Université de Lille

THÈSE

pour obtenir le grade de

DOCTEUR,
SPÉCIALITÉ INFORMATIQUE ET APPLICATIONS

présentée et soutenue publiquement par

BAPTISTE CHOPIN

le 10 Mars 2023

Nouvelle approches pour la prédiction et la génération de mouvement humain utilisant des squelettes 3D : application aux interactions non-verbales en réalité virtuelle

préparée au sein des laboratoires CRIStAL UMR CNRS 9189 et SCALab UMR CNRS 9193

sous la direction de

Prof. Mohamed Daoudi et **Prof. Angela Bartolo**

MOHAMED DAOUDI	Directeur
Professeur	IMT NORD EUROPE
ANGELA BARTOLO	Co-Directrice
Professeur	UNIVERSITÉ DE LILLE
ANTITZA DANTCHEVA	Rapporteure
CR INRIA	INRIA SOPHIA ANTIPOLIS - MÉDITERRANÉE
RENAUD SÉGUIER	Rapporteur - Président du jury
Professeur	CENTRALESUPÉLEC
PIETRO PALA	Examinateur
Professeur	UNIVERSITY OF FLORENCE
CATHERINE PELACHAUD	Examinatrice
DR CNRS	SORBONNE UNIVERSITÉ
NICU SEBE	Examinateur
Professeur	UNIVERSITY OF TRENTO
HATICE GUNES	Examinatrice
Professeur	UNIVERSITY OF CAMBRIDGE

ACKNOWLEDGEMENTS

Firstly, I would like to express my gratitude to Prof. Mohamed Daoudi and Prof. Angela Bartolo for supervising this thesis and helping me in making it successful, especially when the pandemic forced us to change focus.

I thank the reviewers and examiners, Dr. Antitza Dantcheva, Prof. Renaud Séguier, Prof. Pietro Pala, Prof. Catherine Pelachaud, Prof. Nicu Sebe, and Prof. Hatice Gunes for taking the time to evaluate this work.

I thank Dr. Naima Otberdout and Dr. Hao Tang for the collaborations we had during this thesis.

I thank the Image group and the 3D-SAM team of the CRIStAL laboratory and the members of SCALab for the great time I had in those teams and for regularly helping me.

I would like to thank Hugo Pina Borges, Pierre Cilluffo, and Alain Tran who helped me build the virtual agent application and improve my ideas during their internships at CRIStAL.

I thank people at the Equipex Continuum and especially Laurence Delbarre for letting us use their facilities and helping us with the dataset collection.

Finally, I wish to thank the CNRS for financing this thesis through the 80-prime project.

ABSTRACT

In this thesis, we address various tasks for generating 3D skeletons of humans in motion. The ability to predict and generate human motion has become an important topic in recent years in many domains including self-driving vehicles, animation, and virtual reality. While in recent years deep learning has greatly increased the performance of generative models, the generation of human motion remains an open issue. Even the more recent methods still struggle to generate high-quality human motion. This is due to the need to model both spatial and temporal components and of understanding the interactions of human body parts. The task is also challenging due to the high variability of motions both in terms of time since the same motion can be performed at a different speed, and in terms of space, since the amplitude of motion can vary greatly. Furthermore, the generated 3D motions must be accurate, realistic, and smooth. We propose a new predictive Wasserstein generative adversarial network (GAN) to predict the end of a person's motion. Our predictive network uses the SRVF representation to model human motion and allow the prediction of accurate motion without discontinuities in real-time as shown in our experiments against state-of-the-art methods. We then work on the generation of interaction motions between two persons. We present a new method to generate a reaction motion in response to an action. Unlike the state-of-the-art methods that focus on generating the motion of a single person, we propose *Interformer*, a Transformer to predict the reaction to an action using the temporal modeling abilities of the Transformer network as well as new skeleton adjacency and interaction distance modules to model the interactions. We compare our results to interaction generation and motion prediction methods and outperform them. We develop a new architecture to generate the motion of two people interacting based on a class label. Our architecture leverages the capabilities of diffusion models, Transformer architecture, and bipartite graph networks. Our results show that our method outperforms the state-of-the-art both quantitatively and qualitatively. We propose an application that uses our motion prediction method to allow a virtual agent to predict and recognize a person's motion in non-verbal interactions in a virtual environment. For this purpose, we propose a new 3D motion database captured with a high-quality motion capture system and a depth camera.

RÉSUMÉ

Dans cette thèse, nous abordons diverses tâches de génération de squelettes 3D de corps humain en mouvement. La capacité à prédire et générer des mouvements humains est devenue un sujet important dans de nombreux secteurs tel que la conduite de véhicules autonomes, l'animation et la réalité virtuelle. Bien que l'apprentissage profond ait considérablement amélioré les performances des modèles génératifs ces dernières années, la génération de mouvements humains reste un problème ouvert. Les méthodes les plus récentes ont toujours du mal à générer des mouvements humains de bonne qualité. Cela résulte de la nécessité de modéliser les composantes spatiales et temporelles simultanément et de comprendre les interactions entre les différentes parties du corps. La tâche est également difficile en raison de la grande variabilité des mouvements, à la fois en termes de temps, puisque le même mouvement peut être effectué à une vitesse différente, et en termes d'espace, puisque l'amplitude du mouvement peut varier considérablement. De plus les mouvements 3D générés doivent être précis, réalistes et fluides. Nous proposons un nouveau réseau antagoniste génératif (GAN) prédictif de Wasserstein pour prédire la fin du mouvement d'une personne. Notre réseau prédictif utilise une répresentation des courbes appelée SRVF pour modéliser la trajectoires des mouvements humains et permet une prédiction précise, en temps réel, de mouvement sans discontinuités comme le montrent nos expériences. Dans une seconde étape de la thèse nous nous intéressons à la génération des mouvements d'interaction entre deux personnes. Tout d'abord, nous présentons une nouvelle méthode pour générer un mouvement de réaction en réponse à un mouvement d'action. Contrairement aux méthodes de l'état de l'art qui se focalisent sur la génération du mouvement d'une personne, nous proposons Interformer, un Transformer qui génère des mouvements de réaction en utilisant les capacités de modélisation temporelles des réseaux Transformer ainsi que de nouveaux modules pour modéliser les interactions. Nos résultats montrent que l'approche Interformer surpasse les méthodes de l'état de l'art. Ensuite nous développons une nouvelle architecture pour générer le mouvement d'interaction de deux personnes en fonction de la classe du mouvement. Notre architecture exploite les capacités des modèles de diffusion, de l'architecture Transformer et l'apprentissage de graphes bipartis. Nos résultats montrent que notre méthode surpasse l'état de l'art quantitativement et qualitativement. Nous proposons une application qui utilise la méthode de prédiction du mouvement afin de permettre à un agent virtuel de prédire et de reconnaître le mouvement d'une personne dans le cadre des interactions non-verbales dans un environnement virtuel. Pour cela nous avons proposé une nouvelle base de données de mouvement 3D capturée avec un système de capture de mouvement de haute qualité et une caméra de profondeur.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

Introduction

CHAPTER 1

INTRODUCTION

1.1 Goals

The goal of this thesis is to develop novel generative models for the generation of human motion. Human motion generation is an important scientific topic where we need to model both spatial and temporal components of a motion to generate data of the highest quality. There are many human motion generation tasks but we will focus on only three:

- **3D skeleton-based human motion prediction** The problem of forecasting future human motion play a vital role in many applications in computer vision and robotics, such as human-robot interaction, autonomous driving, and computer graphics. The objective of this task consists in forecasting future human poses based on a prior skeleton pose sequence. In this thesis, we propose a predictive model for short and long-term future 3D skeleton poses given an initial prior history.

- **Human interaction motion generation** aims at generating 3D human interaction motions. What makes interaction generation challenging is the non-linearity of human motion interaction and the diversity of the interaction between humans. Several questions rise to tackle these challenges. We will focus on how to represent the interaction between humans and how to model motion and generate diverse motion interactions.

- **Application to Avatar Interaction** Virtual agents or avatars are artificial intelligences that have a human appearance. Allowing users to interact with such agents increases their realism. By leveraging our work on 3D human motion prediction we build an interactive virtual agent. Being able to predict the end of a motion allows the virtual agent to react faster and more naturally.

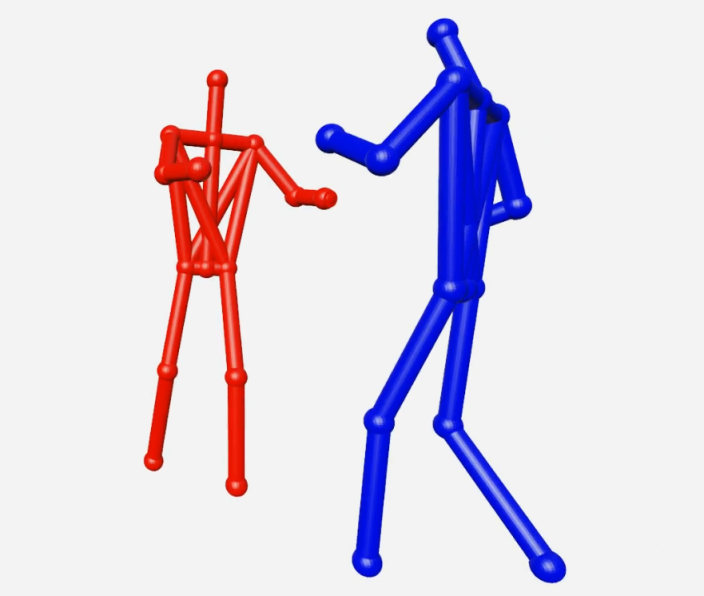
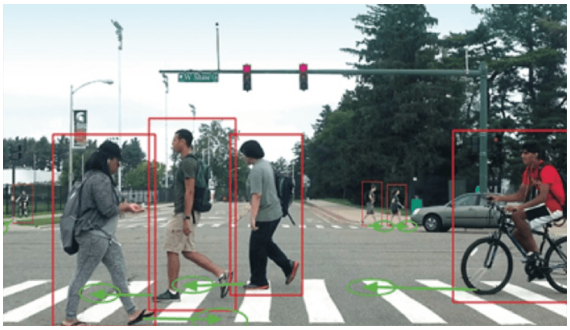


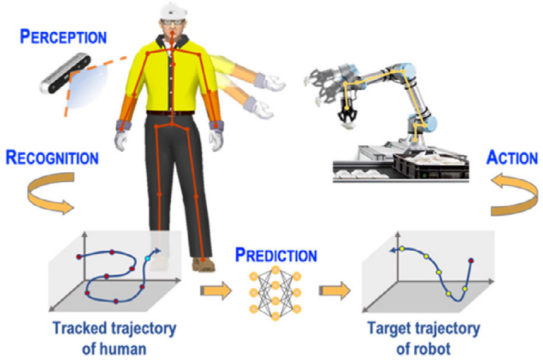
Figure 1.1: Examples of 3D skeletons. Example from the DuetDance dataset [83] rendered with Blender.

1.2 Motivations

Being able to accurately generate 3D motions has become a hot topic in computer vision. From motion prediction based on historical sequences and generation of motions for a particular action class to conditional motion generation that follows a trajectory or a song, many methods are being proposed to generate 3D human motion. Researchers are trying to generate ever more realistic and ever more complex motions using state-of-the-art methods. The interest in generative methods for human motions stems from the possible uses of these methods in multiple areas.



(a) Some algorithms for self-driving cars try to predict the direction in which pedestrians move.



(b) Framework of human motion prediction to help a robot interact with the user

(a) source: https://www.aau.edu/research-scholarship/featured-research-topics/vision-smarter-safer-self-driving-vehicles
(b) source: [164]

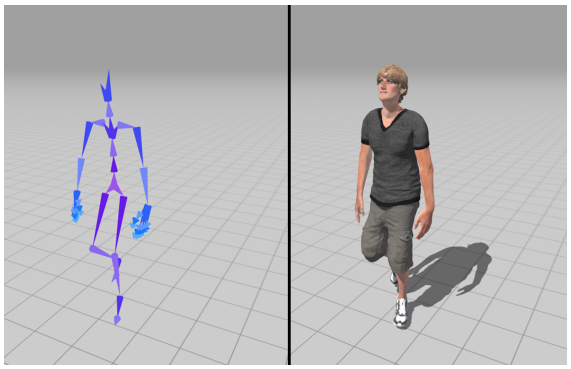
Figure 1.2: Use cases for motion prediction.

Robotic. In robotics being able to predict the motion of the human user allows robots to interact more quickly and more accurately with the user. For example, if the robot needs to grab something that a human is handing to it, usually the robot needs to wait for the human to finish his motion before moving to grab the object. If we can predict the end of the motion of the human, the robot can start moving earlier and grab the object more quickly. With an accurate enough prediction, it is possible to grab the object right when the human stops moving, much like what would happen with a real person. This would lead to more realistic interactions and a better appreciation of the robot by users.

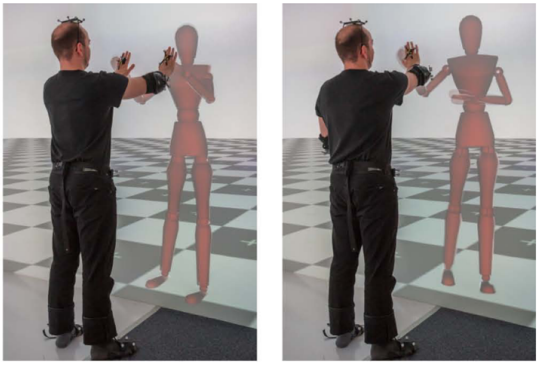
Self-driving vehicle. By enabling self-driving vehicles to predict the motion of humans or how an interaction between several persons will unfold we can make them safer for passengers and pedestrians alike. For example by predicting the future motion of a person based on gait we could see if there is a high chance for a pedestrian to cross the road. If the probability is high then the car can decelerate or take an avoiding trajectory to avoid the accident. When looking at people interacting if we can predict the behavior of a participant based on the motion of others we can avoid even more accidents. If a person is pushing another and we are able to predict if the pushing motion is enough for the person to fall on the road. If it is the case the car will stop automatically to avoid causing an accident. It would make the autonomous car safer, and the driver more comfortable by avoiding slowing down

or requesting the driver to take over when there are people on the side of the road that do not represent a risk.

Animation. In the domain of animation, the use of motion prediction and reaction generation will speed up the animation process for the artist. With human motion prediction, if the artist animates a short sequence an algorithm could propose several future sequences to complete it. To model an interaction, the artist would only need to animate an action sequence and an algorithm would propose several possible reactions by the other character. Or the artist could generate the entire interaction simply by inputting the kind of motion needed. Using these methods would help animators reduce their workload as well as enable amateurs to make their own animation with less effort on their part. Fig 1.3a shows an automatic rigging (linking a 3D skeleton to a character mesh) method from Adobe Mixamo [1]. Coupled with motion generation methods, automatic rigging allows for a faster and simpler animation process.



(a) Automated rigging method for animation allows the application of a skeleton to an existing 3D character.
(a) source: https://www.mixamo.com/



(b) A virtual agent is animated in real-time to interact with a human user.
(b) source: [147]

Figure 1.3: Use cases for motion generation.

For computer vision research. The available 3D human motion data is still relatively limited and some of the newer architectures require a lot of data. Motion generation methods can then be used to create synthetic datasets for several computer vision tasks such as action recognition and human motion prediction.

Virtual agents Virtual agents are characters controlled by artificial intelligence in virtual settings. They will interact with the user and, as virtual reality gain traction, these interactions need to be more realistic than before. Motion generation methods can help with this by providing more motions for a given action, predicting the user's motion to react faster, or generating the proper reaction to an action from the user. Fig 1.3b shows a method that animates a virtual agent in real time while interacting with a human user. Like the method presented in [147], existing methods use prerecorded motions that are then modified on the fly to better fit the spatial and temporal context. Motion generation methods can improve the performance of such systems by providing more diverse high-quality motions.

[1] https://www.mixamo.com/

1.3 Challenges

Methods that could be used in these applications need to solve several challenges.

Human Motion Prediction seeks to predict as accurately as possible the motion of a person based on a past sequence of motion and must face two main challenges: long-term accuracy and smooth motion. In addition to these two challenges, we also want the model to be able to predict in real time so it can be used in a virtual agent application. Long-term accuracy is difficult to achieve mostly because of the accumulation of errors in the predicted poses. Error accumulation happens when we feed to the network its own output that contains errors, the results will be based on data that contains errors and consequently contain even more errors. If the process is repeated recursively it can lead to strong degradation in the results. This is, for example, the case for RNN-based methods that treat one frame at a time. Since each frame is generated based on previously generated frames the error accumulates very quickly and leads to a bad long-term prediction. Another reason for the low accuracy of long-term predictions originates from the way the problem is treated: we only use motion data as information to guide the prediction. This limits the maximum prediction length. In the human motion prediction literature, long-term prediction corresponds to the prediction of $1s$ of motion [107]. To get accurate predictions of longer sequences we would need to consider other modalities such as the environment in which the person evolves (walls, furniture, roads, other people...) and the intent behind his actions.

Human reaction generation takes an action motion and predicts possible reactions by another person. This is a new domain that we explore in this thesis. The generated motions must be smooth, and realistic as for human motion prediction but must also fit well the condition *e.g* if the action is 'punching', the generation reaction must be 'being punched'. This means that we want the reaction to fit the conditioning action *e.g* with a punching action the reaction must show the impact of the location that has been punched and follow the direction from which the punch comes. Another example is when the action is a person shaking a hand we want the hand of the reaction to be in contact with the hand of the action body. Unlike motion prediction we do not seek perfect accuracy, the reaction must correspond to the action but we also want diversity in the generated reaction *e.g* reacting more or less strongly to being punched.

Human interaction generation is the generation of two-person interaction motions given an input class label, *e.g* by inputting "shake hands" we want to generate motions of two persons shaking hands. What we seek with interaction generation is similar to the reaction generation task: smooth and realistic motion for both persons and a realistic interaction between the two persons. In addition, we would like high diversity in the generated motion.

1.4 Contributions

In this thesis, we present a new method for human motion prediction, an architecture for human reaction generation, a diffusion model for human interaction generation, a dataset of gestures, and an application consisting of a virtual agent that reacts to human gestures in real

time and in a realistic manner. Our contributions are the following.

Human Motion Prediction To predict the future motion of a single person we propose a manifold aware Wasserstein generative adversarial network. We predict future human poses based on a historical sequence of poses by representing the motion as a trajectory. This allows us to simplify the motion prediction problem to generate a point on a hypersphere. We outperform the literature for long-term prediction and are competitive for short-term prediction.

Human Reaction Generation To generate the reaction motion based on an action sequence in a two-person interaction we propose a Transformer network using spatial attention and graphs. The architecture is based on the original Transformer [146] and we add modules to deal with the structure of the skeletons and the interaction between the two subjects. Our method allows the generation of various reaction motions even complex ones like duet dancing. We are also able to generate long motion (up to 40 seconds). As there is no other method dealing with reaction generation, we compare our approach with methods for motion prediction and interaction generation. We show that our method offers much better quantitative and qualitative results than some state-of-the-art methods.

Human Interaction Generation To generate two-person interaction motions we propose BiGraphDiff, a novel bipartite graph diffusion method. Specifically, bipartite node sets are constructed to model the inherent geometric constraints between skeleton nodes during interactions. The interaction graph diffusion model is transformer-based, combining some state-of-the-art motion methods. We compare BiGraphDiff to state-of-the-art action-based motion generation and achieve more realistic results than the existing methods while providing higher diversity.

Gestures Dataset A new dataset of 3D gestures of 11 subjects to train our models that are used with the virtual agent, is proposed. Each subject was asked to perform at least 10 times the motions of 6 classes. A seventh class contains random motions to use as an "Idle" class performed by 2 subjects. Each motion is recorded by motion capture and a Kinect V2 camera resulting in 1456 data samples after the manual removal of bad samples.

Virtual Agent Application A virtual agent that reacts to gestures and a virtual environment in which it can interact with the user is proposed. We capture motion with a Kinect V2 camera, predict the end of the motion and then classify it. The application also features a facial expression recognition module to see if the user is distressed. The application can also be used with a virtual reality headset.

1.5 Thesis Outline

Chapter 2: In this chapter, we present our work on human motion prediction. We introduce the method to obtain Square Root Velocity Function (SRVF) [137] representation from motion data. We then show the architecture we use to generate SRVF representing the future motion and the losses used to train it. We compare our method to others from the

literature on Human3.6M [64] and CMU MoCap [2] and show that we outperform them on long term prediction while remaining competitive for short term prediction. We also present an ablation study and extrapolation capabilities with recursive generation and cross-dataset prediction.

Chapter 3: Here we introduce a method that generates a reactive motion that responds to an action motion, *e.g* from a punching motion we generate the motion of the person being punched. We present the Transformer architecture used for natural language processing and computer vision. We show how we represent the problem of reaction motion generation conditioned on another motion and how we adapt Transformer to solve it. We present two new modules to deal with the nature of skeleton data and to help model the interaction. As there exists no other method that deals with reaction generation we compare our work with methods for motion prediction and interaction generation [11, 5]. We evaluate our approach on the SBU [163] and K3HI [61] datasets for simple human interactions and on duet-dancing motions from the DuetDance dataset [83] for more complex motion. We show that we outperform the other approach both quantitatively and qualitatively and present an ablation study to show how our method improves the generation over a vanilla Transformer.

Chapter 4: We present a method to generate two-person interaction from a label (*e.g* inputting "Punching" to generate a character punching a character being punched) using a diffusion process with a Transformer architecture and Bipartite graphs. The diffusion process has proved to be very powerful for generative tasks, Transformer helps us model the temporal and spatial (Intra-personal) correlation while the Bipartite Graph model the interactive part of the motion. We compare our method to state-of-the-art motion generation methods including one other diffusion-based method [165]. The comparison is done on 26 interaction classes from the large NTU RGB+D 120 [94] dataset and on the DuetDance dataset [83].

Chapter 5: In this chapter, we present our new gesture dataset captured with both a motion capture system and a Kinect V2 camera. We then introduce our virtual agent interaction application built by using the methods described in the two previous chapters and the necessary modification to properly run the application in real time. We present the results of the separate module on the new dataset and several use cases for the application.

Chapter 6: We will end this thesis with a summary of the contribution followed by a list of the limitations of our methods and finally a description of the possible improvements and extensions.

[2] http://mocap.cs.cmu.edu

1.6 Publications

Baptiste Chopin, Naima Otberdout, Mohamed Daoudi, Angela Bartolo. "Human Motion Prediction Using Manifold-Aware Wasserstein GAN". In: 2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021). IEEE, 2021. p. 1 [28]

Baptiste Chopin, Naima Otberdout, Mohamed Daoudi, Angela Bartolo. "3D Skeleton-based Human Motion Prediction with Manifold-Aware GAN," in IEEE Transactions on Biometrics, Behavior, and Identity Science, 2022, doi: 10.1109/TBIOM.2022.3215067. [27] Selected Works From Automated Face and Gesture Recognition 2021

Baptiste Chopin, Hao Tang, Naima Otberdout, Mohamed Daoudi, Nicu Sebe. "Interaction Transformer for Human Reaction Generation", in IEEE Transaction on Multimedia, doi: 10.1109/TMM.2023.3242152. [29]

Baptiste Chopin, Hao Tang, Mohamed Daoudi. "Bipartite Graph Diffusion Model for Human Interaction Generation", International joint Conference on Artificial Intelligence (IJCAI) 2023, Submitted.

Part II

Human motion Prediction using Wasserstein Generative Adversarial Network

CHAPTER 2

HUMAN MOTION PREDICTION USING WASSERSTEIN GENERATIVE ADVERSARIAL NETWORK

2.1 Introduction

The problem of forecasting future human motion plays a vital role in many applications in computer vision and robotics, such as human-robot interaction [79], autonomous driving [114] and computer graphics [82]. In this work, we propose a predictive model for short and long-term future 3D skeleton poses given an initial prior history. Addressing this issue involves two main challenges: How to represent the temporal evolution of human motion to ensure the smoothness of the predicted sequences? and how to take the spatial correlations between human joints into account to avoid implausible poses?

Because of the explosion of deep learning and the availability of large-scale datasets for human motion analysis, deep learning models have been widely exploited to address the problem of human motion prediction and especially Recurrent Neural Networks (RNN) [47, 65, 49, 109]. Indeed, RNN-based approaches achieved good advance in term of accuracy, however, the motions predicted with these methods present significant discontinuities due to the frame-by-frame regression process that discourage the global smoothness of the motion. In addition, recurrent models suffer from error accumulation across time, which increase error and worsen long-term forecasting performance. To remedy this, more recent works avoid these models and explore feed-forward networks instead. Including CNN [87], GNN [150] and fully-connected networks [20]. Thanks to their hierarchical structure, feed-forward networks can better deal with the spatial correlations of human joints than RNNs. However, an additional strategy is required to encode the temporal information when using these models. To face this issue, an interesting idea was to model the human motion as trajectory [108], [16].

