] Judith Bütepage, Michael J. Black, Danica Kragic, and Hedvig Kjellström. 2017. Deep Representation Learning for Human Motion Prediction and Classification. In *CVPR*.

[9] Haoye Cai, Chunyan Bai, Yu-Wing Tai, and Chi-Keung Tang. 2018. Deep Video Generation, Prediction and Completion of Human Action Sequences. In *ECCV*.

[10] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A. Efros. 2019. Everybody Dance Now. In *ICCV*.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*.

[12] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. 2016. Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. *CoRR* abs/1611.02648 (2016).

[13] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent Network Models for Human Dynamics. In *ICCV*.

[14] Haifeng Gong, Jack Sim, Maxim Likhachev, and Jianbo Shi. 2011. Multi-hypothesis motion planning for visual object tracking. In *ICCV*.

[15] Yunchao Gong and Svetlana Lazebnik. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*. 817–824.

[16] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 12 (2013), 2916–2929.

[17] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NeurIPS*.

[18] Liang-Yan Gui, Yu-Xiong Wang, Xiaodan Liang, and José M. F. Moura. 2018. Adversarial Geometry-Aware Human Motion Prediction. In *ECCV*.

[19] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social GAN: Socially Acceptable Trajectories With Generative Adversarial Networks. In *CVPR*.

[20] Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and R. Venkatesh Babu. 2017. DeLiGAN: Generative Adversarial Networks for Diverse and Limited Data. In *CVPR*.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

[22] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *NeurIPS*.

[23] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely Connected Convolutional Networks. In *CVPR*.

[24] Ashesh Jain, Amir Roshan Zamir, Silvio Savarese, and Ashutosh Saxena. 2016. Structural-RNN: Deep Learning on Spatio-Temporal Graphs. In *CVPR*.

[25] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised classification with Graph Convolutional Networks. In *ICLR*.

[26] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *NeurIPS*.

[27] Hema Swetha Koppula and Ashutosh Saxena. 2013. Anticipating human activities for reactive robotic response. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*. 2071.

[28] Lucas Kovar, Michael Gleicher, and Frédéric H. Pighin. 2002. Motion graphs. *ACM Trans. Graph.* 21, 3 (2002), 473–482. https://doi.org/10.1145/566654.566605

[29] Andreas Krause, H. Brendan McMahan, Carlos Guestrin, and Anupam Gupta. 2008. Robust Submodular Observation Selection. *Journal of Machine Learning Research* 8, 2 (2008), 2761–2801.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS*.

[31] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. 2002. Interactive control of avatars animated with human motion data. *ACM Trans. Graph.* 21, 3 (2002), 491–500.

[32] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B. Choy, Philip H. S. Torr, and Manmohan Chandraker. 2017. DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents. In *CVPR*.

[33] Chen Li, Zhen Zhang, Wee Sun Lee, and Gim Hee Lee. 2018. Convolutional Sequence to Sequence Model for Human Dynamics. In *CVPR*.

[34] Diogo C. Luvizon, David Picard, and Hedi Tabia. 2018. 2D/3D Pose Estimation and Action Recognition Using Multitask Deep Learning. In *CVPR*.

[35] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. 2019. Learning Trajectory Dependencies for Human Motion Prediction. In *ICCV*.

[36] Julieta Martinez, Michael J. Black, and Javier Romero. 2017. On Human Motion Prediction Using Recurrent Neural Networks. In *CVPR*.

[37] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. [n.d.]. 3D Human Pose Estimation in Video With Temporal Convolutions and Semi-Supervised Training. In *CVPR, year = 2019*.

[38] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. 2016. NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis. In *CVPR*.

[39] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. 2010. HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion. *Int. J. Comput. Vis.* 87, 1-2 (2010), 4–27.

[40] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

[41] Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating Text with Recurrent Neural Networks. In *ICML*.

[42] Jacob Walker, Kenneth Marino, Abhinav Gupta, and Martial Hebert. 2017. The Pose Knows: Video Forecasting by Generating Pose Futures. In *ICCV*.

[43] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. 2008. Gaussian Process Dynamical Models for Human Motion. *IEEE Trans. Pattern Anal. Mach. Intell.* 30, 2 (2008), 283–298.

[44] Sijie Yan, Zhizhong Li, Yuanjun Xiong, Huahan Yan, and Dahua Lin. 2019. Convolutional Sequence Generation for Skeleton-Based Action Synthesis. In *ICCV*.

[45] Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *AAAI*.

[46] Xinchen Yan, Akash Rastogi, Ruben Villegas, Kalyan Sunkavalli, Eli Shechtman, Sunil Hadap, Ersin Yumer, and Honglak Lee. 2018. MT-VAE: Learning Motion Transformations to Generate Multimodal Human Dynamics. In *ECCV*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (Eds.).

[47] Ceyuan Yang, Zhe Wang, Xinge Zhu, Chen Huang, Jianping Shi, and Dahua Lin. 2018. Pose Guided Human Video Generation. In *ECCV*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (Eds.). Springer.

[48] Ye Yuan and Kris Kitani. 2020. DLow: Diversifying Latent Flows for Diverse Human Motion Prediction. In *ECCV*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.).

[49] Ye Yuan and Kris M. Kitani. 2020. Diverse Trajectory Forecasting with Determinantal Point Processes. In *ICLR*.

[50] Yi Zhou, Zimo Li, Shuangjiu Xiao, Chong He, Zeng Huang, and Hao Li. 2018. Auto-Conditioned Recurrent Networks for Extended Complex Human Motion Synthesis. In *ICLR*.