In this work, we follow the idea of modeling motions as trajectories in time but in a different context from the previous work. Among the benefits of our representation, is the possibility to map these trajectories to single compact points on a manifold, which helps preserve the continuity and the smoothness of the predicted motions. Besides, the compact representation avoids the problem of error accumulation across time and makes our approach suitable for long-term prediction as illustrated in Figure 2.7. Nevertheless, the challenge here is that the resulting representations are manifold-valued data that cannot be manipulated with traditional generative models in a straightforward manner. To face this challenge, we introduce in this chapter, a manifold-aware Wasserstein Generative Adversarial Networks (WGAN) that predict future skeleton poses given the input prior motion sequence that is encoded as manifold-valued data. The spatial dependencies between human joints are taken into consideration in our method through additional loss functions that add more constraints on the predicted skeleton poses to ensure their plausibility. An overview of our prediction process is illustrated in Figure 2.4.

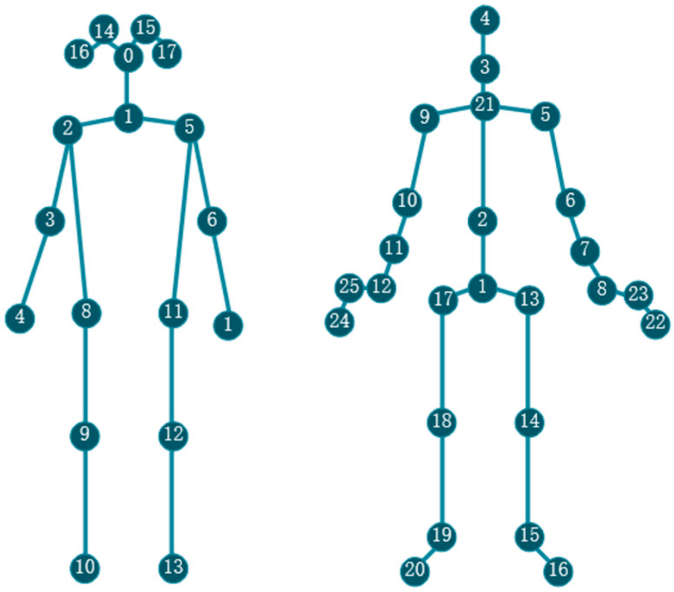
The contribution of this work can be summarized as follows: (1) To the best of our knowledge, we are the first to propose an approach that exploits compact manifold-valued representation for human motion prediction. By doing so, we model both temporal and spatial dependencies involved in human motion, resulting in smooth motions and plausible poses in long-term horizons. (2) We propose a predictive manifold-aware WGAN for motion prediction. (3) We propose a new loss function based on the Gram matrix of the 3D poses that avoids predicting implausible poses. (4) Experimental results on Human 3.6M and the CMU MoCap datasets show quantitatively and visually the effectiveness of our

method for short-term and long-term prediction. We present a new metric to evaluate the smoothness and the temporal evolution of the predicted motion. We provide a qualitative evaluation for our ablation study to highlight the importance of the different losses of our method. We also demonstrate in this chapter the ability of our method to predict longer sequences by recursive generation. The code and some videos of prediction are available at https://github.com/CRISTAL-3DSAM/PredictiveMA-WGAN

2.2 Related Work

Human motion Representation When modeling human action and motion several modalities can be considered: RGB videos [80], RGB-Depth videos [131], 2D skeletons [7], 3D skeletons [117], 3D shape [97]... We decided to focus on the 3D skeleton representation for human motion prediction but also for the other part of this thesis. As the name implies this representation consists in representing the human body as a skeleton with 3D coordinates. The skeleton is composed of joints positioned where real body joints are located (*e.g* knee, shoulder...) and bones that are the connections between the joints that mimic the real bones of the human body (*e.g* a bone from the elbow to the shoulder). The 3D coordinates are the coordinates of these joints. They can be captured using a motion capture system [64], a depth camera with software to detect the skeleton like the Kinect camera [94], or from RGB video using software like Mediapipe's BlazePose [145]. Due to the various sources of skeleton data, the number of joints can vary and the position of a particular joint can change slightly from one database to another (*e.g* in some case the "head joint" correspond to the middle of the forehead in other to the top of the head) as shown in Fig.2.1. Furthermore, the coordinates can be described using two different representations: the 3D cartesian coordinates or the angle format. The angle format can also be represented in various forms: Euler angle, quaternion, and axis-angle. A skeleton joint in 3D Cartesian coordinates is defined by three values x, y, and z which represent its position in space according to each axis. In Euler angle, the same joint is represented by θ_x, θ_y, and θ_z which is the rotation of the joint on each axis. Angle-axis joints are represented by a unit vector u characterized by its direction coordinates u_x, u_y, and u_z as well as the rotation angle on that vector θ. The quaternion representation also uses a unit vector u and an angle θ but the rotation of the joint is described by these four values: $cos(\frac{\theta}{2})$, $u_x sin(\frac{\theta}{2})$, $u_y sin(\frac{\theta}{2})$ and $u_z(sin\frac{\theta}{2})$. We can easily switch from one of the angular forms to another and also from the angle format to 3D coordinates. However, we can not easily convert 3D coordinates to angle format and conversion between these two formats can cause issues *i.e* two different angle poses can correspond to the same 3D coordinates pose [150]. For this reason and the fact that some datasets only provide 3D coordinates data, we decided to use only the 3D cartesian coordinates representation in this thesis.

Human motion prediction describes the generation of future poses based on a sequence of prior known poses. This can be helpful for many applications such as self-driving cars, accident prevention, or 3D animation. As such it is a very popular subject and many methods have been investigated too, solve it. Given that the task of human motion prediction is a temporally dependent problem, recurrent models (RNN) were the first potential solution to be investigated, hence several works applied RNN and their variants to tackle this task. In [47],



(a) (b)

Figure 2.1: Figure from [93]. **Left**: skeleton structure from Kinectics [75] extracted with Openpose. **Right** skeleton from NTU-RGB+D [94]. We can see that the two skeletons are very different with a different number of joints and different positions for joints representing the same body part.

the authors proposed a model that incorporates a nonlinear encoder and decoder before and after recurrent layers. Their approach suffers from error accumulation and discontinuity between the last frame of the prior and the first frames of the generated sequence. Moreover, their approach only captures the temporal dependencies but ignores the spatial correlations between articulations. To deal with this problem, [65] proposed a Structural-RNN model relying on high-level spatio-temporal graphs. [49] take a different direction to minimize the error accumulation effect in RNNs; they used a feed-forward network for pose filtering and a RNN for temporal filtering. However, this strategy only minimizes the accumulated error that still exists and deteriorates the performance of recurrent models in long-term prediction. In [109] the authors showed that RNN methods were often beaten by a simple baseline in terms of quantitative results. Trying to solve some of the issues of RNN, more recent works exploit convolutional neural networks (CNNs). To model the temporal evolution with these models, various strategies have been suggested. In [87, 20], convolution across time was exploited to model the temporal dependencies with convolution networks, while [108] adopt Discrete Cosine Transform to encode the motion as trajectory. [91] propose a convolutional autoencoder with 1D convolution layers and a hierarchical structure that exploit the structure of the human body. In [95] the authors use CNN to capture the spatial and temporal features simultaneously by using a trajectory space to model the motion. The methods based on CNN achieved better results than the RNN methods but a particular type of CNN was of interest for works using 3D skeletons: the Graph convolutional networks (GCN). Graph convolutional networks were also applied for motion prediction [150, 65] as a suitable tool to model the spatial correlations involved between the articulations. Indeed the human skeleton

in the 3D representation can be seen as a graph where the nodes are the skeleton joints and the edges are the bones. The simplicity of the conversion from skeleton to graph sparked great interest in the community and works succeeded in building coherent spatial and temporal graph [168, 133] leading to results that far surpassed those of RNNs and conventional CNNs. In this thesis, we take a completely different direction and we propose to deal with human motions by exploiting a manifold-valued representation with generative adversarial models.

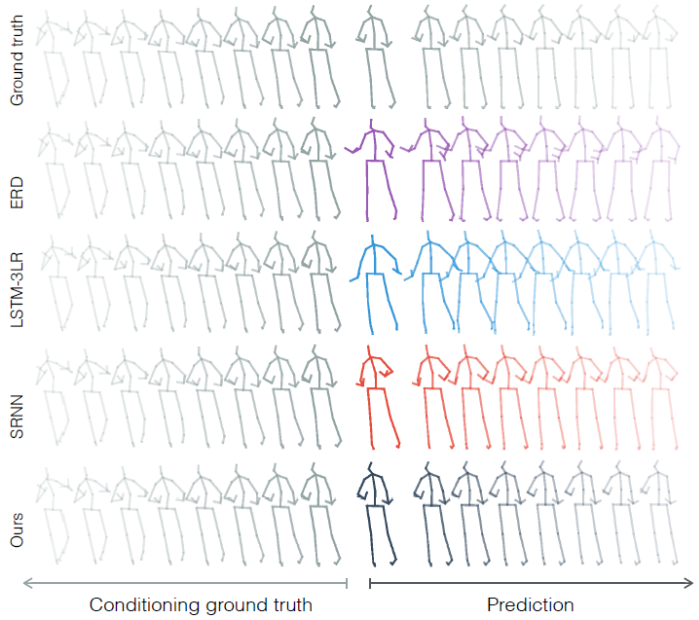


Figure 2.2: Figure from [109] showing the discontinuity problem between the last historical frame and the first predicted frame that is very visible on the first prediction methods.

Generative adversarial network Generative adversarial networks have been introduced in [51], its architecture consists of a generator that must generate data and a discriminator that must compare the generated data and decide whether the data is real or not Fig.2.3. Both are trained with opposite goals: the generator must generate data good enough to fool the discriminator while the discriminator must be able to always differentiate between real and generated data, hence the "adversarial" networks.

The adversarial process is ensured by the use of a specific loss that takes into account the objectives of both the generator and the discriminator called adversarial loss:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}(log(D(x))) + \mathbb{E}_{z\sim p_z(z)}(log(1D(G(z)))), \tag{2.1}$$

with x the data, $p_z(z)$ a prior on input noise variables , G the generator function and D the discriminator function. $D(x)$ represents the probability that x comes from the real data rather than the generated data. D is trained to maximize the probability of assigning the correct label (real or false) to each sample while G is trained to minimize $log(1 - D(G(z)))$.

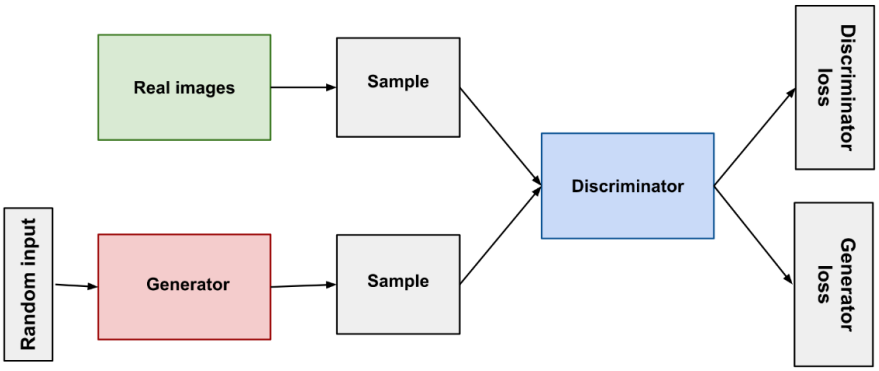


Figure 2.3: The GAN architecture [51]

The architecture of the generator and discriminator is very flexible and can contain anything from CNN [123] to VAE [162] and transformer [66]. A GAN can also contain several generators or several discriminators [41] and can be modified to perform conditional generation [6]. Due to this great flexibility, GANs have been used extensively in computer vision but also to generate music [42] or in NLP [92]. In computer vision the use of GAN led to an improvement in performance in a lot of domains like image translation [170], facial landmark detection [156], image inpainting [161], image super-resolution [84], 3D objects generation [152] and video generation [142].

Generative adversarial networks have also been investigated for human motion prediction after showing impressive performance on image generation tasks. Many works using GAN for human motion prediction appeared, most of them applying and enhancing methodology developed for RNN, CNN or autoencoder to the GAN architecture. In [52] and [9], however, in order to model the temporal dependencies involved, they build their generator on RNN structures. In this way, the error accumulation problem is present in their model which may deteriorate its performance in the long term. [10] propose a GAN architecture that learns a probability density function to generate future human poses, enabling the generation of multiple futures based on a single historical sequence. Generating several possible future sequences may have advantages for some applications but also make it more difficult to evaluate the method as the standard metrics for human motion prediction are based on a comparison with the real motion sequence. [106] uses two generators to generate the motion, one for the lower body motion and one for the upper body motion. The lower body generator is conditioned on the prior sequence while the upper body generator is conditioned on the prior sequence and the generated lower body motion. Pursuing further the idea of separating the human body into parts [96] split the body into five parts each treated by a different GAN and then combined and presented to a global discriminator. [100] argue that methods based on the skeleton structure are difficult to generalize as different datasets can have different skeleton representations and propose a GAN that considers each skeleton joint separately and formulate them as a basic stochastic variable modeled using the Langevin equation. In our work, we completely discard recurrent models by adopting a compact representation of the human motion. Motivated by the interest of manifold-valued images in a variety of applications, [63] proposed manifold-aware WGAN. Inspired by this work, we build a manifold-aware WGAN that predicts the future points of a pose's trajectory given the

previous pose sequence. However, our model is different from the one proposed in [63] in two ways. Firstly, instead of unsupervised image generation from a vector noise, our model addresses the problem of predicting future manifold-valued representations from a manifold-valued input. In addition, we propose different objective functions to train our model on the task at hand

Modeling Human Motions as Trajectories on a Riemannian Manifold: While our present work is the first that explores the benefit of manifold-valued trajectories for human motion prediction, representing 3D human poses and their temporal evolution as trajectories on a manifold was adopted in many recent works for action recognition. Different manifolds were considered in different studies [143], [15], [70]. More related to our work, in [37], a human action is interpreted as a parametrized curve and is seen as a single point on the sphere by computing its Square Root Velocity Function (SRVF). Accordingly, different actions were classified based on the distance between their associated points on the sphere. All papers mentioned above show the effectiveness of motion modeling as a trajectory in action recognition. Motivated by this fact, we show in this chapter the interest of using such representation to address the recent challenges that are still encountered in human motion prediction.

2.3 Representation of Pose Sequences as Trajectories in $\mathbb{R}^n$

Let k be the number of joints that compose the skeleton, we represent P_t the pose of the skeleton at frame t by a n-dimensional tuple: $P_t = [x_1(t), y_1(t), z_1(t) \dots x_k(t), y_k(t), z_k(t)]^T$, The pose P_t encodes the positions of k distinct joints in 3 dimensions. Consequently, an action sequence of length T frames, can be described as a sequence $\{P_1, P_2 \dots, P_T\}$, where $P_i \in \mathbb{R}^n$ and $n = 3 \times k$.

This sequence represents the evolution of the action over time and can be considered as a result of sampling a continuous curve in $\mathbb{R}^n$. Based on this consideration, we model in what follows, each pose sequence of a skeleton, as a continuous curve in $\mathbb{R}^n$ that describes the continuous evolution of the sequence over time.
Let us represent the curve describing a pose sequence by a continuous parameterized function $\alpha(t) : I = [0, 1] \rightarrow \mathbb{R}^n$. In this work, we formulate the problem of human motion prediction given the first consecutive frames of the action as the problem of predicting the possible next points of the curve describing these first frames. More formally, the problem of predicting the future poses $\{P_{\tau+1}, P_{\tau+2}, \dots, P_T\}$, given the first τ consecutive skeleton poses $\{P_1, P_2, \dots, P_\tau\}$, where $\tau < T$, is formulated as the problem of predicting $\alpha(t)_{t=\tau+1\dots T}$ given $\alpha(t)_{t=1\dots\tau}$, such that, $\alpha(t)$ is the continuous function representing the curve associated to the pose sequence $\{P_1, P_2, \dots, P_T\}$.

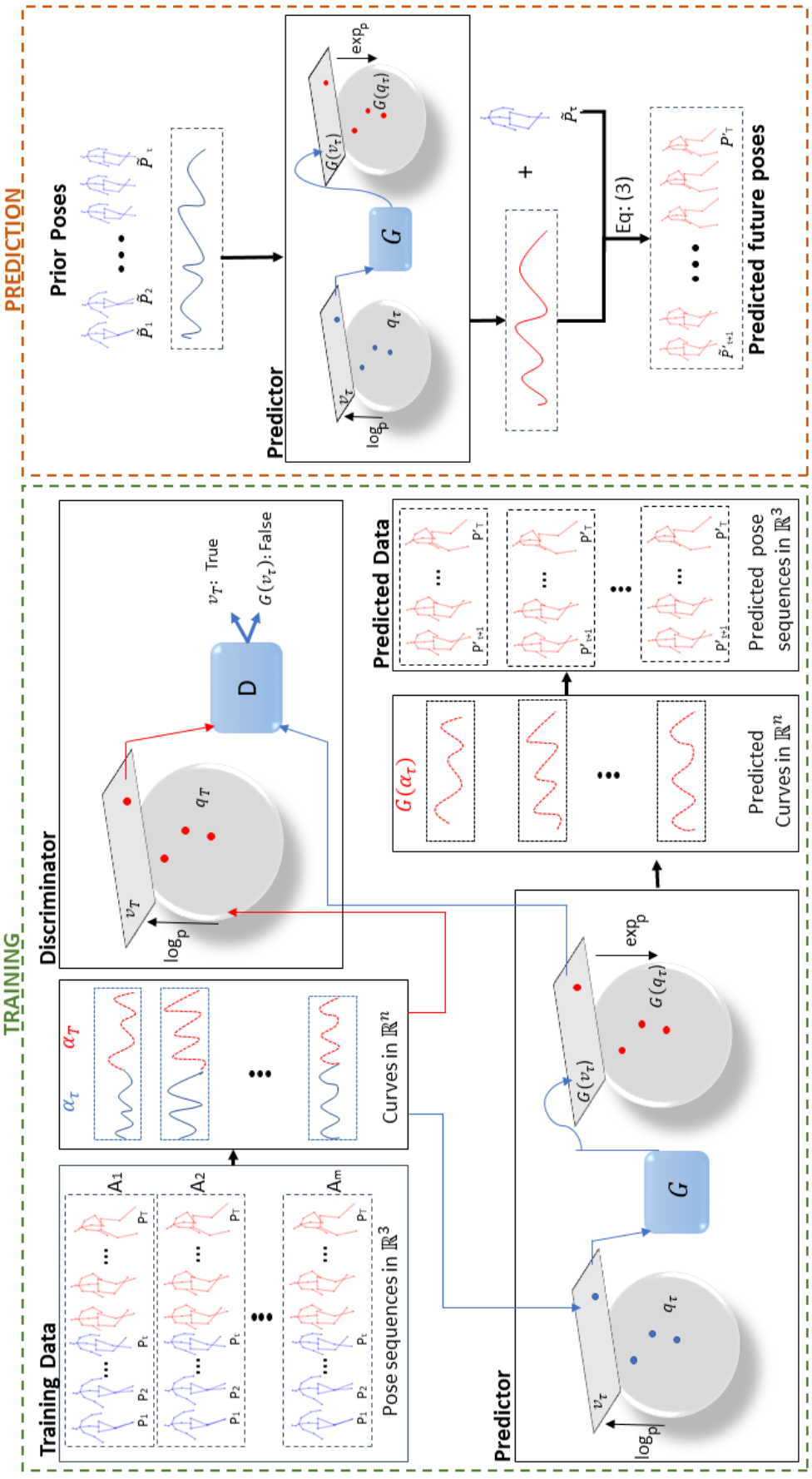


Figure 2.4: **Overview of the human motion training and prediction processes**. Given a pose sequence history represented as a curve, then mapped to a single point in a hypersphere. The predictor maps the input point to a tangent space, then feeds it to the network $\mathcal{G}$ that predicts the future motion as a vector in $T_\mu(\mathcal{C})$. During training, a discriminator is used to compare the mapped points from the ground truth to the generated ones. The exponential operator maps this vector to $\mathcal{C}$, before transforming it to a curve representing a motion. The predicted motion is transformed into a 3D human pose sequence corresponding to the future poses of the prior ones.

2.3.1 Representation of Human Motions as Elements in a Hypersphere $\mathcal{C}$

For the purpose of modeling and studying our curves, we adopt the square-root velocity function (SRVF) proposed in [137]. It was successfully exploited for human action recognition [37], 3D face recognition [40] and facial expression generation [112]. Conveniently for us, this function maps each curve $\alpha(t)$ to one point in a hypersphere which provides a compact representation of the human motion. Specifically, for a given curve $\alpha(t) : I \to \mathbb{R}^n$, the square-root velocity function (SRVF) $q(t) : I \to \mathbb{R}^n$ is defined by the formula

$$q(t) = \begin{cases} \frac{\dot{\alpha}(t)}{\sqrt{\|\dot{\alpha}(t)\|}}, & \text{if } \|\dot{\alpha}(t)\| \neq 0 \\ 0, & \text{if } \|\dot{\alpha}(t)\| = 0 \end{cases} \tag{2.2}$$

where, $\|\cdot\|$ is the Euclidean 2-norm in $\mathbb{R}^n$. We can easily recover the curve (*i.e*, pose sequence) $\alpha(t)$ from the generated SRVF (*i.e*, dynamic information) $q(t)$ by,

$$\alpha(t) = \int_0^t \|q(s)\| q(s) ds + \alpha(0) \ , \tag{2.3}$$

where $\alpha(0)$ is the skeleton pose at the initial time step which corresponds in our case to the final time step of the history. In order to remove the scale variability of the curves, we scale them to be of length 1. Consequently, the SRVFs corresponding to these curves are elements of a unit hypersphere in the Hilbert manifold $\mathbb{L}^2(I, \mathbb{R}^n)$ as explained in [137]. We will refer to this hypersphere as $\mathcal{C}$, such that, $\mathcal{C} = \{q : I \to \mathbb{R}^n | \ \|q\| = 1\} \subset \mathbb{L}^2(I, \mathbb{R}^n)$. Each element of $\mathcal{C}$ represents a curve in $\mathbb{R}^n$ associated with a human motion. As $\mathcal{C}$ is a hypersphere, the geodesic length between two elements q_1 and q_2 is defined as:

$$d_{\mathcal{C}}(q_1, q_2) = \cos^{-1}(\langle q_1, q_2 \rangle) \ . \tag{2.4}$$

2.4 Architecture and Loss Functions

Given a set of m action sequences $\{\{P_1, P_2, \dots P_T\}_i\}_{i=1}^m$ of T consecutive skeleton poses. Let us consider the first τ poses ($\tau < T$) as the actions history represented by their corresponding SRVFs $\{q_\tau^i\}_{i=1}^m$, and the last $(T - \tau)$ skeleton configurations as the future poses $\{q_T^i\}_{i=1}^m$ to be predicted.
Motivated by the success of generative adversarial networks, we aim to exploit these generative models to learn an approximation of the function $\Phi : \mathcal{C} \to \mathcal{C}$ that predicts the $(T - \tau)$ future poses from their associated τ prior ones. This can be achieved by learning the distribution of SRVFs data corresponding to future poses, on their underlying manifold *i.e.*, hypersphere. As stated earlier, SRVFs representations are manifold-valued data that cannot be used directly by classical GANs. This is due to the fact that the distribution of data having values on a manifold is quite different from the distribution of those lying on Euclidean space. [63], exploited the tangent space of the involved manifold and propose a manifold-aware WGAN that generates random data on a manifold. Inspired by this work, we propose a

manifold-aware WGAN for motion prediction, to which we refer as PredictiveMA-WGAN, that can predict future poses from past ones. This is achieved by using the prior poses as input conditions to the MA-WGAN. This condition is also represented by its SRVF; as a result, PredictiveMA-WGAN takes manifold-valued data as input to predict its future, which is also manifold-valued data.

2.4.1 Network Architecture

PredictiveMA-WGAN consists of two networks trained in an adversarial manner: the predictor $\mathcal{G}$ and the discriminator $\mathcal{D}$. The first network $\mathcal{G}$ adjusts its parameters to learn the distribution $\mathbb{P}_{q_T}$ of the future poses q_T conditioned on the input prior ones q_τ, while *D* tries to distinguish between the real future poses q_T and the predicted ones $\hat{q}_T$. During the training of these networks, we iteratively map the SRVF data back and forth to the tangent space using the exponential and the logarithm maps, defined at a particular point on the hypersphere.

The predictor network is composed of multiple upsampling and downsampling blocks. It takes as input the prior poses q_τ and outputs the predicted future poses $\hat{q}_T$. A fully connected layer with 36864 output channels and five upsampling blocks with 512, 256, 128, 64, and 1 output channels, processes the input prior pose. These upsampling blocks are composed of the nearest-neighbor upsampling followed by a 3×3 stride 1 convolution and a Relu activation. The Discriminator *D* contains three downsampling blocks with 64, 32, and 16 output channels. Each block is a 3×3 stride 1 Conv layer followed by batch normalization and Relu activation. These layers are then followed by two fully connected (FC) layers of 1024 and 1 outputs. The first FC layer uses Leaky ReLU and batch normalization.

2.4.2 Loss Functions

In general, the objective of the training consists in minimizing the Wasserstein distance between the distribution of the predicted future poses $\mathbb{P}_{\hat{q}_T}$ and that of the real ones $\mathbb{P}_{q_T}$ provided by the dataset. Toward this goal we make use of the following loss functions:

Adversarial loss – We propose an adversarial loss for predicting manifold-valued data from their history. The predictor takes a manifold-value data q_τ as input rather than a random vector as done in [63], which requires mapping these data to a tangent space using the logarithm map before feeding them to the network. Our adversarial loss is the following:

$$\begin{aligned}\mathcal{L}_a =& \mathbb{E}_{q_T \sim \mathbb{P}_{q_T}} \left[\mathcal{D}\left(\log_\mu(q_T)\right)\right] \\ & - \mathbb{E}_{\mathcal{G}(log_\mu(q_\tau)) \sim \mathbb{P}_{\hat{q}_T}} \left[\mathcal{D}\left(\log_\mu\left(\exp_\mu(\mathcal{G}(\log_\mu(q_\tau)))\right)\right)\right] \\ & + \lambda \mathbb{E}_{\widetilde{q} \sim \mathbb{P}_{\widetilde{q}}} \left[\left(\|\nabla_{\widetilde{q}} \mathcal{D}(\widetilde{q})\| - 1\right)^2\right],\end{aligned} \tag{2.5}$$

where the exponential map, $\exp_\mu(.)$: $T_\mu(C) \mapsto \mathcal{C}$ has a simple expression:

$$\exp_\mu(s) = cos(\|s\|)\mu + sin(\|s\|)\frac{s}{\|s\|},$$

and the inverse exponential map also called logarithm map $\log_\mu(q)$: $\mathcal{C} \mapsto T_\mu(C)$ is given by:

$$\log_\mu(q) = \frac{d_\mathcal{C}(q,\mu)}{sin(d_\mathcal{C}(q,\mu))}(q - cos(d_\mathcal{C}(q,\mu))\mu)$$

where $d_\mathcal{C}(.,.)$ is the geodesic distance defined by (2.4). The last term of $\mathcal{L}_a$ represents the gradient penalty proposed in [53]. $\widetilde{q}$ is a random sample following the distribution $\mathbb{P}_{\widetilde{q}}$, which is sampled uniformly along straight lines between pairs of points sampled from the real distribution $\mathbb{P}_{q_T}$ and the generated distribution $\mathbb{P}_{\hat{q}_T}$. It is given by: $\widetilde{q} = (1-a)\log_\mu(q_T) + a\log_\mu(\exp_\mu(\mathcal{G}(\log_\mu(q_\tau))))$, where $\nabla_{\widetilde{q}} D(\widetilde{q})$ is the gradient with respect to $\widetilde{q}$, and $0 \leqslant a \leqslant 1$. The reference point μ of the tangent space used in our training is set to the mean of the training data. For a given set of training trajectories $q_1, \ldots, q_m$. The mean is given by the Karcher mean [72] in $\mathcal{C}$,

$$\mu = \underset{q_i \in \mathcal{C}}{\operatorname{argmin}} \sum_{i=1}^{m} d^2_\mathcal{C}(\mu, q_i) \tag{2.6}$$

where $\{q_i\}_{i=1}^m$ is m training data. We present a commonly used algorithm for finding Karcher mean for a given set of curves [136]. This approach is presented in Algorithm 1. This computation is based on an iterative calculation that converges to the optimal solution which is the mean.

Algorithm 1: Karcher mean on $\mathcal{C}$

Input: Given SRVFs $\{q_1, q_2 \cdots q_N\}$,
$\epsilon = 0.9$, τ: threshold which is a very small number
Output: μ_j : mean of $\{q_i\}_{i=1:N}$

1- μ_0: initial estimate of Karcher mean, for example, one could just take $\mu_0 = q_1$,
j=0
repeat
  for $i \leftarrow 1$ **to** N **do**
    2- Compute $v_i = \frac{\theta_i}{sin(\theta_i)}(q_i^* - cos(\theta_i)\mu_j)$, where $cos(\theta_i) = \langle \mu_j, q_i^* \rangle$
    3- Compute the average direction $\overline{v} = \frac{1}{n}\sum_{i=1}^n v_i$
    4- Move μ_j in the direction of $\overline{v}$ by ϵ: $\mu_{j+1} = cos(\epsilon\|\overline{v}\|)\mu_j + sin(\epsilon\|\overline{v}\|)\frac{\overline{v}}{\|\overline{v}\|}$
  5- j=j+1
until $\|\overline{v}\| < \tau$;

Reconstruction loss – In order to predict motions close to their ground truth, we add a reconstruction loss $\mathcal{L}_r$. This loss function quantifies the similarities in the tangent space $T_\mu(\mathcal{C})$ between the tangent vector $\log_\mu(q_T)$ of the ground truth q_T and its associated reconstructed vector $\log_\mu(\exp_\mu(\mathcal{G}(\log_\mu(q_\tau))))$. It is given by,

$$\mathcal{L}_r = \|\log_\mu(\exp_\mu(\mathcal{G}(\log_\mu(q_\tau)))) - \log_\mu(q_T)\|_1 \ , \tag{2.7}$$

where $\|.\|_1$ denotes the L_1-norm.

Skeleton integrity loss – We propose a new loss function $\mathcal{L}_s$ that minimizes the distance between the predicted poses and their ground truth as a remedy to the generation of abnormal skeleton poses. Indeed, the aforementioned loss functions rely only on the SRVF representations, which impose constraints only on the dynamic information. However, to capture the spatial dependencies between joints that avoid implausible poses, we need to impose constraints on the predicted poses directly instead of their motions. By doing so, we predict dynamic changes that fit the initial pose and result in long-term plausibility. The proposed loss function is based on the Gram matrix of the joint configuration P, $G = PP^T$, where P can be seen as $k \times 3$ matrix. Let G_i, G_j be two Gram matrices, obtained from joint poses $P_i, P_j \in \mathbb{R}^{k\times 3}$. The distance between G_i and G_j can be expressed [50, p. 328] as:

$$\Delta(G_i, G_j) = \mathrm{tr}\,(G_i) + \mathrm{tr}\,(G_j) - 2\sum_{i=1}^{3}\sigma_i \ , \tag{2.8}$$

where $\mathrm{tr}(.)$ denotes the trace operator, and $\{\sigma_i\}_{i=1}^{3}$ are the singular values of $P_j^T P_i$. The resulting loss function is,

$$\mathcal{L}_s = \frac{1}{m}\frac{1}{\tau}\sum_{i=1}^{m}\sum_{t=1}^{\tau}\Delta(P_{i,t}, \hat{P}_{i,t}) \ , \tag{2.9}$$

where m represents the number of training samples, τ is the length of the predicted sequence, P is the ground truth pose and $\hat{P}$ is the predicted one.

Bone length loss – To ensure the realness of the predicted poses, we impose further restrictions on the length of the bones. This is achieved through a loss function that forces the bone length to remain constant over time. Considering $b_{i,j,t}$ and $\hat{b}_{i,j,t}$ the j-th bones at time t from the ground truth and the predicted i-th skeleton, respectively, we compute the following loss :

$$\mathcal{L}_b = \frac{1}{m}\frac{1}{\tau}\frac{1}{B}\sum_{i}^{m}\sum_{t=1}^{\tau}\sum_{j}^{B}\|b_{i,j,t} - \hat{b}_{i,j,t}\| \ , \tag{2.10}$$

with B the number of bones in the skeleton representation.

Global loss – PredictiveMA-WGAN is trained using a weighted sum of the four loss functions $\mathcal{L}_a$, $\mathcal{L}_r$, $\mathcal{L}_s$ and $\mathcal{L}_b$ introduced above, such that,

$$\mathcal{L} = \beta_1\mathcal{L}_a + \beta_2\mathcal{L}_r + \beta_3\mathcal{L}_s + \beta_4\mathcal{L}_b. \tag{2.11}$$

The parameters β_i are the coefficients associated to different losses, they are set empirically in our experiments.

The algorithm 2 summarizes the main steps of our approach. It is divided into two stages, first, we outline the steps needed to train our model, then we present the prediction stage, where the trained model is used to predict future poses of a given sequence.

Algorithm 2: PredictiveMAWGAN algorithm

// Training

Data: $\{q_\tau^i\}_{i=1}^m$: SRVFs of training prior poses, $\{q_T^i\}_{i=1}^m$: real future poses, θ_0 : initial parameters of $\mathcal{G}$, η_0 : initial parameters of $\mathcal{D}$, ϵ: learning rate, K: batch size, λ: balance parameter of gradient penalty, ζ: iterations number.

Result: θ: generator learned parameters.

for $i = 1 \dots \zeta$ **do**

- Sample a mini-batch of K random prior poses $\{q_\tau^j\}_{j=1}^K \sim \mathbb{P}_{q_\tau}$;
- Sample a mini-batch of K real future poses; $\{q_T^j\}_{j=1}^K \sim \mathbb{P}_{q_T}$;
- $D_\eta \leftarrow \Delta_\eta(\mathcal{L}), \mathcal{L}$ is given by Eq. 10;
- $\eta \leftarrow \eta + \epsilon.AdamOptimizer(\eta, D_\eta)$;
- Sample a mini-batch of K random prior poses; $\{q_\tau^j\}_{j=1}^K \sim \mathbb{P}_{q_\tau}$;
- Compute $\{\mathcal{G}_\theta(\log_\mu(q_\tau^j))\}_{j=1}^K$;
- $G_\theta \leftarrow \Delta_\theta(-D_\eta\left(\log_\mu\left(\exp_\mu(\mathcal{G}_\theta(\log_\mu(q_\tau)))\right)\right))$
- $\theta \leftarrow \theta + \epsilon.AdamOptimizer(\theta, G_\theta)$;

// Prediction

Data: θ: generator learned parameters,
$\{P_i\}_{i=1}^\tau$: Prior poses of a testing sequence.

Result: $\{\hat{P}_i\}_{i=\tau+1}^T$: Predicted future poses.

Compute q_τ from $\{P_i\}_{i=1}^\tau$ with Eq. 1;

Compute $\hat{q}_T = \exp_\mu(\mathcal{G}_\theta(\log_\mu(q_\tau)))$ using the learned parameters θ;

Transform $\hat{q}_T$ into pose sequence $\{\hat{P}_i\}_{i=\tau+1}^T$ using Eq. 2, with $\alpha(0) = P_\tau$

2.5 Experiments

We evaluate the proposed approach with extensive experiments on two popular datasets. In this part, we show and discuss our results.

2.5.1 Datasets and Pre-processing

Human 3.6M [64]. it is a database that contains 11 subjects performing 15 different actions (Walking, Phoning, Taking photos…). It is one of the largest datasets and the most commonly used for evaluating human motion prediction with 3D skeletons. Following the protocol set by previous approaches [109, 34] we train our model on 6 subjects and test it on the specific clips of the 5th subject. In the same way as [34] out of the 32 skeletal joints we only use 17, we remove the joints that correspond to duplicate joints, hands, and feet.

For Human3.6M we take the database processed by [65] formatted in exponential map and we use their code to convert them to Cartesian coordinates. During our preprocessing step, we down-sample the sequence from 50 fps to 25 fps and then perform normalization by subtracting the mean, dividing by the norm, and subtracting the coordinates of the root

joint (hips). In the dataset proposed by [65] each class of each subject is composed of 2 long sequences. We divide those into smaller sequences for short-term prediction (60 frames) and long-term prediction (75 frames), following [87]. When generating these smaller sequences we avoid overlap, *e.g.* when generating sequences for long-term prediction (75 frames) the first sequence contains the frames 1 to 75, the second frames 76 to 150, and so on. This leaves us with 3480 training samples and 812 testing samples for short-term prediction and 2769 training samples and 644 testing samples for long-term prediction.

CMU Motion Capture (CMU MoCap). CMU Mocap dataset [1] is a database that contains 5 categories of motion, each containing several actions. Following [87], we keep only 8 actions: 'basketball', 'basketball signal', 'directing traffic', 'jumping', 'running', 'soccer', 'walking', and 'washing window'. We keep the same joint configuration as for Human3.6M and preprocess the data the same way. This leads to 2871 training samples and 704 test samples for short-term prediction and 2825 training samples and 677 test samples for long-term prediction.

2.5.2 Implementation Details

We train our network with a batch size of 64 on 500 epochs and with a learning rate of 10^{-4} using the Adam optimizer [78]. We use $\beta_1 = 1$, $\beta_2 = 1$, $\beta_3 = 10$ and $\beta_4 = 10$ for the loss coefficients. Our Implementation runs on a PC with a Nvidia Quadro RTX 6000 GPU, two 2.3Ghz processors, and 64Go of RAM using Tensorflow 2.2.

2.5.3 Evaluation Metrics and Baselines

We use state-of-the-art methods for motion prediction that were based on 3D coordinate representation for our comparison. This includes RNN-based methods (Residual sup). [109], CNN based method (ConvSeq2Seq) [87] and graph models; (FC-GCN) [150] and (LDRGCN) [34].

The zero velocity baseline introduced by [109] is a very simple baseline that uses the last observed frame at $t = \tau$ as the value for all the predicted frames, we also compare ourselves to this baseline. The results of LDRGCN are those reported by the authors for the method trained with data in 3D coordinate space. Concerning FC-GCN, ConvSeq2Seq, and Residual sup., the results are those reported by [150] using 3D coordinate data for training. We report the results presented by [34] for long-term prediction (1000ms) results on Human 3.6M since they are not provided in [150]. The long-term results for Residual sup. are not available, we did not include it in our results.

We base our quantitative evaluation on the Mean Per Joint Position Error (MPJPE) [64] in millimeters following the state-of-the-art [34]. The metric compares the 3D coordinates of the ground truth with the predicted motions. It is given by,

[1] http://mocap.cs.cmu.edu

$$MPJPE = \sqrt{\frac{1}{\Delta t}\frac{1}{k}\sum_{t=\tau+1}^{\tau+\Delta t}\sum_{j=1}^{k}\|p_{t,j} - \hat{p}_{t,j}\|^2} \ , \tag{2.12}$$

where $p_{t,j} = [x_j(t), y_j(t), z_j(t)]$ are the coordinates of joint j at time t from the ground truth sequence, $\hat{p}_{t,j}$ the coordinates from the generated sequence, k the total number of joints in the skeleton, τ the number of frames in prior sequence and Δt the number of predicted frames at which the sequence is evaluated.

While MPJPE evaluates the generated samples based on joint positions, it is not enough to assess the evolution of the motion. To complete our assessment we further compare our method with the other approaches based on the evolution along time of the speed of the predicted sequences, we refer to this metric as MPJS (Mean Per Joint Speed). It is computed as follows,

$$MPJS(t) = \frac{1}{k}\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{k}\|p_{i,t-1,j} - p_{i,t,j}\| \ , \tag{2.13}$$

with $p_{i,t-1,j}$ and $p_{i,t,j}$ the coordinates of joint j at time t-1 and t respectively, k the number of joints in the skeleton and M the total number of samples in the test set.

2.5.4 Quantitative Comparison

Joints position-based evaluation

To be consistent with recent works, the results are reported for short-term prediction and long-term prediction. For short-term prediction, we predict 10 future frames within 400ms given 10 historical frames while we predict 25 in 1s based on 25 prior frames for long-term prediction. In Table 2.1 we show the comparison of our results with recent methods that use 3D joint coordinates representation. This representation has been proven to provide a more reliable comparison than the angle-based representation by [150]. The results in the table show the clear superiority of our method over methods from the state-of-the-art on both datasets. We highlight that our approach is very competitive with the LDRGN approach for very short-term prediction (80ms and 160ms) while outperforming it for longer prediction (320ms, 400ms, and 1s). This demonstrates that it is robust when predicting long-term motions that stay close to the ground truth.

In Table 2.2 and 2.3 we report the results for the literature and for our method on all action classes of Human3.6M and CMU Mocap datasets respectively. The baseline methods adopt a protocol that consists in reporting the average error on eight randomly sampled test sequences. We found that this random sampling can significantly affect the error and makes it hard to present a fair comparison. To avoid this, we decided to report to run the experiment on 8 randomly selected test sequences 100 times, we then report the average error and the standard deviation for these 100 runs for the results of our model. With the standard deviation, we can have a better measurement of the general performance of our architecture on different test sequences.

	Human3.6M average					CMU MoCap average				
millisecond (ms)	80	160	320	400	1000	80	160	320	400	1000
Zero velocity	19.6	32.5	55.1	64.4	107.9	18.4	31.4	56.2	67.7	130.5
Residual sup.	30.8	57.0	99.8	115.5	-	15.6	30.5	54.2	63.6	96.6
convSeq2Seq	19.6	37.8	68.1	80.3	140.5	12.5	22.2	40.7	49.7	84.6
FC-GCN	12.2	25.0	50.0	61.3	114.7	11.5	20.4	37.8	46.8	96.5
LDRGCN	**10.7**	**22.5**	43.1	55.8	97.8	**9.4**	17.6	31.6	43.1	82.9
Ours	12.6	**22.5**	**41.9**	**50.8**	**96.4**	**9.4**	**15.9**	**29.2**	**38.3**	**80.6**

Table 2.1: Average error over all actions of Human3.6M and CMU MoCap. The short-term in 80, 160, 320, $400ms$, and long-term in $1s$.

According to Tables 2.2 and 2.3, our method performs better than the state-of-the-art, especially when dealing with long-term prediction, these results are consistent with the average error over all actions classes. Interestingly our results also show that the simple zero velocity baseline sometimes outperforms the state-of-the-art approach on long-term prediction (*e.g,* "Photo", "Sitting" and "Walking dog" for Human3.6M, "Soccer" and "Jumping" for CMU MoCap). On the other hand on short-term prediction, it is always outperformed by the prediction methods. This may be an indication that the MPJPE is not the best-suited metric for the problem and a motivation to find a better more representative metric in future works. The results show that the previous approaches' performances decrease over time, while ours proves more robust in long-term horizons, we are shown to perform better than both the zero velocity baseline and the literature. We can notice that some classes present a very large variance (*e.g,* jumping) while for others the variance is very low (*e.g,* running). This is due to the number of samples which can be very different from one class to another but also to the high diversity of samples for some classes. Other classes that present less variability (*e.g,* walking) have a reduced variance.

Motion-based evaluation

To further assess the generated sequences, we evaluate their motion based on the MPJS introduced before. By looking at the evolution of this metric, we can compare our generated motion with the ground truth ones and evaluate the ability of our model to predict motion in long-term prediction. To this end, we show in Figures 2.5 the evolution of the MPJS over time steps on the Human3.6M dataset.
Results show that the average ground truth speed varies slightly around 9 *mm/frame*. The zero velocity baseline obviously shows the worst results as no motion is being produced and the speed is always null. Both FC-GCN and ConvSeq2Seq speeds continuously decrease over time meaning that less motion is being produced on average for long-term horizons. On the other hand, our method shows variations in the average produced speed, up to $1s$ with even an increase during long-term prediction. The results from Tables 2.2 and 2.1 show that the pose error is good for long-term prediction, meaning that the increase in average speed does not correspond to a degradation in the quality of the prediction. This supports our claim that our method is a good fit to predict long motions that keep their spatial and temporal coherency. We report in Figure 2.6 the evolution of MPJS for the CMU dataset in the same

	Directions					Discussion					Eating					Greeting				
millisecond (ms)	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000
Zero velocity	16.0	27.1	46.4	53.9	83.9	17.8	29.7	51.0	59.8	103.1	13.5	21.9	37.0	43.9	83.3	26.4	43.7	70.1	80.5	**124.9**
Residual sup.	36.5	56.4	81.5	97.3	-	31.7	61.3	96.0	103.5	-	17.6	34.7	71.9	87.7	-	37.9	74.1	139.0	158.8	-
convSeq2Seq	22.0	37.2	59.6	73.4	118.3	18.9	39.3	67.7	75.7	123.9	13.7	25.9	52.5	63.3	74.4	24.5	46.2	90.0	103.1	191.2
FC-GCN	12.6	24.4	48.2	58.4	89.1	9.8	22.1	39.6	44.1	78.5	8.8	18.9	39.4	47.2	57.1	14.5	30.5	74.2	89.0	148.4
LDRGCN	13.1	23.7	44.5	50.9	**78.3**	**9.4**	**20.3**	**35.2**	**41.2**	**67.4**	**7.6**	**15.9**	37.2	41.7	**53.8**	**9.6**	**27.9**	66.3	78.8	129.7
Ours	**11.1**	**20.9**	**38.8**	**47.0**	**83.5**	11.9	**22.7**	44.8	54.6	102.2	**9.0**	**15.9**	**29.1**	**35.0**	65.3	19.6	35.1	**64.0**	**78.2**	**126.8**
	±2.7	±4.9	±8.4	±9.7	±15.3	±1.9	±3.4	±6.5	±7.7	±16.5	±1.5	±2.8	±4.8	±5.3	±6.8	±3.4	±6.8	±13.1	±16.1	±16.7

	Phoning					Photo					Posing					Purchase				
millisecond (ms)	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000
Zero velocity	15.8	26.5	43.7	51.0	92.3	16.9	28.4	49.2	58.3	**98.8**	20.4	34.7	61.5	73.3	136.1	22.1	36.5	61.8	72.2	126.3
Residual sup.	25.6	44.4	74.0	84.2	-	23.6	47.4	94.0	112.7	-	27.9	54.7	131.3	160.8	-	40.8	71.8	104.2	109.8	-
convSeq2Seq	17.2	29.7	53.4	61.3	127.5	14.0	27.2	53.8	66.2	151.2	16.1	35.6	86.2	105.6	163.9	29.4	54.9	82.2	93.0	139.3
FC-GCN	11.5	20.2	37.9	43.2	94.3	**6.8**	15.2	38.2	49.6	125.7	9.4	23.9	66.2	82.9	143.5	19.6	38.5	64.4	72.2	127.2
LDRGCN	**10.4**	**14.3**	**33.1**	**39.7**	85.8	7.1	**13.8**	**29.6**	44.2	116.4	**8.7**	**21.1**	58.3	81.9	**133.7**	16.2	36.1	62.8	76.2	**112.6**
Ours	**11.7**	19.4	**34.9**	**42.3**	**81.8**	**8.8**	**16.0**	**32.4**	**40.9**	**98.9**	13.7	**25.9**	**50.0**	**61.1**	**137.7**	**14.2**	**26.5**	**48.3**	**58.1**	**120.8**
	±2.2	±3.6	±6.4	±7.6	±9.8	±2.0	±3.5	±6.9	±8.6	±16.1	±3.3	±6.3	±11.0	±12.7	±12.8	±2.5	±4.8	±9.8	±12.5	±19.0

	Sitting					Sitting Down					Smoking					Waiting				
millisecond (ms)	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000
Zero velocity	14.6	23.9	40.9	48.4	**94.7**	19.5	32.4	53.5	61.8	**112.2**	14.9	24.6	41.7	49.3	84.0	17.0	28.2	48.9	57.8	99.4
Residual sup.	34.5	69.9	126.3	141.6	-	28.6	55.3	101.6	118.9	-	19.7	36.6	61.8	73.9	-	29.5	60.5	119.9	140.6	-
convSeq2Seq	19.8	42.4	77.0	88.4	132.5	17.1	34.9	66.3	77.7	177.5	11.1	21.0	33.4	38.3	52.2	17.9	36.5	74.9	90.7	205.8
FC-GCN	10.7	24.6	50.6	62.0	119.8	11.4	27.6	56.4	67.6	163.9	7.8	14.9	25.3	28.7	44.3	9.5	22.0	57.5	73.9	157.2
LDRGCN	**9.2**	23.1	47.2	57.7	106.5	**9.3**	**21.4**	**46.3**	**59.3**	144.6	8.1	**13.4**	**24.8**	**24.9**	**43.1**	**9.2**	**17.6**	47.2	71.6	127.3
Ours	**10.4**	**17.9**	**33.1**	**40.7**	**97.7**	15.8	28.2	**52.9**	**64.5**	**125.2**	**7.9**	**14.3**	**25.2**	**30.4**	63.4	**11.4**	**20.3**	**38.8**	**47.2**	**94.0**
	±2.8	±3.5	±5.3	±6.4	±14.0	±3.4	±5.1	±9.3	±11.5	±23.3	±1.6	±2.7	±4.5	±5.2	±9.7	±3.1	±4.3	±7.6	±9.0	±13.7

	Walking Dog					Walking					Walking Together					Average				
millisecond (ms)	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000
Zero velocity	26.9	42.3	69.2	79.5	119.2	28.1	49.2	86.0	100.3	149.1	23.5	39.2	65.4	75.6	111.3	19.6	32.5	55.1	64.4	107.9
Residual sup.	60.5	101.9	160.8	188.3	-	23.8	40.4	62.9	70.9	-	23.5	45.0	71.3	82.8	-	30.8	57.0	99.8	115.5	-
convSeq2Seq	40.6	74.7	116.6	138.7	210.2	17.1	31.2	53.8	61.5	89.2	15.0	29.9	54.3	65.8	149.8	19.6	37.8	68.1	80.3	140.5
FC-GCN	32.2	58.0	102.2	122.7	185.4	**8.9**	15.7	29.2	33.4	50.9	8.9	18.4	35.3	44.3	102.4	12.2	25.0	50.0	61.3	114.7
LDRGCN	25.3	56.6	87.9	99.4	143.2	**8.9**	**14.9**	**25.4**	**29.9**	**45.8**	**8.2**	**18.1**	**31.2**	**39.4**	79.2	**10.7**	**22.5**	43.1	55.8	97.8
Ours	**19.3**	**34.2**	**65.6**	**77.5**	**117.8**	12.0	21.1	35.6	42.4	68.2	11.6	**19.7**	34.5	**41.8**	**63.4**	12.6	**22.5**	**41.9**	**50.8**	**96.4**
	±5.9	±9.5	±17.8	±19.7	±23.7	±1.1	±1.7	±2.9	±3.8	±5.3	±1.1	±1.6	±3.0	±3.8	±6.4						

Table 2.2: Motion prediction results measured with Equation 2.12 on Human3.6M dataset. Short-term results are reported within 80, 160, 320, 400ms, and long-term in 1s. Best results in bold while state-of-the-art best results that fit in our confidence interval are also written in bold.

	Basketball					Basketball signal					Directing traffic					Jumping				
millisecond (ms)	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000
Zero velocity	20.3	34.6	62.2	75.0	143.5	6.4	11.0	19.9	24.2	50.5	26.6	41.9	69.1	81.9	155.3	21.4	36.3	63.2	75.2	138.8
Residual sup.	18.4	33.8	59.5	70.5	106.7	12.7	23.8	40.3	46.7	77.5	15.2	29.6	55.1	66.1	127.1	36.0	68.7	125.0	145.5	195.5
convSeq2Seq	16.7	30.5	53.8	64.3	**91.5**	8.4	16.2	30.8	37.8	76.5	10.6	20.3	38.7	48.4	**115.5**	22.4	44.0	87.5	106.3	162.6
FC-GCN	14.0	25.4	49.6	61.4	106.1	3.5	6.1	11.7	15.2	53.9	7.4	**15.1**	31.7	42.2	152.4	16.9	34.4	76.3	96.8	164.6
LDRGCN	13.1	22.0	37.2	55.8	97.7	3.4	6.2	**11.2**	**13.8**	47.3	**6.8**	16.3	**27.9**	**38.9**	131.8	13.2	32.7	65.1	91.3	153.5
Ours	**9.1**	**16.6**	**34.7**	**44.5**	108.4	**3.3**	**5.9**	**11.5**	**14.7**	**44.7**	**19.6**	**31.3**	**54.8**	**66.1**	**155.5**	**12.5**	**22.7**	**44.4**	**55.8**	**120.4**
	±0.7	±1.5	±3.5	±4.4	±5.1	±1.1	±2.0	±3.7	±4.7	±15.0	±16.7	±23.2	±34.4	±37.4	±52.1	±2.0	±3.8	±7.4	±9.6	±21.0

	Running					Soccer					Walking					Wash window				
millisecond (ms)	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000	80	160	320	400	1000
Zero velocity	30.6	52.8	94.1	112.2	242.6	10.3	17.5	31.8	39.0	79.4	18.3	31.2	55.1	66.2	137.7	12.3	21.1	37.8	45.7	90.9
Residual sup.	15.6	19.4	31.2	36.2	43.3	20.3	39.5	71.3	84.0	129.6	8.2	13.7	21.9	24.5	**32.2**	8.4	15.8	29.3	35.4	61.1
convSeq2Seq	14.3	**16.3**	**18.0**	**20.2**	**27.5**	12.1	21.8	41.9	52.9	94.6	7.6	12.5	23.0	27.5	49.8	8.2	15.9	32.1	39.9	58.9
FC-GCN	25.5	36.7	39.3	39.9	58.2	11.3	21.5	44.2	55.8	117.5	7.7	11.8	19.4	23.1	40.2	5.9	11.9	30.3	40.0	79.3
LDRGCN	15.2	19.7	23.3	35.8	47.4	10.3	21.1	42.7	50.9	91.4	**7.1**	**10.4**	**17.8**	**20.7**	37.5	5.8	12.3	27.8	38.2	**56.6**
Ours	**12.4**	19.7	32.3	39.0	68.9	**4.9**	**7.9**	**14.2**	**18.0**	**53.1**	8.1	13.6	22.1	26.1	**32.4**	**5.5**	**9.8**	**19.2**	**24.3**	**61.3**
	±0.0	±0.0	±0.0	±0.0	±0.0	±0.3	±0.6	±1.2	±1.6	±4.5	±0.0	±0.0	±0.0	±0.0	±1.8	±0.8	±1.4	±2.5	±3.1	±8.4

Table 2.3: Motion prediction results measured with Equation 2.12 on CMU dataset. Short-term results are reported within 80, 160, 320, 400ms, and long-term in 1s. Best results in bold while state-of-the-art best results that fit in our confidence interval are also written in bold.

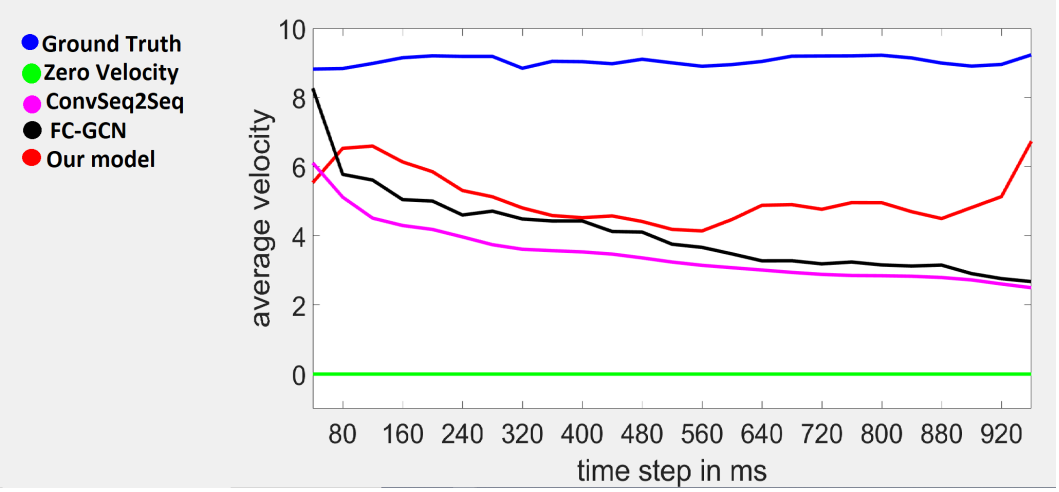


Figure 2.5: The average speed (MPJS) evolution over 1000 *ms* of all action classes of the Human3.6M dataset.

way as for Human3.6M. We do not report the results for FC-GCN as there was no pre-trained model available and we were unable to train their model ourselves. The results are similar to those of Human3.6M: average speed of prediction method lower than the ground truth, constant decrease for ConvSeq2Seq, more variation for our method with a significant increase in long-term horizons. However, we see that this time for short-term prediction ConvSeq2Seq produces an average speed closer to the ground truth than ours. This seems to confirm the MPJPE results for ConvSeq2Seq that show that it performs better on CMU than on HUMAN3.6M, we still perform better for long-term prediction, showing that our method is robust to different kinds of datasets. Interestingly however all evaluated methods have average speeds well below the values of the ground truth. While this may be explained in part by the presence of sudden, high amplitude and hard-to-predict motion in action classes like Direction or Greeting, it still indicates that using losses that solely constraint the poses during training leads to generating sequences with slower motion since fast motions are more prone to error. This might be a hint that using losses on the speed of the motion will help produce even better predictions.

We present in Table 2.4 the average MPJS for each class over all time steps on Human3.6M. In consistency with Figure 2.5 this table shows a significant difference between the ground truth and generated motions with all methods. However, this difference changes significantly between classes; some classes like Walking Dog or Greeting present a high difference (6.19 and 4.89 respectively when compared with our method) while others have a lower difference like Eating or Smoking (1.93 and 1.70 respectively when compared with our method). Furthermore, our method is still able to outperform ConvSeq2Seq and FC-GCN on all classes except Walking and Walking together where FC-GCN performs better indicating a capability to better model periodical motion. On the other hand for non-periodic motion, our method outperforms FC-GCN by a large margin (Greeting, Sitting Down, etc.).

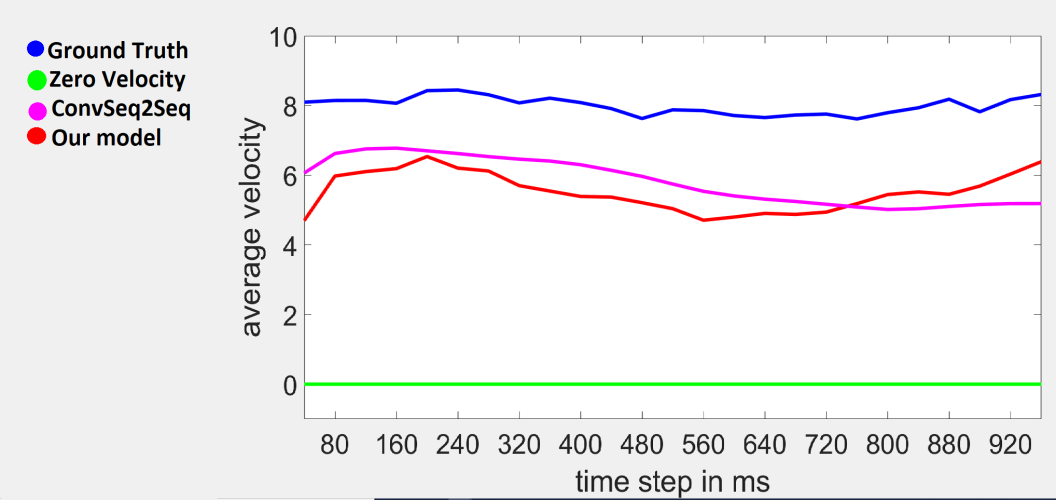


Figure 2.6: The average speed (MPJS) evolution over 1000 *ms* of all action classes of the CMU MoCap dataset.

	Ground truth	ConvSeq2Seq	FC-GCN	Ours
Direction	5.97	2.43	2.39	3.41
Discussion	8.42	3.03	3.28	4.7
Eating	6.24	3.35	3.77	4.31
Greeting	11.54	3.41	3.82	6.65
Phoning	7.77	3.29	3.74	4.66
Photo	7.77	2.42	2.99	3.81
Posing	10.56	3.34	3.85	4.84
Purchases	10.28	2.60	3.41	4.97
Sitting	7.37	1.85	2.04	3.34
Sitting Down	9.58	2.50	2.37	4.53
Smoking	6.33	2.90	4.01	3.95
Waiting	7.98	3.37	3.56	4.63
Walking Dog	13.29	4.59	5.33	7.1
Walking	12.76	8.11	9.9	8.78
Walking together	9.95	4.95	6.87	6.59
Average	9.05	3.48	4.09	5.08

Table 2.4: Averaged MPJS over 1000 *ms* for all classes of Human3.6M dataset. Closer to the ground truth is better.

2.5.5 Qualitative Comparison

In this part, we present some examples that illustrate the smoothness of the generated motion with our method compared to the ground truth and the baselines.
In Figure 2.7 we present the 3D pose sequences of a predicted motion using a model trained for long-term prediction with our architecture. We also show the prediction of the same 3D pose sequence by the baseline methods ConvSeq2Seq [87] and FC-GCN [150] using their

publicly available code. LDRGCN [34] is not included as the code for his method is not yet available. We observe that visually our method produces a realistic and smooth motion and that our pose sequence follows more closely the ground truth than the other methods event for long-term prediction. The motion produced by our method does not show any discontinuity, this is the consequence of applying the predicted dynamic of the motion to a starting pose, it prevents the discontinuity that can appear when predicting directly the 3D poses as the other methods do.

2.5.6 Motion Smoothness

In Figure 2.8 we show the evolution of the y coordinate from the skeleton's left foot over time and in Figure 2.9 the evolution of the x-axis of the right hand. The 25 frames samples were selected randomly from the walking and walking together action classes respectively from the Human3.6M dataset. We see clearly in the figure that our method is able to generate a smooth motion in both cases and that we are able to follow the real motion from the ground truth, closely for the walking sample and with a small temporal delay for walking together while for this later, the other methods show a completely different movement.

2.5.7 Computation Time

We show a comparison of the computing time in Table 2.5 of our method with ConvSeq2Seq and FC-GCN. This time comparison is done for long-term prediction (*i.e*, predicting 25 frames) with 8 sequences for each of the 15 action classes from the Human3.6M dataset using the code provided by the author for ConvSeq2Seq and FC-GCN. The results from Table 2.5 show that despite the additional computations required to map the motion back and forth to the tangent space compared to standard GAN architecture, we can predict motion with a speed similar to the other two methods and faster than ConvSeq2Seq.

	total time	time per sample (25 frames)
ConvSeq2Seq	$3.04s$	$\approx 25ms$
FC-GCN	$1.67s$	$\approx 14ms$
Ours	$2.42s$	$\approx 20ms$

Table 2.5: Prediction time comparison for 8 predicted samples per action on Human3.6M.

2.5.8 Distribution Visualization

With Figure 2.10 we further assess the quality of the predicted samples using the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm [104]. We present a 2D visualization of 677 samples of long-term prediction from the CMU MoCap dataset. The resulting representation clearly indicates that the motion from the ground truth and the predicted motions

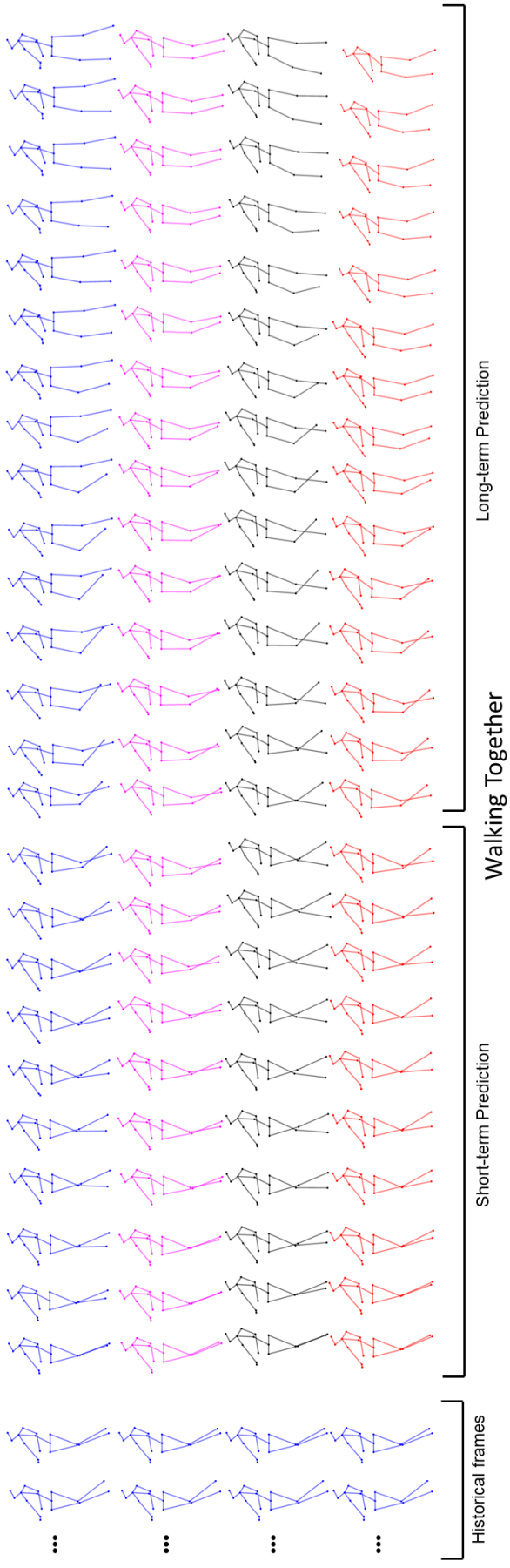


Figure 2.7: **Qualitative results on Human3.6M**. The left frames correspond to the sequence used as a prior. From top to bottom: ground truth, the results of ConvSeq2Seq [87], FC-GCN [150] and our method. The illustrated action corresponds to 'Walking Together' from Human3.6M dataset. Short-term frames shown correspond to predicted frames 1, 9, and 10, and long-term frames to frames 11, 12, 22, 23, 24, and 25.

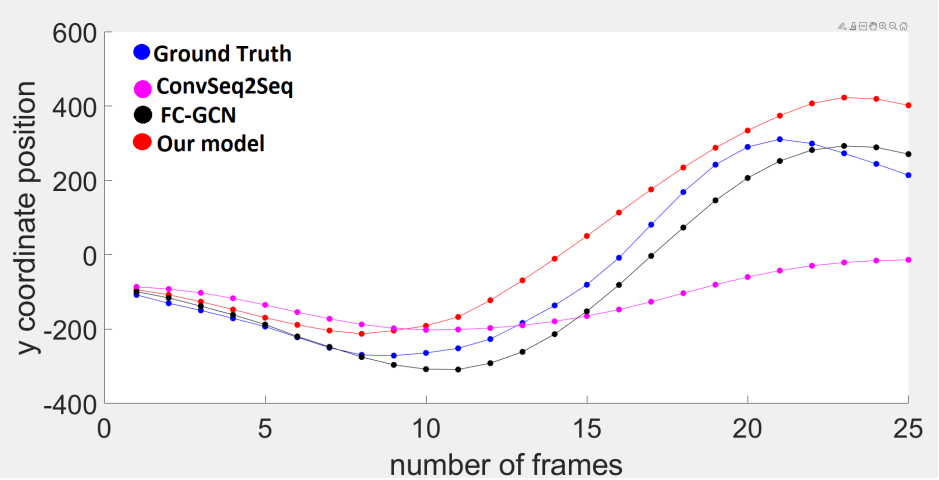


Figure 2.8: **Left foot position evolution, Walking class from Human3.6M**. X-axis and y-axis correspond respectively to frame numbers and joint position on the y-axis.

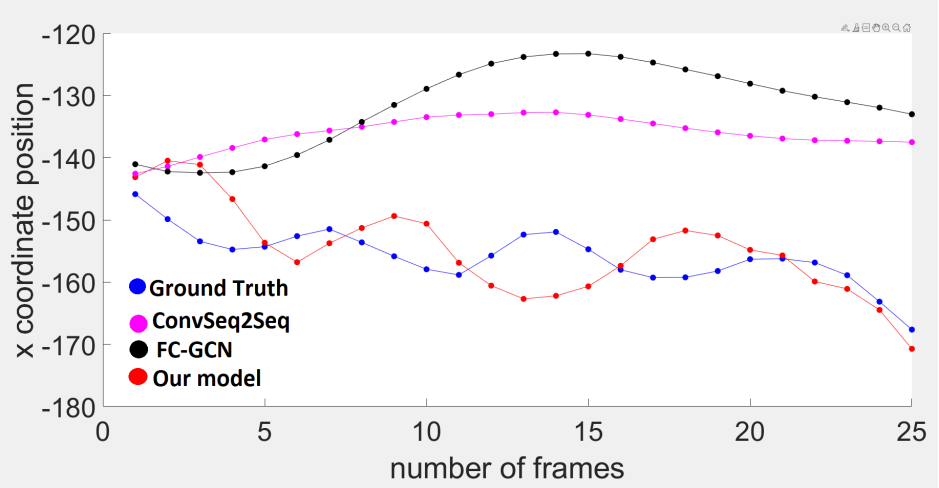
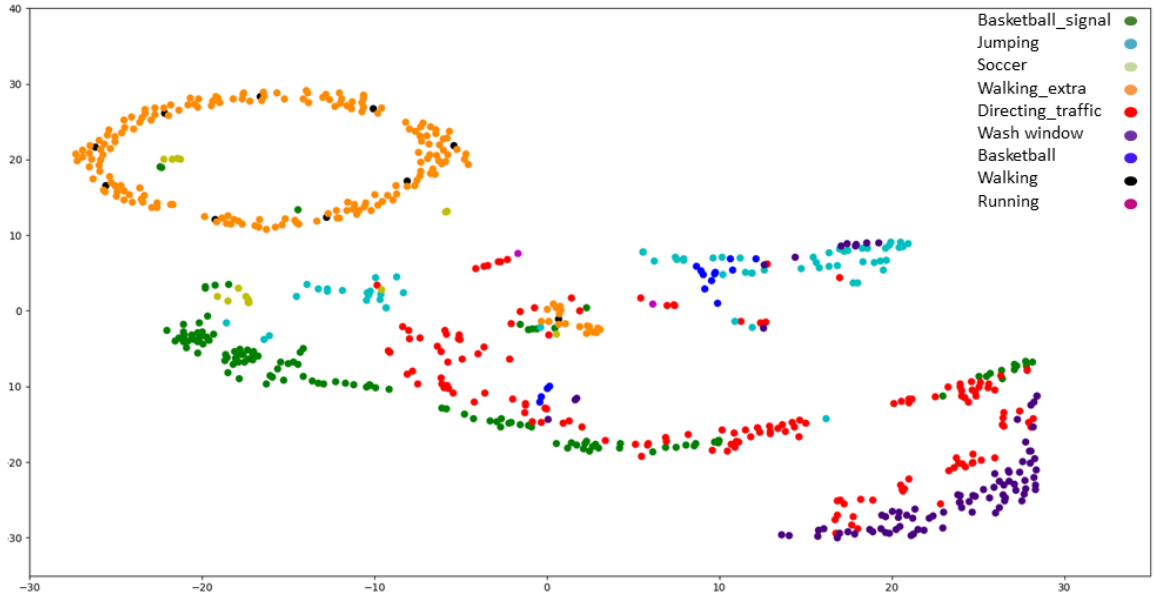
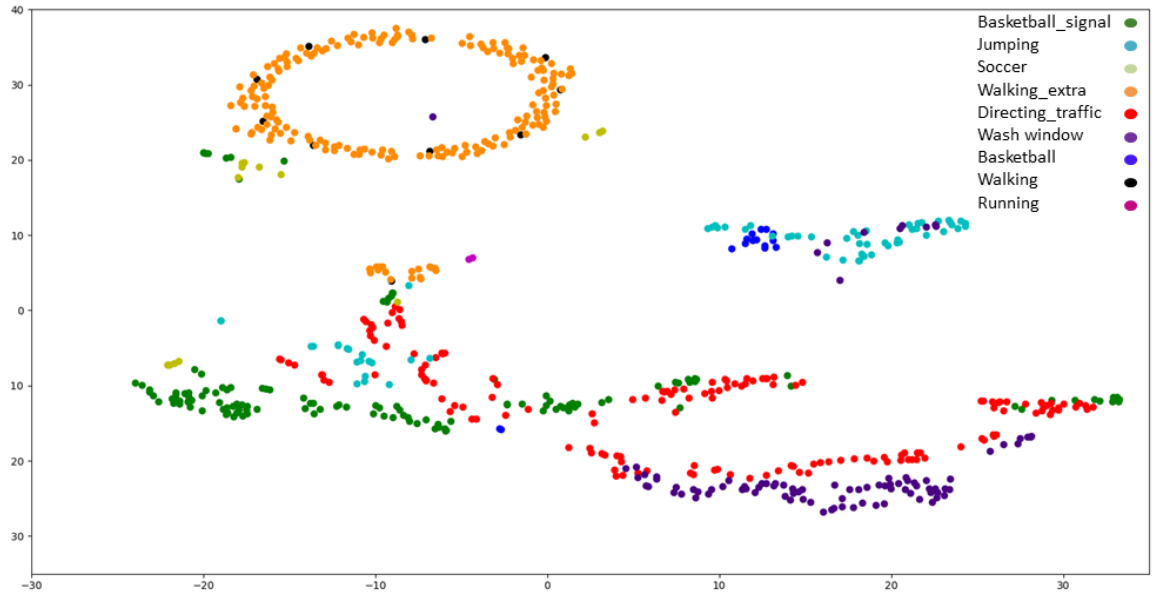


Figure 2.9: **Right hand position evolution. Walking together class from Human3.6M**. X-axis and y-axis correspond respectively to frame numbers and joint position on the x-axis for the right hand joint.

are from very close distributions. Moreover, we can see that the different generated 3D sequences from the same action are relatively distant from each other, meaning for the same action class our model can predict several motions while respecting the prior motion used for the prediction.



(a) Predicted motions



(b) Ground truth motions

Figure 2.10: 2D visualization of the predicted motions by our method and their associated ground truth using the t-SNE algorithm based on Gram distance eq.2.8. Each color represents an action.

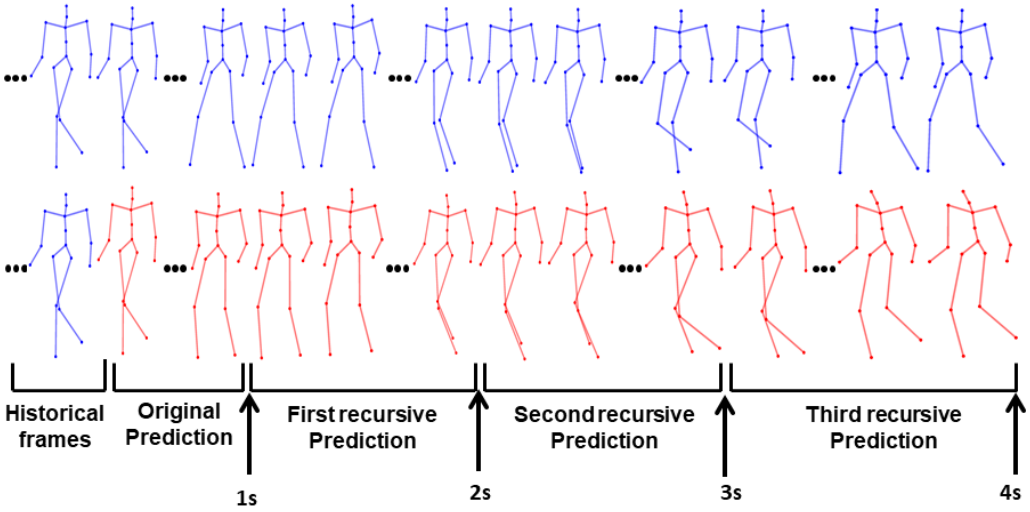


Figure 2.11: Exemple of recursive prediction on a sample of action class walking from the CMU MoCap Dataset for a total of 4 seconds of prediction. On top, the ground truth, on the bottom our prediction.

2.5.9 Recursive Generation

One of the main limitations of our method is its inability to generate sequences of lengths it has not been trained on. We can however still generate longer sequences through recursive generation by predicting subsequent motion based on previous predictions. This recursive generation can be done without specific training simply by modifying the input during testing. However, by feeding our prediction to the network to get further prediction we cause the network to accumulate error over each recursive iteration. In fact, we can not reliably extend the duration of the prediction more than 2 or 3 times. For all types of motion the first and second predictions using predicted data as input are good, the third one is usually still good for periodic motion (*e.g* walking) but not for non-periodic motion (*e.g* greeting). From the fourth prediction onward even the periodic motion will start to deteriorate significantly. Non-periodic motions will usually freeze into a static pose which is to be expected as our prediction can only predict the end of the motion, not infer what other motion might follow. For periodic motions, the deterioration comes from the accumulating error which will cause the skeleton to deform. Still, we are able to generate motion 3 to 4 times longer than what the network was trained on, which allows us to tackle one of the limitations of the method in some ways. We present in Figure 2.11 an example of recursive prediction on the action class walking from the CMU MoCap dataset. The figure shows the original prediction and the three subsequent predictions. This shows that our model can predict motion for longer sequences than it has been trained on as we observe significant differences only during the third recursive prediction, the motion however still follows walking action.

2.5.10 Cross-dataset capabilities

Due to differences in the skeleton formats used by Human3.6M and CMU MoCap, it is not possible to perform a cross-dataset evaluation on these datasets. We have however trained our model on the NTU RGB+D 120 dataset [94] and then predicted motion from data captured with a Kinect camera in real-time. Qualitative results are presented in figure 2.12 (in blue the ground truth and in red the prediction). We show the prediction of a person rubbing their hands. We can see that our method is able to predict on data that does not come from the dataset used for training. While there is a small difference between the ground truth and our prediction we see that we do not reproduce the discontinuities from the Kinect and our prediction has not been influenced by the discontinuities in the prior

2.5.11 Ablation Study

To show the efficiency of the different losses used by our network especially the effect of the combination of the skeleton integrity loss $\mathcal{L}_s$ and the bone length loss $\mathcal{L}_b$, we perform our ablation study using models that were trained using only the mentioned losses. The ablation is performed on the Human3.6M dataset due to the huge quantity of data from the dataset. The ablation results are reported in Table 2.6 for short-term and long-term prediction using the average error of all action classes at different time steps. The results show a clear improvement when adding one of either the skeleton integrity loss or the bone length loss compared to using only $\mathcal{L}_a$ and $\mathcal{L}_r$. Furthermore using both $\mathcal{L}_s$ and $\mathcal{L}_b$ improve significantly the results for long-term prediction while keeping a similar accuracy for short-term prediction with regard to using only $\mathcal{L}_s$ or only $\mathcal{L}_b$. This evidences the importance of using both losses when doing long-term prediction, it allows the model to capture the spatial dependencies between joints and to be able to predict plausible poses even for longer-term horizons.
We show in Figure 2.13 the effect of the losses on the visual quality of the prediction. We notice that excluding $\mathcal{L}_s$ and $\mathcal{L}_b$ leads to important deformations in the upper body but the produced legs motion is rather coherent. Adding $\mathcal{L}_s$ helps produce a motion closer to the ground truth, we however still see noticeable bone deformations even if we are able to keep a coherent skeleton. Using only $\mathcal{L}_b$ leads to a skeleton without any deformation even during long-time prediction but also to very little motion being produced. Using both losses allows us to keep the best skeleton coherency while producing a motion that is close to that of the ground truth. We show in table 2.7 the MPJPE values for different input sequence lengths for long-term prediction on Human3.6M. We report the values for sequences of 25 frames (default value used for comparison with the state-of-the-art), 15 frames, 10 frames, and 5 frames. We see that a shorter prior leads to a decrease in performance for both short and long-term prediction but this decrease is less important for long-term prediction (except for the 5 frames prior) highlighting the ability of our network to generate accurate predictions for long-term motions. We observe that we can use priors of 10 or 15 frame with a moderate drop in performance but with only 5 frames the drop increase significantly, especially for long-term prediction.

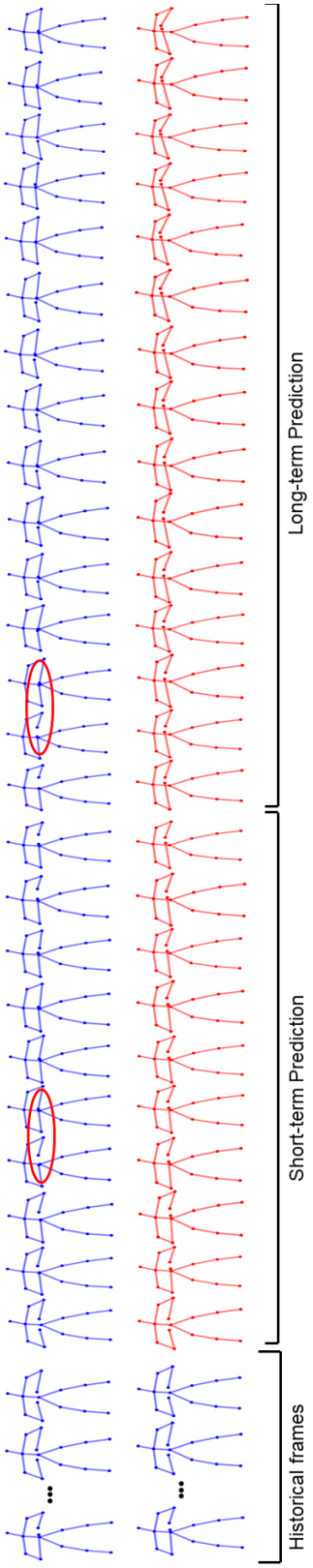


Figure 2.12: Cross database results from NTU RGB+D to Kinect real-time capture. In blue the ground truth and in red the prediction. The discontinuities from the Kinect camera are highlighted in red

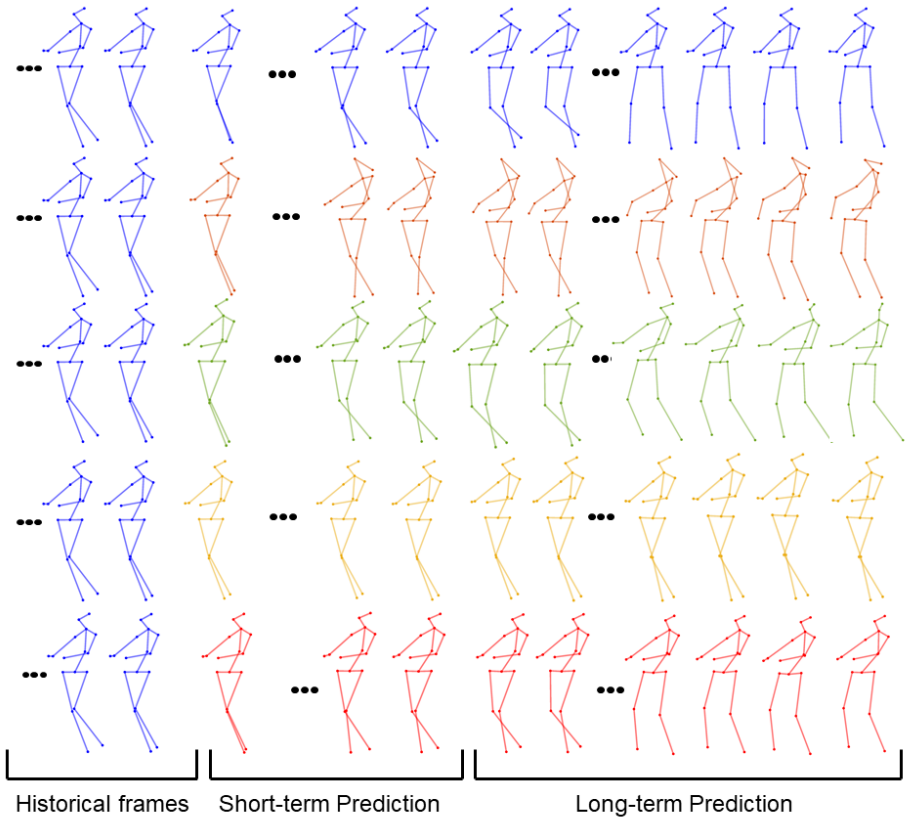


Figure 2.13: Impact of the bone length loss and the skeleton integrity loss on prediction quality on a sample from action class Walking together from Human3.6M. From top to bottom: the ground truth, neither $\mathcal{L}_s$ nor $\mathcal{L}_b$, only $\mathcal{L}_s$, only $\mathcal{L}_b$ and both $\mathcal{L}_s$ and $\mathcal{L}_b$

loss functions	80	160	320	400	1000
$\mathcal{L}_a + \mathcal{L}_r$	20.2	34.9	62.4	74.9	133.3
$\mathcal{L}_a + \mathcal{L}_r + \mathcal{L}_s$	13.6	23.4	42.6	51.6	103.8
$\mathcal{L}_a + \mathcal{L}_r + \mathcal{L}_b$	12.6	22.4	**41.3**	**49.9**	105.6
$\mathcal{L}_a + \mathcal{L}_r + \mathcal{L}_s + \mathcal{L}_b$	**12.3**	**22.2**	**41.3**	50.1	**96.2**

Table 2.6: Impact of the bone length loss and the skeleton integrity loss on the prediction performance for short-term and long-term.

loss functions	80	160	320	400	1000
5 frames prior	14.2	25.3	47.0	56.5	104.4
10 frames prior	13.6	24.2	44.7	53.5	98.6
15 frames prior	13.3	23.6	43.7	52.4	96.8
25 frames prior	**12.3**	**22.2**	**41.3**	**50.1**	**96.2**

Table 2.7: Impact of the prior length on long term prediction

2.6 Conclusion and Limitations

In this chapter, we presented a new and robust method to deal with human motion prediction. In our method we represent the temporal evolution of 3D human poses as trajectories, these trajectories can be mapped to points on a hypersphere. To be able to learn this manifold-valued representation we use a manifold-aware Wasserstein GAN that can capture both the spatial and temporal dependencies involved in human motion. Through extensive experiments, we prove the robustness of our method for long-term motion prediction when compared to recent literature. With our qualitative results, we confirm that we are able to predict plausible poses and smooth motions in long-term horizons.
The two main limitations of the proposed method are the following: the fixed length of sequences and the inability to deal with sudden changes in motion. The fixed length in motion is a consequence of the GAN architecture where the input and output sizes a fixed. We demonstrate in our experiment that we can deal with this problem by using recursive generation, which shows the ability of the model to generate up to 4s motion for some classes when trained on 1s sequences. The inability to deal with a sudden change of motion is inherent to the way motion prediction is usually approached. Indeed, we only consider the historical motion as a condition to predict the motion but it is not always enough to get an accurate prediction. Things like the environment, the goal of the motion, and the motion of other persons can influence the future. Taking some of these modalities into account would surely allow for longer and more accurate predictions.

Part III

Human Reaction generation with Transformer Network

CHAPTER 3

HUMAN REACTION GENERATION WITH TRANSFORMER NETWORK

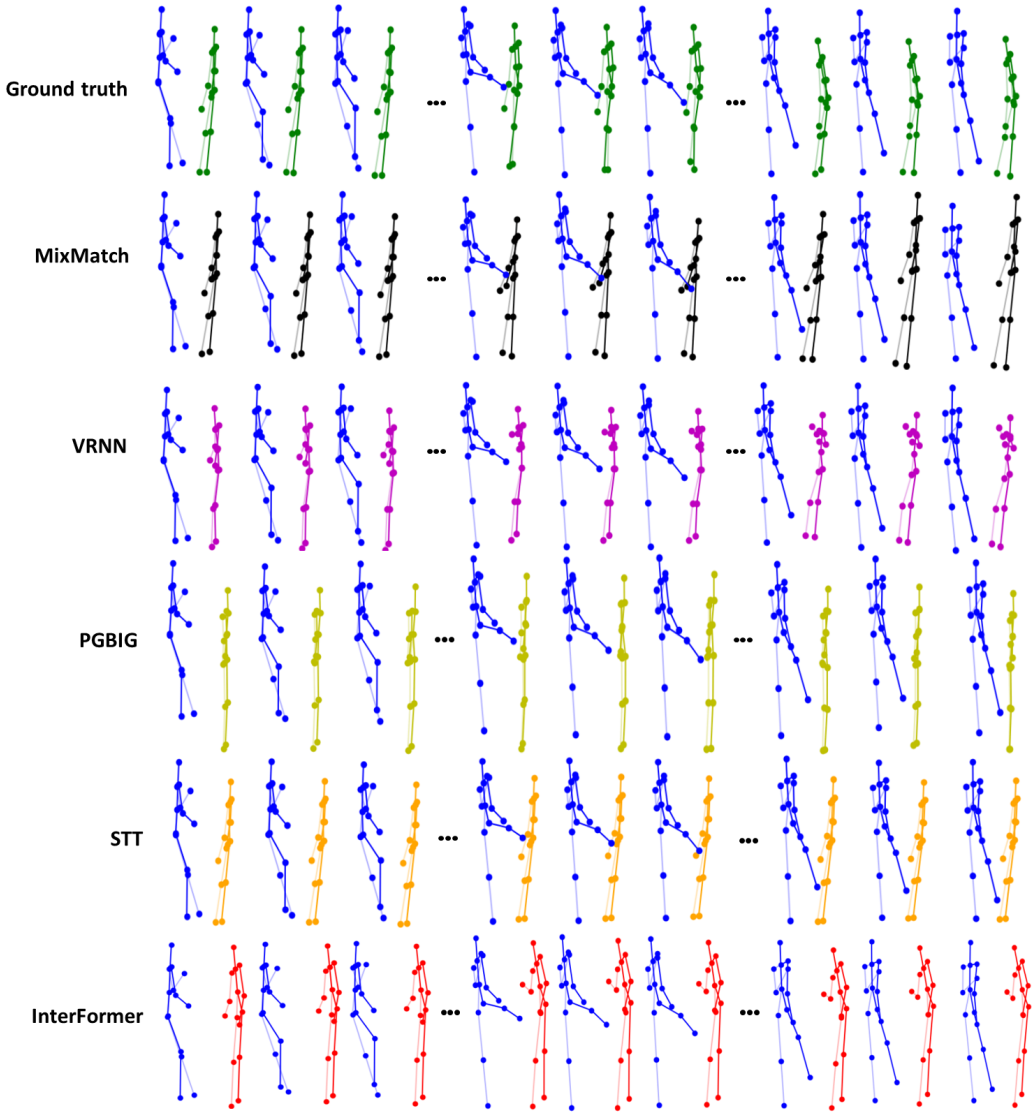


Figure 3.1: **Example of reaction generation.** In blue the action motion is used as a condition. In other colors, the reaction is either from the ground truth or generated by the different models. Example from the kicking class of the SBU dataset. Our model generates a more realistic motion than the competing approaches.

3.1 Introduction

Modeling the dynamics of human motion is at the core of many applications in computer vision and robotics. Most works on human motion generation ignore human interactions and focus rather on the generation of actions of a single person. In addition, only a few works investigating human interaction generation [11] look at the reaction generation problem. What makes human reaction generation a challenging problem is a nonlinearity in the temporal evolution of human motion and the two sources that condition the motion: the action and its corresponding reaction. The first issue arises because human motion is generally performed at varying evolution rates. In other words, a person performing the same activity will go

roughly through the same stages but at slightly different rates every time. In addition, as stated by [83], unlike simple actions such as walking or running, complex human interactions such as duet dancing generate highly complex pose sequences operating close to the limit of human kinematics with very low periodicity. The second issue arises because the same action can have a different reaction depending on the interaction context, e.g., when reacting to a punch depending on the position, one can react more or less strongly. These two issues make the generation and the evaluation of reaction challenging. Several questions arise as we try to tackle this challenge. How to translate action to reaction? How to model the long-term sequence? How to represent a complex action-reaction sequence?

Our goal is to learn the reaction from a training sequence of actions and reactions by using Transformer architectures. The breakthroughs from Transformer networks in Natural Language Processing (NLP) domain have sparked great interest in computer vision. Transformer architectures are based on a self-attention mechanism that learns the relationships between elements of a sequence. Unlike recurrent networks that process the elements of the sequence recursively, Transformers can attend to complete sequences and thereby are able to learn spatial and temporal relationships making them a good candidate for modeling human motion. In this chapter, we propose InterFormer, which with its spatial and temporal attention modules, is able not only to model the spatial and temporal dependencies in the action and in the reaction but also in the interaction between the two humans providing a solution to the two previously mentioned issues. Figure 3.6 (Left) shows how our InterFormer can generate a proper reaction sequence (red skeleton) by taking as input an action sequence (blue skeletons) and the initial position of the reaction sequence. Green circles highlight the reaction parts of the motion: the head goes backward in reaction to the punch; the hand is raised as the body continues to move backward to keep its balance. Figure 3.1 shows a generated reaction from the "kicking" class of the SBU dataset. Our method is able to generate a proper motion. The code and some video examples of motions generated by our method are available at https://github.com/CRISTAL-3DSAM/InterFormer

Our major contributions are as follows:

- We propose a novel Interaction Transformer framework for the challenging human reaction generation task. To the best of our knowledge, this is the first work that challenges the task of human reaction prediction given the action of the interacting skeleton using a Transformer based architecture.
- We formulate the reaction generation problem as a translation problem, where we translate a given action of a skeleton to its corresponding reaction such that the entire interaction looks coherent and natural.
- We adopt a graph representation for self-attention to better exploit the skeleton structure while we ignore this representation for computing the attention between the two interacting skeletons. In this case, instead of a graph representation, we exploit the distance between the interacting joints assuming that closer joints involve stronger interaction. By introducing this distance, we provide the prior knowledge that helps to model the interaction.
- While the previous methods for interaction generation address limited and simple

short-term interactions, we evaluate our method on the DuetDance dataset that provides more complex and long-term interactions.

3.2 Related Work

Human Action Generation. Human action recognition and prediction from 3D skeletons is a popular topic [71, 109, 34, 37, 46]. Inspired by the recent advances in generative models, several works [167, 149, 169, 90] proposed human action generation models in order to generate a consecutive sequence of human motions. Recently there has been an increase in motion generation based on different modalities, [159] use control signals such as the global trajectory of the person to generate human motion in long-term horizons while [1] and [56] generate motion based on speech audio. Meanwhile, others use only knowledge of the past motion which allows them to work in real-time but on shorter motion [109, 34, 134]. However, these works only focus on the generation of individual actions. More recently, interaction prediction and generation have also been addressed [11, 83]. For instance, [11] use a multimodal variational recurrent neural networks to predict the future motion of both participants in an interaction based on pasts sequences of motion. More recently, a lot of focus has been devoted to human pose and motion generation from text or action labels, as well as its reciprocal task [54, 98]. However, our approach proposes to generate and predict human motion reactions from an action. In addition, these papers focus only on one person, while our approach is dedicated to the generation of reactions in two-person interaction. To complement the existing dataset with interactions [94, 163], different types of complex interaction datasets have also emerged like [85] and their collection of conversational hand motions or triadic interactions [68]. However the number of human interaction databases with 3D skeletons and large enough to be used for motion generation is low. Table 3.1 shows a list of existing 3D skeleton datasets containing interaction motions. We can see that most of these datasets contain few classes and few samples and that the most common capture method is a depth camera. Datasets captured with depth cameras suffer a lot from the occlusion that can happen in some interactions classes such as "hugging" up to the point where these classes become hard to use. It should be noted that some datasets with a low number of sequences such as UoL3DSi, Uol3DAi, G3Di, and CMU panoptic consist of long sequences (up to 8 min for CMU panoptic) in which several motions are performed or repeated. The 3D skeletons from DuetDance are obtained from RGB videos using LCR-Net++[128]. Some works also look at human reactions with other modalities such as walking trajectories [55] or conversational data [2, 157, 160]. However, there are very few works on human reaction generation. In this thesis, we focus on this and propose InterFormer, a novel Transformer architecture. This idea has not been investigated by any other existing work.

Graph Representation has been widely used for 3D classification and segmentation [121, 120], visual question answering [111], human interaction recognition [171, 119]. For instance, [171] proposed a Dyadic Relational Graph Convolutional Network (DR-GCN) for skeleton-based interaction recognition. When dealing with 3D skeletons, it is natural to use a graph representation as the graph of the skeleton exists physically in the form of segments linking joints. While most works use the graph representation of the skeleton directly as an input, doing so when dealing with interaction leads to losing information. We propose to

dataset	classes	samples	persons	method
CMU panoptic[67]	4	54	2-7	depth camera
NTU-RGB+D 120 [94]	26	24808	2	depth camera
SBU [163]	8	269	2	depth camera
K3HI[61]	8	312	2	depth camera
CMU MoCap[1]	36	55	2	motion capture
UoL3DSi[31]	8	80	2	depth camera
Uol3DAi[32]	8	20	2	depth camera
PKU MMD[30]	10	200	2	depth camera
M2I[154]	9	720	2	depth camera
LindyHop600K[74]	1	9	2	depth camera
G3Di[17]	4	24	2	depth camera
DuetDance[83]	5	416	2	from RGB

Table 3.1: List of 3D skeletons motion databases with interactions. Only the classes containing interaction are counted and only the samples from these classes are considered.

use the graph as part of the attention module to take advantage of the graph representation without losing the information about the interaction. Experiments show the effectiveness of the proposed InterFormer over existing methods.

Transformer is a network architecture first introduced in [146] to deal with natural language processing (NLP) translation tasks. Its impressive results on this task and easily adaptable architecture led to great interest in several fields including NLP tasks such as question answering, Natural Language Inference [38, 19] but also protein modeling [18] and many others, among which computer vision. Transformer architecture has been extensively used in computer vision for many tasks including object detection [22, 23], image generation [126, 115], depth estimation [127], image classification [125], image synthesis [43], action recognition [119]. Closer to our problem, works have used Transformer to generate human motion: [118, 153] generate human motions based on the class labels while [3] use them to predict future motion based on a historical sequence. Different from these methods, we use a Transformer architecture with temporal and spatial attention for solving the reaction generation task. Generating a reaction responding to an action can be seen as a translation problem: translating from a language "action" to a language "reaction". The performance of the Transformer on natural language translation tasks and its use of temporal information are a good fit for our task of reaction generation. By adding spatial attention and graph information, we can produce a realistic reaction to an action. To the best of our knowledge, InterFormer is the first Transformer architecture used to solve the problem of human reaction generation. Transformer uses an encoder-decoder architecture shown in Fig.3.2 making extensive use of the attention mechanism. Attention can be explained as a mapping of a query, a set of key-value pairs, and an output. The output is a weighted sum of the values, where the weights are computed by a compatibility function of the query with the corresponding key. The goal of attention is to find correlations between two sequences. The scaled dot product attention is calculated following:

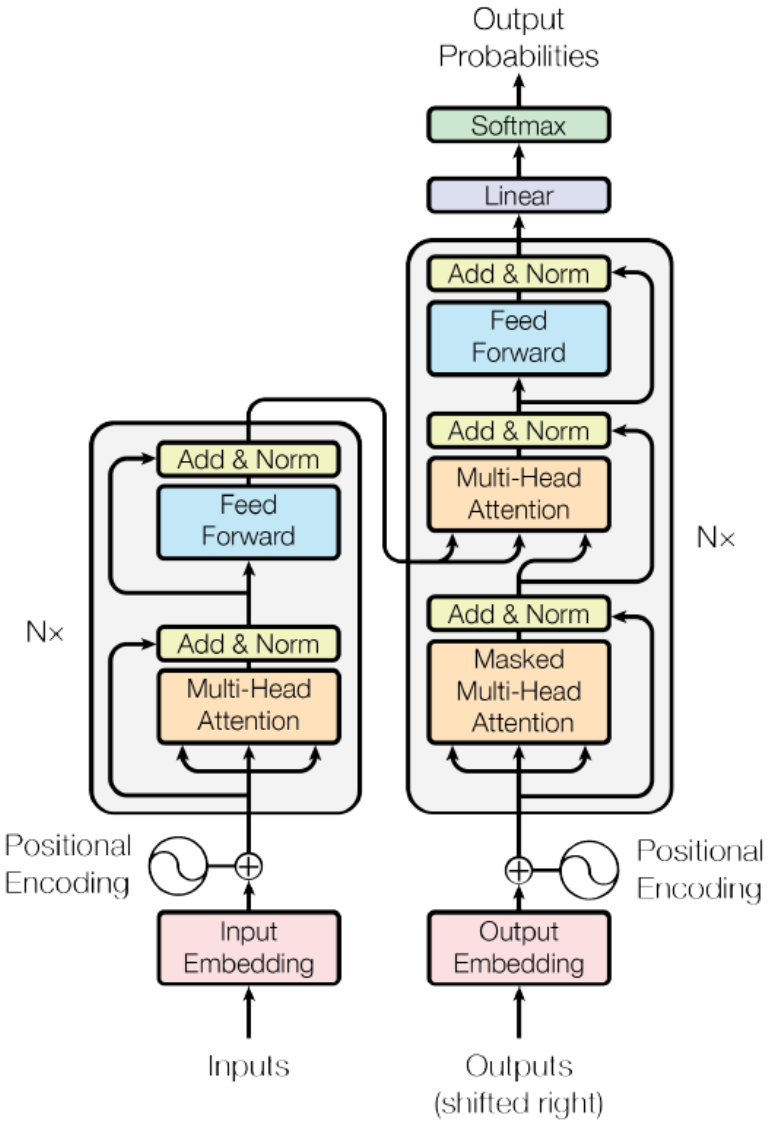


Figure 3.2: The Transformer model architecture [146]

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{3.1}$$

With Q, K, V the query, key and values matrix respectively and d_k the dimension of the keys. In Fig.3.2 we see that in the decoder (right part) there are two attention layers. These two layers differ only by their inputs. For the first layer, all the inputs come from the decoder while for the second layer, two inputs come from the encoder. They represent two types of attention, self-attention, and cross-attention. Cross-attention maps the data from the encoder to the data of the decoder. In the NLP translation task from English to French, it corresponds to mapping French words to English words. For cross-attention K and V come from the encoder and represent a sentence in English while Q comes from the decoder and represents

a sentence in French. Fig.3.3 shows an example of cross-attention for NLP translation. Self-attention on the other hand only takes data from the decoder. It tries to map the french sentence with itself, finding correlations between the words inside the sentence. Fig.3.5 show an example of self-attention for NLP, we see that "it" is strongly related to "The" and "animal". Self-attention helps Transformer learn the structure of the sentence. With self-attention Q, K, and V all come from the decoder and represent the French sentence. The attention layer in the encoder is also self-attention but works on the English sentence instead.

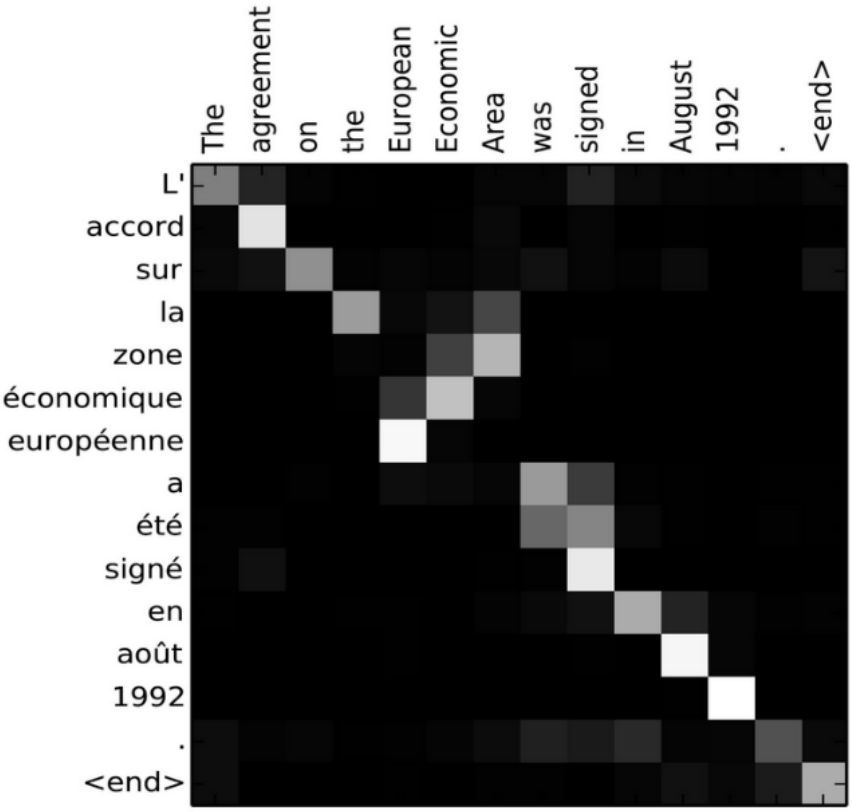


Figure 3.3: Figure from[8]. Example of cross attention between a French sentence and an English sentence. Lighter squares indicate stronger correlations.

The implementation of the attention used in [146] is presented in Fig.3.4 as well as the multihead attention architecture. Multihead attention consists in computing attention several times with the inputs linearly projected with different projection weights. The results of these attention calculations are then concatenated to obtain the attention layer output. The multihead attention can be described as:

$$\begin{aligned} \text{Multihead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}(head_1, ..., head_n)W^S \\ with\,\text{head}_\text{i} &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{3.2}$$

where n is the number of heads, W_i^Q, W_i^K, W_i^V, and W_i^S are the projection weights for Q, K, V, and the output respectively.

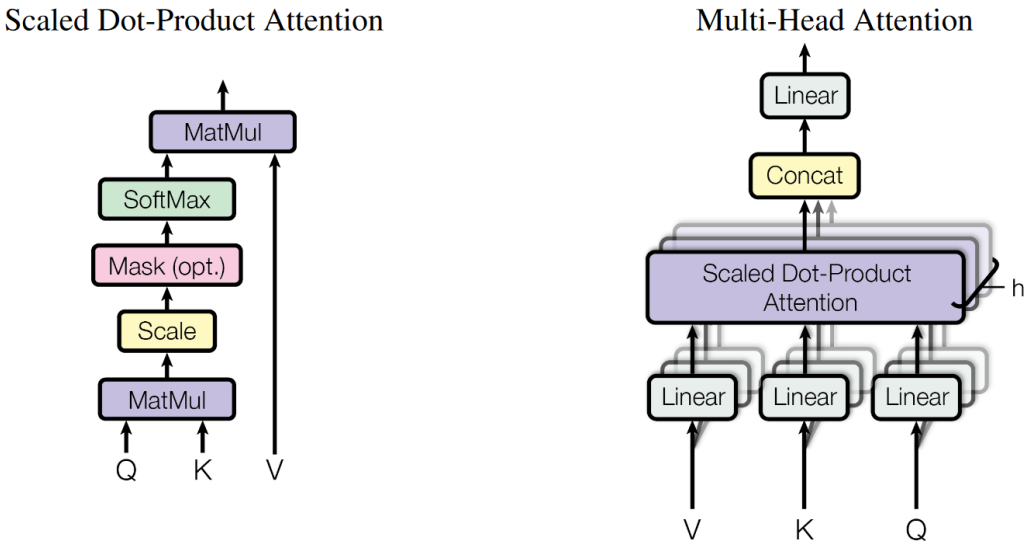


Figure 3.4: **Left:** scaled dot product attention. **Right:** Multiheads attention, several attention layers running in parallel [146]

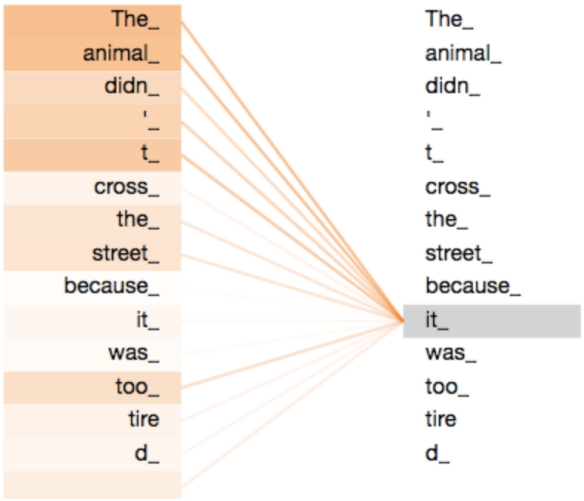


Figure 3.5: Illustration of the attention when encoding the word "it" the darker colors show that the attention makes the link with "The animal".
source: https://jalammar.github.io/illustrated-transformer/

The encoder starts with an embedding layer that converts the text into vectors of fixed size. Then we use positional encoding, a method that injects information about the temporal position of each element in the sequence. In [146] the positional encoding is defined as:

$$\begin{aligned} PE_{pos,2i} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{pos,2i+1} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{3.3}$$

with pos the position and i the dimension and d_{model} the dimension of the input. Then we go through several layers with the same architecture: multihead attention followed by residual connection, which prevents the loss of information, and normalization then a feed-forward network. The feed-forward network is a simple network containing several layers of neurons

and where information always moves in a single direction. Then there is another residual connection and normalization. The decoder is similar to the encoder for the beginning: embedding, positional encoding then several repeating layers. These layers are also composed of multihead attention followed by residual connection and normalization then another multihead attention operation but the key K and the value V correspond to the output of the encoder. Then this is followed by residual a connection and normalization, a feed-forward network, and another residual and normalization. Finally, the output of these layers goes through a linear layer, that performs a linear transformation of data. Since Fig.3.2 represents a Transformer for NLP there is a final softmax function to obtain probabilities. They are used to decide the next word in the sequence outputted by the network. The first multihead attention of the decoder is different from that of the encoder in that the attention is temporally masked. This means that when training, at time t we hide the embedding from time $t+1$ onward. This is done so the network does not learn to rely on data it is not supposed to know.

3.3 The Proposed Interaction Transformer

Let us consider P_t the positions of k distinct joints at time t. Consequently, an action sequence P of T frames, can be described as a sequence $P{=}\{P_1, P_2, \ldots, P_T\}$, where $P_t{\in}\mathbb{R}^d$ and $d{=}3{\times}k$, where $P_t{=}[J_1(t), \ldots, J_k(t)]$, with k the number of joints in the skeleton, and $J_i(t){=}[x_i(t), y_i(t), z_i(t)]$ the 3D coordinates of joint i. The goal is to generate a reaction $Y{=}\{Y_1, Y_2, \ldots, Y_T\}$ a sequence of skeleton poses from $X{=}\{X_1, X_2, \ldots, X_T\}$ a sequence representing the action motion.

Our overall architecture of InterFormer is illustrated in Figure 3.6 and consists of four modules: a motion encoder, a motion decoder, a skeleton adjacency module, and an interaction distance module. The motion encoder encodes the motion of the skeleton using self spatial skeleton attention and self temporal motion attention. Both aim to find the important spatial and temporal relations within the input action motion to transmit them to the decoder. The motion decoder generates the reaction motion using the encoding from the motion encoder and consists of self spatial skeleton attention, self temporal motion attention, interaction spatial skeleton attention, and interaction temporal motion attention. Moreover, the skeleton adjacency and interaction distance modules help the different spatial attentions to focus on the most important parts of the skeletons and of the interaction.

3.3.1 Motion Encoder

The motion encoder takes as input an action sequence X to which we add positional encoding defined by [146]. This positional encoding encodes temporal information of each frame in the sequence. Inspired by [146] we use temporal attention to capture the temporal relationships within the motion of the skeleton. However, the motion contains both temporal and spatial information. Thus, we add a spatial attention module to complement the temporal attention to help find the spatial dependencies within the skeleton.

Self Spatial Skeleton Attention. For our self spatial skeleton attention module, we consider

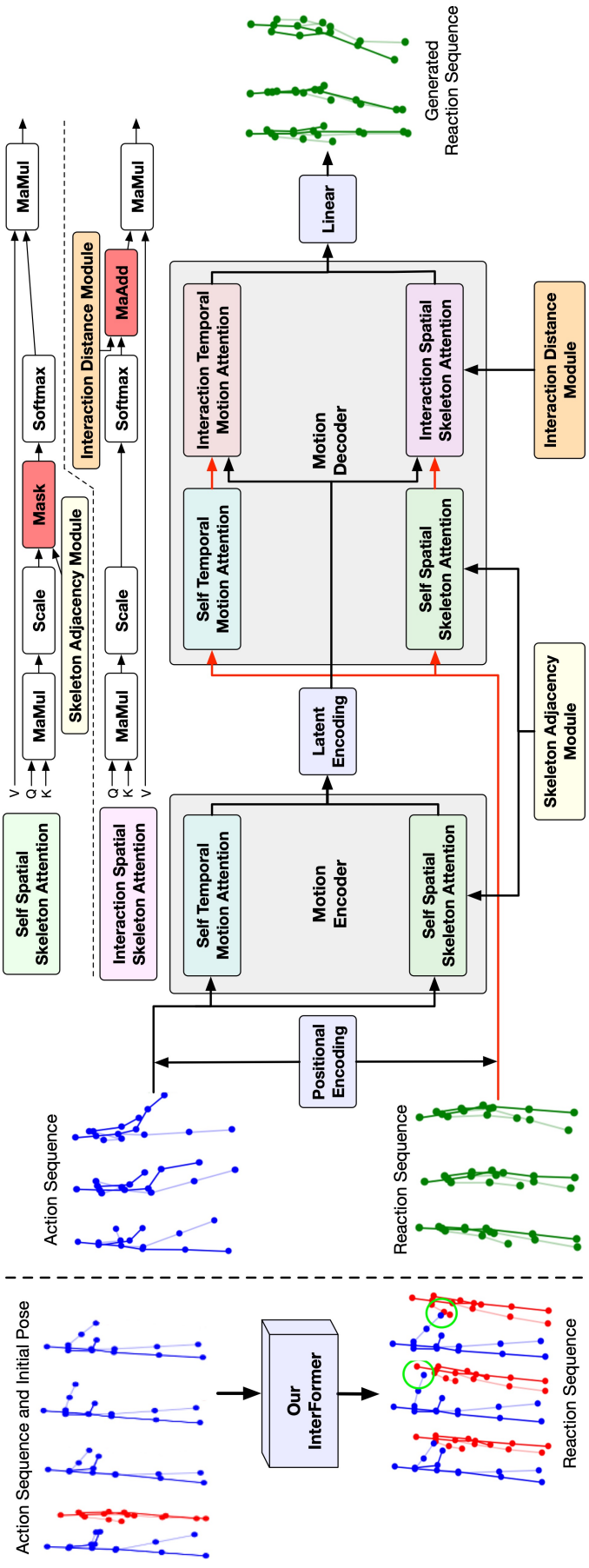


Figure 3.6: **Left:** InterFormer during testing: given an action sequence (blue) and the first frame of a reaction sequence (red), we generate the full reaction sequence. We predict one frame at a time based on the previously generated frames. **Right:** Overview of InterFormer during training. The motion encoder takes an action sequence as the input and outputs a latent encoding. The motion decoder takes as inputs the reaction sequence corresponding to the action sequence and the latent encoding from the encoder. The motion decoder outputs a generated reaction sequence. Both the encoder and decoder contain several attention modules. **Top Right:** The skeleton adjacency and interaction distance modules interact directly with spatial attention.

each frame independently and look at the relation between the position of each joint. We use the scaled dot-product attention from [146]:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{dim}}\right)\mathbf{V}, \tag{3.4}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value matrices of sizes $dim \times |P_t|$ which contain a set of queries, keys, and values (one for each joint in the skeleton for a given frame) of sizes dim which is for spatial attention $|J_1(t)|$. These queries q_i, keys k_i, and values v_i are obtained by multiplying an input a_i, b_i, and c_i by weight matrices $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$ of size $dim \times dim$:

$$q_i = a_i\mathbf{W}_q, \qquad k_i = b_i\mathbf{W}_k, \qquad v_i = c_i\mathbf{W}_v. \tag{3.5}$$

For self-attention $a_i{=}b_i{=}c_i$ and for spatial attention they represent the 3D coordinates of joint i at a given time, either directly or through the value corresponding to the coordinates after going through the previous attention layers. We use the multihead version of the attention [146] where the inputs are split into smaller parts according to the input size of each head. Each part is treated by its own attention module and the outputs of these modules are concatenated. For spatial attention, we fix the number of heads at $|J_1(t)|$, one for each dimension of the 3D coordinates.

Self Temporal Motion Attention. For the self temporal motion attention, we consider the entire skeleton and observe the motion of its joints over time, i.e., we try to find the links between the position of the joints from one frame to another. This is performed in the same way as for self spatial skeleton attention by using Eq. (3.4) and Eq. (3.5). However, here $a_i{=}b_i{=}c_i$ represent the entire skeleton at time $t{=}i$, $dim{=}d$ and Q, K, and V are of size $dim \times T$. We also use the multihead version of the attention, but here the number of heads can be set as a hyperparameter.

3.3.2 Motion Decoder

The decoder receives the encoder's output Z as well as the reaction sequence Y. It is composed of four attention modules as illustrated in Figure 3.6. The self-attention modules work in the same way as the encoder but take Y to which we add the positional encoding as input.
Interaction Spatial Skeleton Attention. The interaction spatial skeleton attention module looks at the relations between the joints of the interacting skeletons at a given frame. The attention is also computed using Eq. (3.4) and Eq.(3.5) but here the query matrix Q comes from the reaction sequence Y and the key and value matrices K and V come from the encoder output Z.

Interaction Temporal Motion Attention. The interaction temporal motion attention module looks at the relations between the frames from the action sequence and the frames from the reaction sequence. Discovering these relations enable the synchrony of the generated reaction. Likewise, the query matrix Q comes from the reaction sequence Y but the key and value matrices K and V come from the encoder output Z.

In both the encoder and decoder, before each attention module, the input is normalized, and after each module, the output is also normalized and added to a residual connection of the non-normalized module input like in [146]. The spatial and temporal attentions are computed in parallel and are added after passing through all modules. This final output then goes through a feed-forward layer and is added to the residual connection. The architectures described here for the encoder and the decoder correspond to a single layer of the encoder and one single layer of the decoder. There are $N{=}6$ of each of these layers, and the input of layer h is the output of layer $h{-}1$. Finally, after the last decoder layer, the output goes through a final linear layer to get the reaction sequence.

3.3.3 Skeleton Adjacency and Interaction Distance

Recently many works using skeletons also use a graph representation which was proved to be a particularly efficient representation for action recognition [171, 119]. Building a graph for a skeleton is particularly intuitive in that the joints of the skeletons are already linked together by body segments. However, in our case, using a graph representation might be ill-fitted. Indeed while graphs provide information about the skeleton structure and help us concentrate on the most interesting parts of the skeleton, they would limit us when modeling the interaction. The information we have about the interaction is contained in the attention between the encoder and the decoder and the relations in the skeleton graph are very different from the relations between the joints of the two skeletons (all relations are possible). However, graphs can still provide important information that we can use to improve our generation.

Skeleton Adjacency Module. We can use the information contained in the graph representation by looking at the adjacency matrices of the joints. We use three adjacency matrices that we combine to create a mask. The three matrices are based on the ones used by [119]: (i) the identity matrix $\mathbf{I}$ used to represent the joints themselves; (ii) the matrix of inward relations $\mathbf{In}$ which are the paths from the extremities (head, hands, and feet) to the root joint (torso or pelvis), and (iii) the matrix of outward relations $\mathbf{Out}$ which represents the paths from the root joint to the extremities. The three matrices of sizes $|P_t|{\times}|P_t|$ are then added to get the mask matrix $\mathbf{M}{=}\mathbf{I}{+}\mathbf{In}{+}\mathbf{Out}$ that we apply to the attention matrix $\mathbf{Att}$ of size $|P_t|{\times}|P_t|$ to hide values that are not part of the graph as illustrated in the top right part of Figure 3.6.

$$\mathbf{Att}_{i,j} = \begin{cases} \mathbf{Att}_{i,j}, & \text{if } \mathbf{M}_{i,j} \neq 0 \\ 0, & \text{if } \mathbf{M}_{i,j} = 0 \end{cases} \tag{3.6}$$

Interaction Distance Module. Interaction attention, which is also the attention between the encoder and the decoder, can also use a graph representation [171], but this graph cannot be fixed since the interesting links between joints vary from class to class e.g., for “punching" we are interested in the link between the hand and the head but not for “kicking". Ultimately, it is the spatial attention between the encoder and decoder that discovers the important links between the two skeletons. However, as suggested by [171] we can add prior knowledge to the attention to help us model the interaction for some classes. This information is the distance between the joints of both skeletons, i.e., joints that are close to each other are more

likely to interact than those that are far away:

$$\mathbf{Dist}_{i,j} = -\|J^i_{action}(t) - J^j_{reaction}(t)\|_2, \tag{3.7}$$

where $J^i_{action}(t)$ and $J^j_{reaction}(t)$ are the joint i and j of the action and reaction skeletons at time t, $\mathbf{Dist}$ is a matrix of size $|P_t|\times|P_t|$. Unlike the graph for self spatial attention, we do not use the distance matrix to create a mask because some of the relations between the two skeletons are not defined by the distance between the joints (e.g., waving and waving back), thus using the distance matrix as a mask would prevent such relations from being discovered. We add $\mathrm{softmax}(\mathbf{Dist})$ to the attention matrix to keep all the information that interests us, as illustrated in the top right part of Figure 3.6. By using the softmax function on the distance matrix, we add values of the same order to the attention matrix while making shorter distances more important.

3.3.4 Objective Optimization

We use two loss functions to direct our model. The first one is the sequence loss (L_s) which compares the generated sequence with the corresponding ground truth using the Mean Square Error (MSE) :

$$L_s = \frac{1}{T}\frac{1}{k}\sum_{t=1}^{T}\sum_{i=1}^{k}(J_i(t) - \hat{J}_i(t))^2, \tag{3.8}$$

where $J_i(t)$ is the position of the real joint i at time t and $\hat{J}_i(t)$ the position of the generated joint i at time t. The second is the first frame loss (L_{ff}) used to add constraints on the first two frames by ensuring that the motion between the two is realistic and limits the discontinuities that can happen at the beginning of the sequences. This loss is necessary as otherwise the model sometimes ignores the initial input frame and generates a sequence based on its own inferred initial position. For this loss, we also use the MSE but on the difference between the two first frames:

$$L_{ff} = \frac{1}{k}\sum_{i=1}^{k}((J_i(2) - J_i(1)) - (\hat{J}_i(2) - \hat{J}_i(1)))^2. \tag{3.9}$$

3.3.5 Implementation Details

We train our InterFormer using Torch 1.8.1 on a PC with two 2.3Ghz processors, 64G RAM, and an Nvidia Quadro RTX 6000 GPU. We use the Adam optimizer [77] with α=0.0001, β_1=0.9, β_2=0.98, and ϵ=1×10^{-9}. The batch sizes are set to 128 for SBU and DuetDance and 64 for K3HI. InterFormer works even if we do not provide the original position of the reaction sequence (the first frame of the sequence) as input, but this can cause the generator to produce a skeleton very far from its actual location, which will lead to a bad generation. To solve this during testing, we give as input to the decoder the first frame of the sequence which gives information about the original location of the skeleton. During testing, we generate sequences of variable lengths depending on the length of the input action motion.

The sequences are generated in an auto-regressive manner and the model generates an end-of-sequence value to indicate the end of the motion generation. If the motion is generated correctly, then this value will correspond to the end of the input action sequence.

3.4 Experiments

We conducted comprehensive experiments to evaluate our proposed approach by comparing state-of-the-art models on three datasets. We also visualize the ability of action-reaction generation. Finally, we perform ablation studies to evaluate the effectiveness of using spatial attention and our skeleton adjacency and interaction distance modules.

3.4.1 Datasets

SBU Dataset [163] contains 8 classes of simple interaction motions: walking toward, walking away, kicking, pushing, shaking hands, hugging, exchanging, and punching. The data which are too noisy, and in particular the class “hugging", have been removed from this dataset. The “walking away" and “walking toward" classes have the same reactions (standing still), so we decided to fuse those two classes into a single “walking" class. This leaves us with 6 classes, 195 training, and 30 test samples.

K3HI Dataset [61] contains the same 8 classes as SBU aside from the “hugging" class which is replaced by “pointing". Also, unlike SBU, “approaching" and “departing" have reactions that are different, so we do not fuse the two classes. We also removed the noisy samples from the dataset but this time we normalize the data in the same way as SBU was normalized by the authors. This leaves us with 236 training samples and 28 test samples.

DuetDance Dataset [83] contains 5 classes of dance motions: cha-cha, jive, rumba, salsa, and samba. Given the nature of the dataset, the motions are more complex than those in SBU and K3HI, and there are a lot of intra-class variabilities. We do not perform normalization, but since most samples are very long sequences (up to 160s), we decided to cut each sequence into smaller sequences of 50 frames (2s), leading to 273 training samples and 3991 test samples.

For all three datasets, the poses are represented by their absolute 3D coordinates, furthermore, training and testing splits are selected randomly for fair comparisons. Duet-Dance was provided with neither train/test split nor subject information, we used a random split. For the two others, the evaluation proposed by their respective authors is made using k-fold validation so we decided to split the dataset between train and test, randomly for K3HI and by selecting all the samples from a random subject for SBU.

3.4.2 Evaluation Metrics

We use metrics commonly used in motion generation. Metrics used for motion prediction based on the distance between the generated sample and the ground truth are not fit for

reaction generation as several different motions can be considered good reactions to the same action. While this choice of metric can seem contradictory with our losses that use direct comparison with the ground truth, it is important to understand that our evaluation metrics do not contain direct information about the skeleton that our network is supposed to generate and could not be efficiently used as losses.

Classification Accuracy measures how well our generated samples are classified by a motion classifier. We use the DeepGRU classifier [105]. We only train and test the classifier on the reaction part of the interaction, so the results are not influenced by the action, which is always the ground truth. We report the percentage of correctly classified samples for each class and the average over the entire test set.

Fréchet Video Distance (FVD) is an adaptation of the Fréchet Inception distance (FID) [58] for video sequences [144]. FVD computes the distance between the ground truth and the generated data distribution.

$$\text{FVD} = |\mu_{gt} - \mu_{gen}|^2 + \text{tr}\left[\mathbf{C}_{gt} + \mathbf{C}_{gen} - 2\left(\mathbf{C}_{gt} * \mathbf{C}_{gen}\right)^{1/2}\right], \tag{3.10}$$

where μ_{gt}, μ_{gen} and $\mathbf{C}_{gt}$ and $\mathbf{C}_{gen}$ are the means and covariance matrices of the deep features from ground truth and the generated samples respectively, tr$(\cdot)$ is the trace. The deep features are obtained from the classifier used for the classification accuracy

Diversity Score. Following the metric defined by [86, 166] we compute the average deep feature distance between all the samples generated by each method and then compare it to the average deep feature distance of the ground truth. A low diversity score means that the generated samples have a diversity close to that of the ground truth and a high score means that the diversity is either lower (all motions are more similar) or higher (more noise in the generation). The average deep feature distance is calculated as follows:

$$div = \frac{1}{b(b-1)} \sum_{i=1}^{b} \sum_{j=1}^{b} ||F_i - F_j||_2, \tag{3.11}$$

where b is the number of samples considered, F_i and F_j are deep features of the samples i and j, respectively. The score is obtained using div_{gt} the diversity distance of the ground truth and div_{gen} the diversity of the generated samples.

$$score = 100 \times \frac{|div_{gt} - div_{gen}|}{div_{gt}}. \tag{3.12}$$

3.4.3 Baselines

To our knowledge, there is no work that deals with the generation of the reaction to an action, so to be able to compare our results to others from the literature, we employ a method for human interaction generation and a method for human motion prediction to show methods used on a range of applications.

Zero Velocity baseline (ZeroV) [109] is a simple baseline where all generated frames are the same (in our case the initial pose), there is no motion for this baseline. Using ZeroV as a

comparison is useful to see what the quantitative result of an obviously bad method are like and help see if the results from the other methods are actually good. We do not show the results for ZeroV in our qualitative evaluation as they are uninteresting since no motion is produced. We do not use them in our user study for the same reason.

Multimodal Variational Recurrent Neural Network (VRNN) [11] deals with the prediction of the future frames of a two-person interaction based on a historical sequence using variational RNNs. The next frame of the reaction is predicted using the past frames of the reaction and information on the past frames of the action; the action is predicted in the same way using the information on the reaction. The past frames are the historical sequence at the beginning and later in the sequence the generated frames. We modified the network to fit our problem. Originally the network takes n historical frames for both action and reaction as input and generates m frames for both action and reaction. We modify some parameters so the network takes $n+m$ frames for the action but only 1 for the reaction and we generate $n+m-1$ frames of reaction motion. Otherwise, we use the default settings provided by the author for the hyper-parameters.

Mix-and-Match Perturbation (MixMatch) [5] uses a recurrent encoder-decoder network with a conditional variational autoencoder block to predict the motion of a single person based on a historical sequence. However, the authors present their method as a general prediction method and the code they provide uses the first half of an image to predict the second half. Since the specific code used for motion prediction is not available we use the one provided by the author but with 3D skeletons data and with the values of the hyperparameters mentioned in [5] for human motion prediction. To ensure a fair comparison we need to base the generation of the reaction on an initial frame but directly using 3D coordinates led to strong discontinuities between the initial position and the generation. To solve this and make the comparison fairer we work with the speed of the motion that we then apply to the skeleton corresponding to the initial position.

Progressively Generating Better Initial Guesses (PGBIG) [103] is an architecture that uses Spatial Dense Graph Convolutional Networks and Temporal Dense Graph Convolutional Networks alternatively to extract spatio-temporal feature and predict human motion. We use the code provided by the authors unchanged and with the recommended parameters. We give the action motion followed by the first frame as input and predict the reaction motion.

Spatio-temporal Transformer (STT) [3] is a Transformer based architecture that uses attention to find temporal and spatial correlations to predict human motion. As for PGBIG, we use the code provided by the authors without changes and with the recommended parameters. As input, we use the action motion followed by the first frame and predict the reaction motion.

3.4.4 State-of-the-Art Comparisons

All presented evaluations were obtained on a model trained on the considered dataset. This is true for our Interformer as well as the baselines.

Quantitative Evaluation. Table 3.2 (left) shows the classification accuracy for SBU, Duet-

Table 3.2: **Left**: Classification accuracy for each class of the SBU, DuetDance, and K3HI datasets. **Right**: User study for each class of the SBU, DuetDance, and K3HI datasets.

Method	GT	ZeroV [109]	VRNN [11]	MixMatch [5]	STT [3]	PGBIG[103]	InterFormer	GT	VRNN [11]	MixMatch [5]	InterFormer
	Classification Accuracy ↑							User Preference ↑			
SBU											
Walking	100.0	0.0	58.3	**100.0**	91.7	58.3	**100.0**	34.2%	21.4%	15.5%	28.9%
Kicking	66.7	66.7	0.0	0.0	0.0	0.0	**33.3**	38.8%	23.8%	5.6%	31.8%
Pushing	80.0	0.0	**60.0**	0.0	0.0	0.0	**60.0**	35.6%	19.7%	15.4%	29.3%
Shaking Hands	100.0	0.0	0.0	0.0	0.0	0.0	**100.0**	37.5%	21.8%	7.8%	32.9%
Exchanging	80.0	0.0	**80.0**	0.0	50.0	60.0	60.0	41.9%	19.4%	13.0%	25.7%
Punching	100.0	33.3	0.0	33.3	0.0	0.0	**100.0**	43.1%	19.3%	11.3%	26.3%
Average	90.0	10.0	46.7	43.3	40.0	33.3	**80.0**	38.5%	20.9%	11.4%	29.2%
DuetDance											
Cha-Cha	28.0	1.8	26.4	19.2	**37.1**	28.6	26.7	45.9%	17.8%	5.5%	30.8%
Jive	24.6	0.4	13.8	**25.8**	16.7	19.7	22.8	48.4%	13.2%	6.7%	31.7%
Rumba	34.8	0.7	**36.4**	30.0	30.0	34.5	32.0	40.7%	16.9%	8.2%	34.2%
Salsa	27.8	93.1	**29.5**	28.9	10.0	20.2	28.1	49.3%	12.8%	7.1%	30.8%
Samba	22.2	18.6	21.0	17.2	18.2	**24.4**	**24.4**	44.5%	15.8%	6.3%	33.6%
Average	28.0	24.6	26.2	24.9	24.4	25.7	**27.1**	45.8%	15.3%	6.7%	32.2%
K3HI											
Approaching	100.0	25.0	**75.0**	50.0	50.0	0.0	0.0	34.9%	23.1%	14.1%	27.9%
Departing	33.3	33.3	0.0	**33.3**	**33.3**	**33.3**	**33.3**	34.2%	24.2%	13.2%	28.4%
Kicking	40.0	80.0	0.0	40.0	40.0	0.0	**60.0**	31.8%	21.7%	18.8%	27.7%
Pushing	100.0	33.3	33.3	33.3	33.3	33.3	**66.7**	33.1%	24.8%	13.5%	28.6%
Shaking	50.0	0.0	50.0	50.0	**100.0**	**100.0**	50.0	36.9%	21.4%	10.8%	30.9%
Exchanging	0.0	0.0	0.0	0.0	0.0	0.0	0.0	36.3%	20.5%	13.9%	29.3%
Punching	100.0	25.0	**50.0**	25.0	0.0	**50.0**	**50.0**	33.9%	21.7%	16.9%	27.5%
Pointing	100.0	50.0	**100.0**	0.0	25.0	25.0	**100.0**	37.3%	20.2%	16.1%	26.4%
Average	67.9	35.7	39.3	28.6	32.1	25.0	**46.4**	34.8%	22.2%	14.7%	28.3%

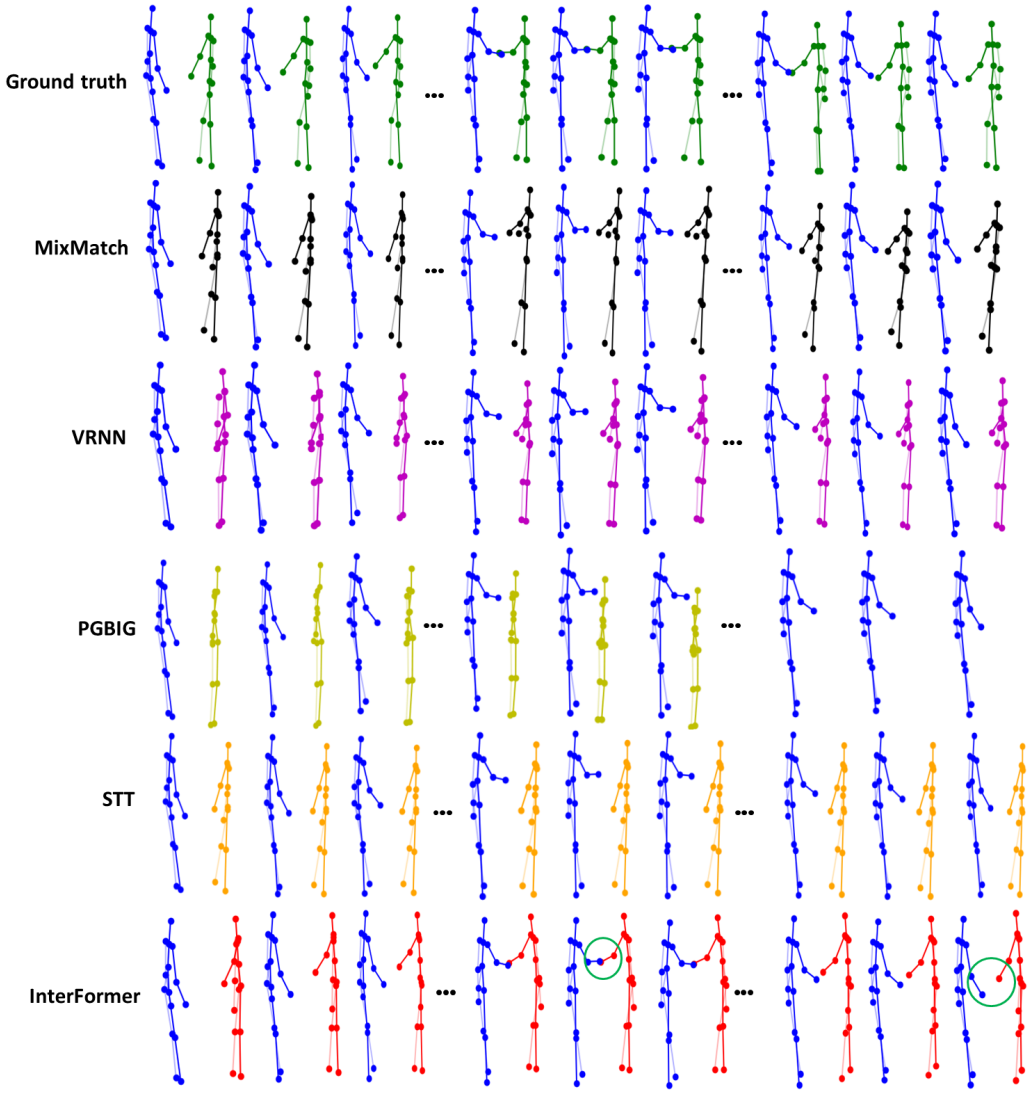


Figure 3.7: **Qualitative results.** In blue the action motion is used as a condition. In other colors, the reaction is either from the ground truth or generated by the different models. Shaking hands class from the SBU dataset.

Dance, and K3HI. Our method outperforms the five others on all the datasets. For SBU, we obtain results very close to the ground truth, and we outperform the other methods on all classes but "exchanging" where [11] get better results and vastly outperform the simple ZeroV baseline. InterFormer is able to generate simple motions that are realistic enough to be correctly classified. We can see however that on "kicking" we score less than ZeroV, this is due to the small size of the SBU dataset. A few misclassifications will cause a sharp drop in classification accuracy, and as we can see, "Kicking" is the class that has the lowest accuracy on the ground truth as the reaction can be similar to those of punching and pushing. The good performance of ZeroV in some classes can be explained by the fact that the overall accuracy is below chance (16.7%). This means that the classifier is unable to properly classify the motion from ZeroV as it only shows unmoving skeletons and for some classes, the two skeletons start in a neutral position that carries no information about the action. All

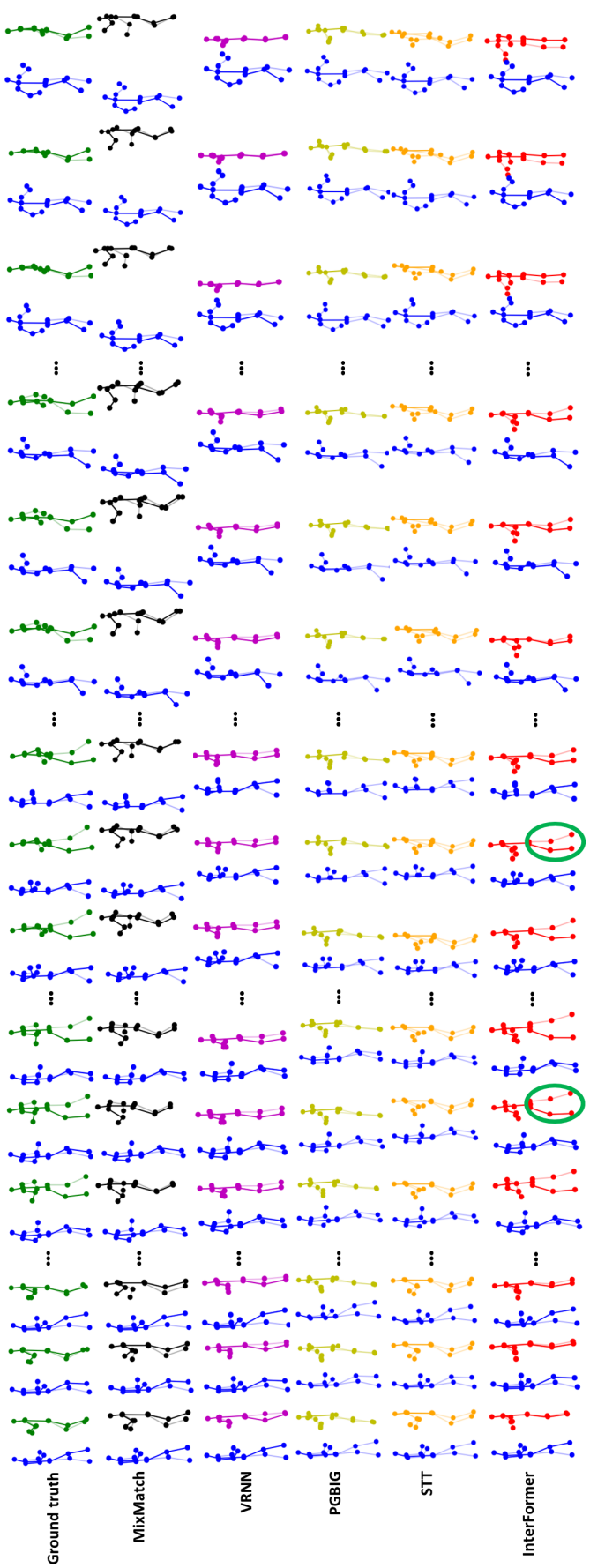


Figure 3.8: **Qualitative results.** In blue the action motion is used as a condition. In other colors, the reaction is either from the ground truth or generated by the different models. Cha-cha class from the DuetDance dataset.

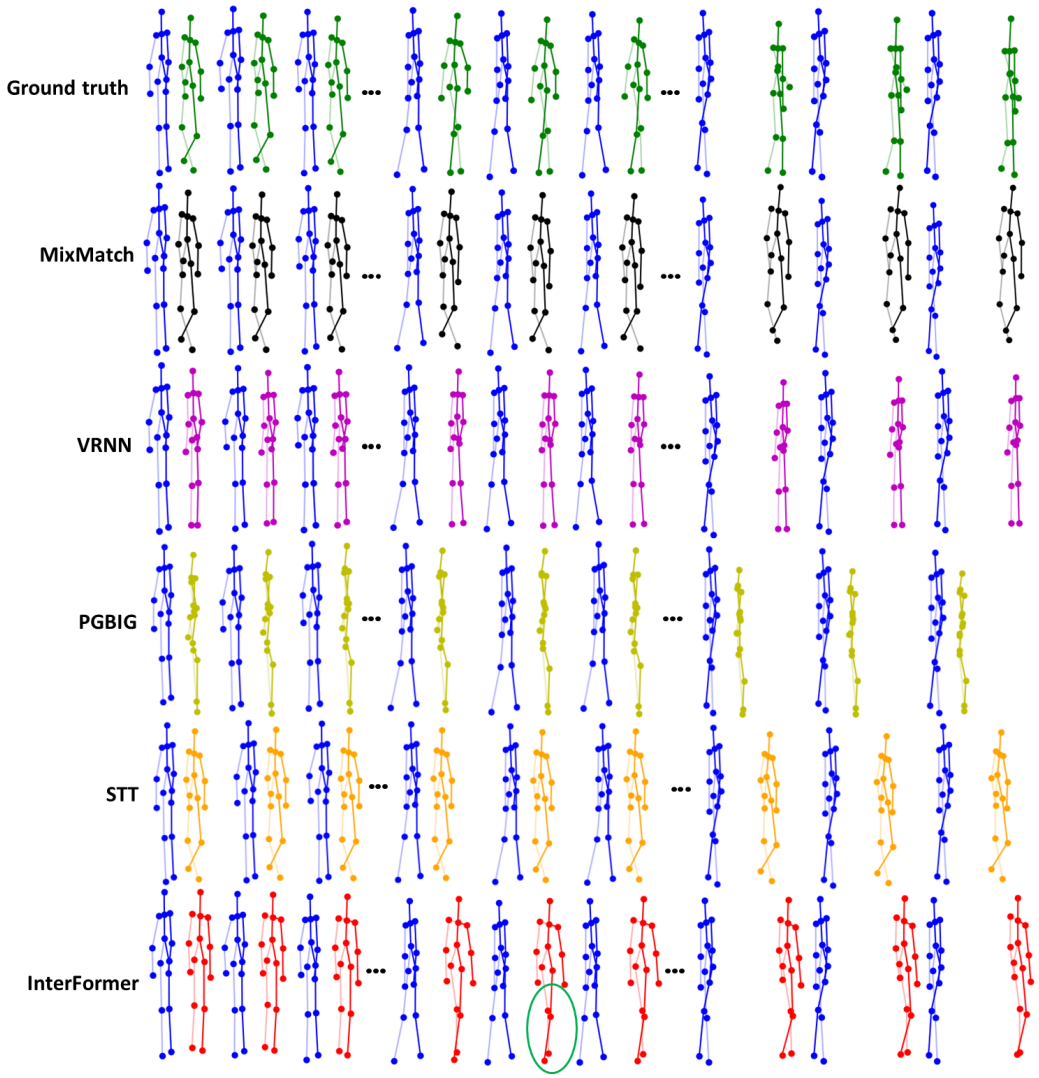


Figure 3.9: **Qualitative results.** In blue the action motion is used as a condition. In other colors, the reaction is either from the ground truth or generated by the different models. Departing class from the K3HI dataset.

these cause the classifier to fail at classifying the sample and likely classify many samples as "kicking", including some that are from the "kicking" class leading to the high score in this class.

For K3HI, we can see that the results are worse than for SBU for all methods and even for the ground truth. This is due to the very noisy nature of the K3HI dataset even after removing the worse samples (that showed extreme deformation and no recognizable motion), the exchanging class has a 0% recognition rate even for the ground truth. However, our method provides better results than the two others in all classes except "approaching" which may be due to the noisy nature of the data for this class. VRNN obtaining very high results in this class might be a consequence of the wrong classification present in many of the classes (a lot of samples are classified as approaching). For shaking PGBIG and STT obtain better results but since the results are worse overall quantitatively and qualitatively this can

Table 3.3: FVD and diversity on all datasets.

Method	FVD ↓			Diversity ↓		
	SBU	DuetDance	K3HI	SBU	DuetDance	K3HI
ZeroV [109]	493.3	41058.1	392.1	65.1	47.2	19.3
VRNN [11]	113.61	789.23	195.47	11.5	6.1	16.8
MixMatch [5]	314.38	1460.44	406.63	45.3	0.9	32.2
STT[3]	321.04	2610.95	7579.87	47.8	3.9	27.6
PGBIG[103]	267.27	317.0	379.4	35.7	1.5	**10.1**
InterFormer (Ours)	**48.78**	**31.81**	**125.40**	**0.9**	**0.4**	13.7

be explained by the classifier putting many samples in that class as we have explained for ZeroV.

For DuetDance, the classification accuracy for all methods and the GT is much closer than for the other datasets. This is due to the complex motions contained in the dataset with a lot of intra-class variabilities. Furthermore, we use sequences of 50 frames which are short enough that some sequences from two different classes can be very similar. We can still notice that our method provides results that are the closest to the ground truth and that, unlike the five other methods no class has a score below chance (i.e., 20%) which means that our results are more consistent and closer to the ground truth, despite being beaten on some individual class e.g., STT score 37.1% on "cha-cha" but only 10.0% on "salsa" while we score 26.7% and 28.1%, respectively.

In Table 3.3 we show the FVD and diversity score for all methods on all datasets. We outperform VRNN, MixMatch STT, and PGBIG on the FVD measure, often by a large margin meaning that the features extracted by the classifier are closer to the features of the ground truth than for [11] and [5]. For the diversity score, we also outperform the two other methods and provide diversity that is close to that of the ground truth. We can see a significant increase in K3HI. This is due to the noisy nature of the dataset, which means that the diversity distance of the ground truth takes into account the noise of the sample, we, however, manage to score the closest to the diversity of the ground truth when compared to the other methods, without generating noisy samples. This can also explain why PGBIG diversity is better than ours despite performing much worse in terms of classification and qualitative results.

User Study. To evaluate the quality of the generated videos, we also conduct a user study. Specifically, the users are given four videos (two generated by existing methods VRNN and MixMatch, one generated by our proposed InterFormer, and one real video) with the corresponding class label, each participant needs to answer one question: 'Which video is more realistic regardless of the input label?'. 20 users have unlimited time to select their choices. PGBIG and STT are not represented in this study due to the extremely low quality of the results, as illustrated by our qualitative results. The results are shown in Table 3.2 (right), we can see that the users show more preference for our method than the other two methods, which indicates the results generated by ours are more realistic.

Qualitative Evaluation. We show in Figures 3.7, 3.8, 3.9, and 3.10 visualizations of the gen-

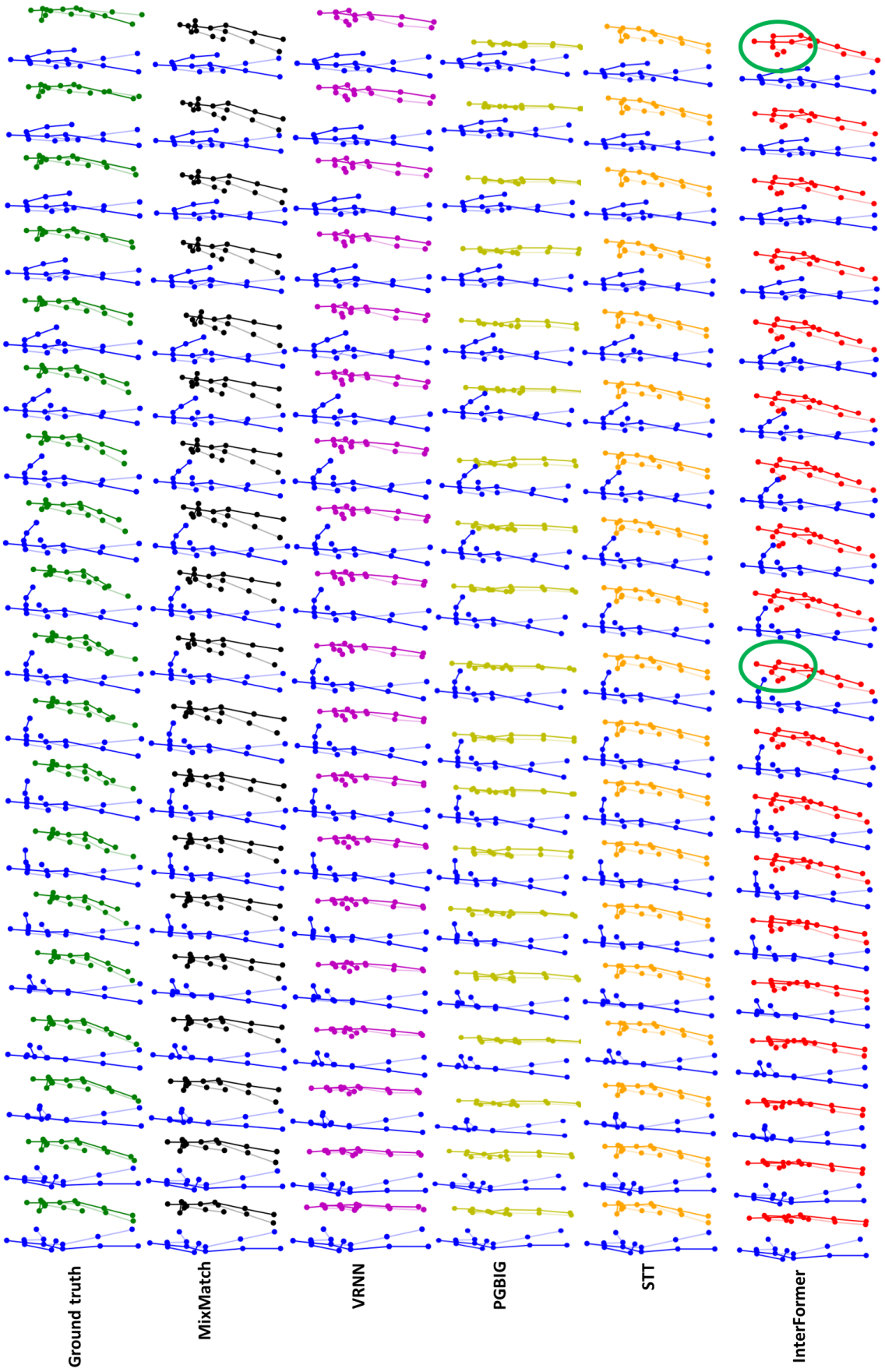


Figure 3.10: **Qualitative results.** In blue the action motion used as a condition, in other colors the reaction either from the ground truth or generated by the different models. Punching class from the SBU dataset.

erated sequences on the SBU (two sequences) DuetDance and K3HI datasets respectively. We show from top to bottom: the ground truth, results for [5], results from [11], results from [103], results from [3] and results from our InterFormer. In blue is the action motion, which serves as a condition and is in all cases the ground truth. In green, black, magenta, yellow, orange, and red are the reactions for the GT and the five methods.

In Figure 3.7, we show an interaction from the “shaking hands" class of SBU. It shows that our method is able to generate the motion better than the two other methods. For [5], the character raises its hand to shake but never comes really close to the other character’s hand and also shifts its entire body backward toward the end of the sequence. [11] generates a motion that raises slightly the hand but is then stuck in this position. [103] does not generate a shaking hand motion and fails to generate poses for the entire length of the action. STT [3] also fails to generate a shaking hand motion. Our method generates motion that is very close to the ground truth and contains the three main steps of the motion: raising the hand, shaking, and going back to starting position. Figure 3.10 shows a sample from the “punching" class from the SBU dataset. We see that we generate a better motion even if there are differences with the ground truth. The character is pushed to the side by the punch and then comes back to a normal position at the end of the sequence. The two other methods also generate a reaction to the punch, [11] moves slightly backward, and [5] moves its upper body to avoid the punch. [103] does not generate a motion that looks like a reaction to the punch and presents noise with the vertical position of the skeleton suddenly changing from one frame to the other. [3] generates a slight motion of being pushed back but the motion continues without trying to go back into a neutral position. It seems, however, that the upper body also became smaller during this motion. The two methods also stay in this avoiding pose and do not go back to a more normal position. In Figure 3.8 we show a sample of the “cha-cha" class from the challenging DuetDance dataset. We can see that [5] produces a motion that resembles a dance even if different from the ground truth, however as the action character moves backward, the generated reaction stays in place, and the distance between both characters grows over time. With [11], the distance between the two characters does not grow, but there is barely any motion for the entire sequence. In motion, it looks like the reaction character is gliding toward the action character. Here [103] and [3] generate something close to [11] with little motion, but the distance between the two skeletons does not grow. [3] also present deformations in the arms. Our method is able to generate a motion that stays close to the ground truth and follows the action character in space without gliding like [11] this can be seen by the change of position of the legs across the sequence. It is only toward the end that the motion differs from the ground truth and even then, the motion still resembles dancing.

In Figure 3.9, we see a sample of the departing class from the K3HI dataset. It shows both characters walking away from each other. This behavior is always reproduced in the samples generated by the three methods, but [11] does not show much motion and simply glides away while [5] shows more motion of the legs but keeps the noise present in the first frame during the entire sequence. Once again [103] does not generate a proper motion, and this time it shows deformation in the skeleton that stays for the entire duration of the motion. Likewise, [3] is unable to generate a proper walking motion.Our method, on the other hand, generates a realistic walking motion with both arms and legs moving to move apart from the first character.

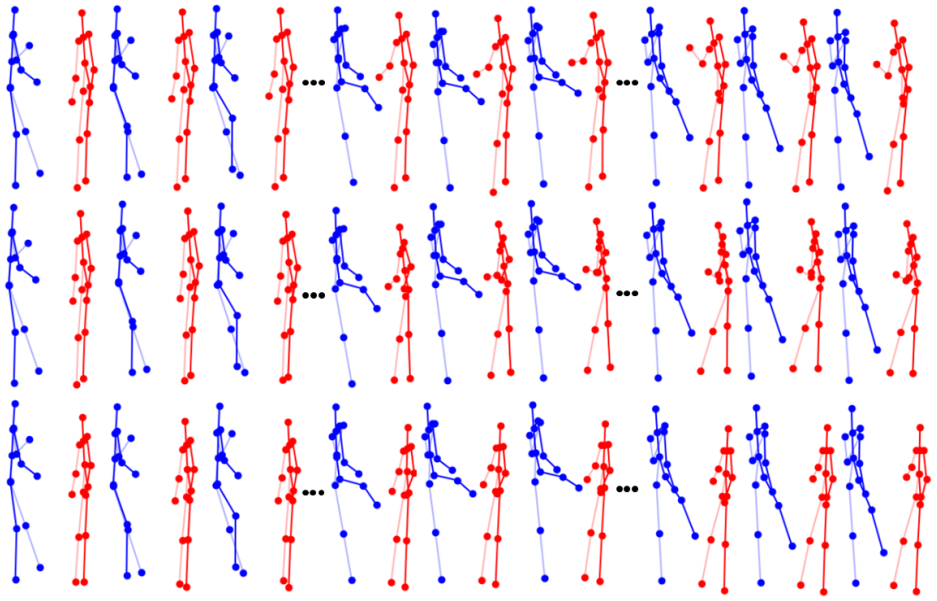


Figure 3.11: **Multi-modality results on SBU kicking class with noise.** We show three different motions generated by our Interformer based on the same input motion.

The very poor performances of PGBIG [103] and STT [3], our two baselines with unmodified code, can be explained by the fact that they were designed for human motion prediction. With human motion prediction, we seek to reduce as much as possible the discontinuities between the input and the output while we want to generate a different skeleton to the one used as input which implies a very strong discontinuity. Also, methods for human motion prediction are typically trained to always take the motion of the same duration as input and predict sequences that always have the same length e.g., the input of $500ms$ to predict $1s$ of motion. With reaction generation, the length of the sequences can vary (greatly in the case of K3HI) and the unmodified motion prediction method might struggle with the varying lengths. This is illustrated by the early stop in the generation of [103] in Figure 3.7 but also by the fact that [3] is unable to stop generating until it reaches the maximum sequence length of the dataset (not pictured in our figures).

Multi-Modality Generation. The main issue with Transformer models is that their output is deterministic. To counter this we can add noise to the encoder input before the first feed-forward layer. This allows us to generate diverse outputs for the same input motion. We show in Figure 3.11 and Figure 3.12 the ability of our method to generate diverse motions with a single input when adding noise in the encoder.

3.4.5 Ablation Study

To validate the effectiveness of each proposed component, we report the ablation studies on SBU with classification accuracy and diversity.

Ablation Models. Our Interformer has four versions (i.e., S1, S2, S3, S4) as shown in Table 3.4. (i) S1 means only using the original NPL Transformer network from [146] modified

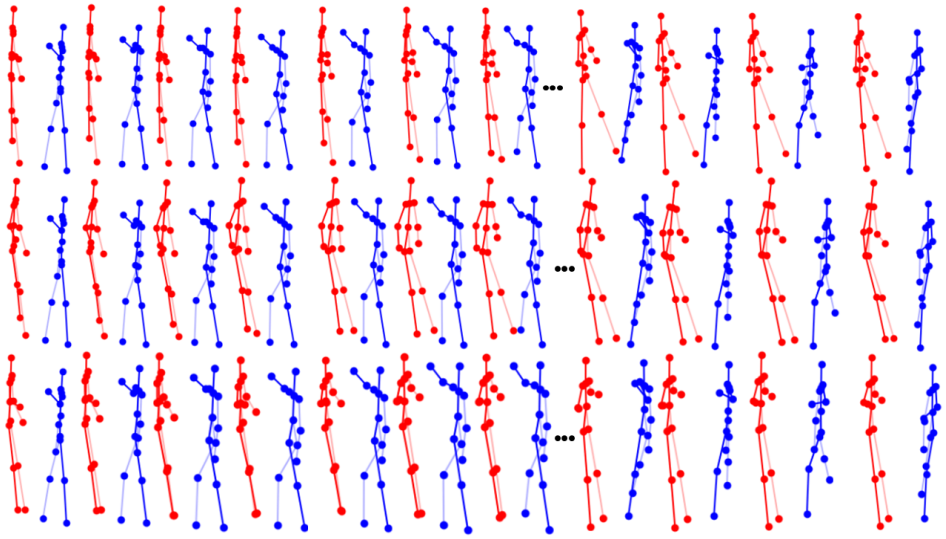


Figure 3.12: **Multi-modality results on SBU Punching class with noise.** We show three different motions generated by our Interformer based on the same input motion.

Table 3.4: Ablation study of Interformer on the SBU dataset.

	Setup	Accuracy ↑	Diversity ↓
S1	Transformer	53.3	9.5
S2	S1 + Spatial Attention	66.7	3.9
S3	S2 + Skeleton Adjacency	73.3	1.7
S4	S3 + Interaction Distance	**80.0**	**0.9**

to take as input and generate skeletons without any of our improvements. (ii) S2 adds to the global Transformer the spatial attention modules (self spatial attention and interaction spatial attention). (iii) S3 adds the skeleton adjacency module to the self spatial attention. (iv) S4 is the full model and includes both the skeleton adjacency module and the interaction distance module.

Effect of Spatial Attention. We validate the effect of spatial attention, as shown in Table 3.4. Introducing spatial attention results in significant improvement in classification accuracy by 13% and diversity by 5.6, which means we improve the quality of the action-reaction sequences.

Effect of Skeleton Adjacency. Using a skeleton adjacency graph on attention improves the classification accuracy and diversity by 7% and 2.2, respectively. This improvement means that the model learns better relations between the different joints inside a skeleton.

Effect of Interaction Distance. By adding the interaction distance module, we increase the results obtained by the skeleton adjacency module by 7% on classification and 0.8 on diversity. These results show that the interaction distance module is able to help spatial interaction attention find the most interesting relations between the two skeletons and thus help generate better motions.

Table 3.5: Ablation study of Interformer on the K3HI dataset

	Setup	Accuracy ↑	Diversity ↓
S1	Transformer	14.3	32.9
S2	S1 + Spatial Attention	28.6	16.7
S3	S2 + Skeleton Adjacency	42.9	**9.3**
S4	S3 + Interaction Distance	**46.4**	13.7

T multihead	S multihead	Accuracy ↑	Diversity ↓
-	-	60.0	10.6
✓	-	70.0	2.0
-	✓	60.0	5.1
✓	✓	**80.0**	**0.9**

Table 3.6: Ablation study of Interformer on the SBU dataset.

Abaltion on K3HI. Table 3.5 shows the ablation for the K3HI dataset and confirm our finding from the SBU ablation. The only difference is a lower diversity when using the graphs but not the interaction distance. We believe this to be due to the more noisy nature of the K3HI dataset, which deteriorates the diversity measures.

Effect of Loss on The First Frames. If we remove the loss on the first frames that allow us to keep a good coherency between the input initial position and the generation, we see a decrease in the generation quality: -3.3% in classification accuracy and -5.2 in diversity score when compared to S4. When the input initial position is not properly taken into account the generated reaction skeleton can be far from the action skeleton. In SBU, for all action classes, the interactions consist of two persons close to each other. Since the model is not trained with samples where people are far from each other when we try to generate the reaction motion of a skeleton far from the action skeleton, little to no motion is generated. This explains the increase in performance brought by the use of the first frame loss.

Effect of Multihead attention. Our Interformer uses the multihead version of attention for both temporal and spatial attention. These choices were made following results from the original Transformer network [146] and our experiments which we report in Table 3.6. The results are obtained by modifying the number of heads for the different attention modules on the full Interformer model (S4 from the ablation study). These experiments show that using the multihead temporal attention (T multihead) increases the classification accuracy by 10% and diversity by 8.6. By using only the spatial multihead attention (S multihead) we increase the diversity by 5.5. Using the multihead attention for both spatial and temporal attention led to an increase of 20% in classification accuracy and 9.7 in diversity. This confirms our choice to use this configuration for Interformer. Table 3.7 shows the same ablation for the K3HI dataset and we observe the same behavior as for SBU except for the diversity where other configurations have lower values than using both multihead attention. We believe this to be due to the more noisy nature of the K3HI dataset, which deteriorates the diversity measures.

T multihead	S multihead	Accuracy ↑	Diversity ↓
-	-	21.4	11.0
✓	-	42.6	**5.3**
-	✓	35.7	5.7
✓	✓	**46.4**	13.7

Table 3.7: Ablation study of Interformer on the K3HI dataset

.

3.5 Limitations

InterFormer presents two main limitations: (i) Due to the huge variability of complex motions, it is hard to stay true to the ground truth, making it difficult to evaluate the results in these cases; (ii) We are able to generate realistic motion for long sequences (tested up to 40 seconds) To do this we cut the action sequence into smaller sub-sequences that we use for generation. We then generate all these sequences the same way as we do for shorter sequences. Only for the second sub-sequence onward the first frame used to give the initial position does not come from ground truth but instead is the last generated frame from the previous sub-sequence. This way InterFormer is able to generate reaction sequences for long motion. However, due to the accumulation of errors over time, the generation diverges more and more from the ground truth up to the point where it is hard to know how much the action is taken into account in the generation. It is even more true that very long motions are usually complex ones, which means we also face the first limitation.

3.6 Conclusion

We present InterFormer, a novel human reaction generation Transformer. InterFormer is the first Transformer architecture used to solve the problem of human reaction generation challenge. InterFormer consists of four modules: a motion encoder, a motion decoder, a skeleton adjacency module, and an interaction distance module. The ablation study on SBU has shown the effectiveness of the four components of the InterFormer. We have both qualitatively and quantitatively evaluated our reaction generation framework. The results show that InterFormer outperforms state-of-the-art approaches in terms of FVD, classification, and diversity score on three challenging datasets SBU, K3HI, and DuetDance. The qualitative results show also the ability of InterFormer to generate realistic human reactions. Interformer is a deterministic approach. Although we have proposed an approach to mitigate this problem, the diversity of responses generated remains limited and should be improved. It is still difficult to generate complex human motion. Although our results on the dance dataset show that we are able to generate dance movements, we are still not able to generate more subtle motions present in the dataset. The lack of large interaction datasets makes it difficult to evaluate feedback generation. Although large interaction datasets exist, such as some classes of NTUs, they are not annotated to separate action from reaction motion. It is difficult to evaluate the performance on long-term motion due to the lack of appropriate data.

Part IV

Bipartite Graph Diffusion Model for Human Interaction Generation

CHAPTER 4

BIPARTITE GRAPH DIFFUSION MODEL FOR HUMAN INTERACTION GENERATION

4.1 Introduction

Modeling dynamics of human motion interaction is at the core of many applications in computer vision and computer graphics. Most works on human motion generation ignore human interactions and focus instead on the generation of actions of a single person [165, 141]. In this paper, we explore the problem of generating 3D human motion interaction. What makes interaction generation challenging are the non-linearity of human motion interaction and the diversity of the interaction between humans. Several questions arise to tackle these challenges. How to represent the interaction between humans? How to model motion and generate diverse motion interaction? To solve the first question, we propose to represent the skeleton interaction by using a bipartite graph [138]. The main goal of the bipartite graph is to capture the relations between humans represented by skeletons. To solve the second question, the motion interaction generation is formulated as a reverse diffusion process. Overall, our contributions are summarized as follows:

- We propose the first Bipartite graph denoising diffusion model (BiGraphDiff) for human interaction generation. Our BiGraphDiff is able to generate motion interaction in a stochastic way, naturally leading to high diversity, and is able to generate very long motion sequences (>1000 frames).
- BiGraphDiff is a denoising diffusion process that learns not only the denoising of the motion, but also it learns a Bipartite graph. The aim of Bipartite graph is to capture the relations between the two persons.
- BiGraphDiff achieves state-of-the-art quantitatively and qualitatively in action interaction and dance tasks. A user study shows that the generated sequences are better qualitatively than the sequences generated by state-of-the-art methods.

The code and some videos of generated interaction are available at https://github.com/CRISTAL-3DSAM/BiGraphDiff

4.2 Related Work

We discuss the relevant literature from two perspectives, namely, previous methods of Human interaction motion synthesis and the literature on diffusion models.

Human Interaction Motion Generation. Recently there has been an increase in motion generation based on different modalities, [159] use control signals such as the global trajectory of the person to generate human motion in long-term horizons while [1] and [56] generate motion based on speech audio. Meanwhile, others use only knowledge of the past motion which allows them to work in real-time but on shorter motion [109, 34, 134]. More recently, several works have been dedicated to human pose and motion generation from text or action labels, as well as its reciprocal task [54, 98]. These papers focus only on one person, while our approach is dedicated to the generation of two-person interactions. [12] propose

a multimodal variational recurrent neural network to predict the future motion of both participants in an interaction based on pasts sequences of motion. In contrast, we propose to generate human interaction between two persons.

Generative Diffusion Models. Diffusion models [135, 59] have shown great promise in terms of generative modeling by showing impressive results in synthesis applications ranging from image generation [57], audio-drive motion synthesis [4], molecule generation [60], to text-driven motion generation [129]. More recently, some concurrent work in the field of text-to-motion introduces a diffusion-based method for generating text-conditioned motion. For example, [165] propose MotionDiffuse, a diffusion model-based text-driven motion generation framework. [141] propose EDGE, a method for generating editable dances that is able to create a realistic dance while remaining faithful to the original music. [35] introduce MoFusion, a denoising-diffusion-based framework for high-quality conditional human motion synthesis that can generate long and temporally plausible motions conditioned based on music or text. Despite achieving impressive performance, these methods use a diffusion-based method for generating the motion of only one person. In contrast, our proposed method BiGraphDiff proposes to generate the interaction between two persons and propose to learn a bipartite graph during the diffusion process. In addition, BiGraphDiff is applied for both text-to-motion and text-to-dance, and it is able to generate a long sequence of dance motion.

Graph Neural Networks Graphs are a way to describe pairwise relations between entities and are used in various domains including physics, chemistry, and computer science. A graph can be represented as $G = \{V, E\}$, with $V = \{v_1, ...v_n\}$ the n nodes, also called vertices, and $E = \{e_1, ..., e_m\}$ the m edge connecting the nodes. A graph can be represented

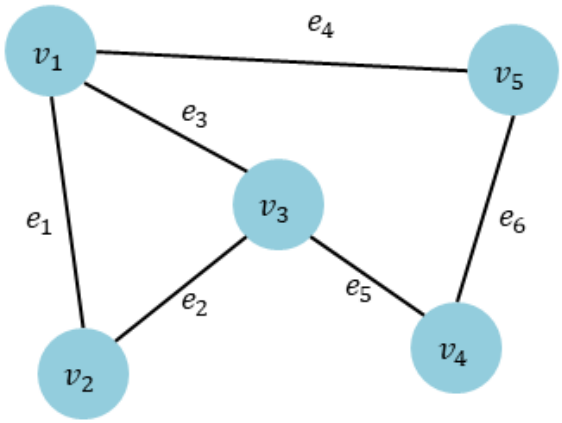


Figure 4.1: Example of a graph with 5 nodes and 6 edges.

visually as seen in Fig.4.1 or as an adjacency matrix. For a graph $G = \{V, E\}$, the adjacency matrix $A \in \{0, 1\}^{n \times n}$ represents the connections between the nodes. Specifically, $A_{ij} = 1$ if v_i and v_j are connected and $A_{ij} = 0$ otherwise. For the graph in Fig.4.1 the corresponding adjacency matrix is:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.1}$$

The degree of a graph's node corresponds to the number of nodes it is connected to. Using the adjacency matrix the degree of node v_i is:

$$d(v_i) = \sum_{j=1}^{n} A_{ij} \tag{4.2}$$

We define the Laplacian matrix of a graph by

$$L = D - A \tag{4.3}$$

with $D = diag(d(v_1), ..., d(v_n))$ the diagonal degree matrix. This Laplacian matrix is useful to create a link between the discrete graph and a continuous representation, *e.g.* vector spaces.

Graphs have been widely used for computer vision tasks, such as visual question answering [122] or human motion recognition [26]. Graph Neural Networks (GNN) [130] are commonly used for these tasks. In recent years, graph convolutional networks (GCN) [151], a version of GNN that uses graph convolution. The graph convolution is computed as follows:

$$H = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}XW) \tag{4.4}$$

where σ is an activation fonction *e.g.* RELU, $\hat{A} = A + I$ with I the identity matrix, $\hat{D}$ a diagonal matrix where $\hat{D}_{ii} = \sum_{j=1}^{n} \hat{A}_{ij}$, X a matrix of feature representing the nodes, and W the trainable weights. This equation corresponds to Laplacian smoothing [88] with specific parameters. Laplacian smoothing considers a graph as a curve made of discrete points and tries to smooth it. The smoothing is done following:

$$X_{smooth} = (I - \omega L_{rw})X \tag{4.5}$$

where $L_{rw} = \hat{D}^{-1}L$ is the normalized Laplacian and $0 \leq \omega \leq 1$ a parameter to control the strength of the smoothing. If $\omega = 1$ and L_{rw} is replaced by $L_{sym} = \hat{D}^{-\frac{1}{2}}L\hat{D}^{-\frac{1}{2}}$ then we have:

$$X_{smooth} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}X \tag{4.6}$$

This is the same form as we have in Equation.4.4 showing that the graph convolution is indeed a Laplacian smoothing.

In this chapter, we use a specific type of graph to model the interactions between two persons. The graph we use is the bipartite graph. A graph is a bipartite graph if its vertices

can be split into two independent sets U and V. Those set are called the parts of the graph. The edge of a bipartite graph can only connect a vertex of U to a vertex of V, but not two vertices of the same set. In Figure.4.2 we show an example of a bipartite graph. If every vertices of U connect to every vertices of V then we call it a complete bipartite graph. Bipartite graphs are used in various domains from coding theory to computer science [76, 89, 140, 139]. With BiGraphDiff we use them to model the interaction between two 3D skeletons.

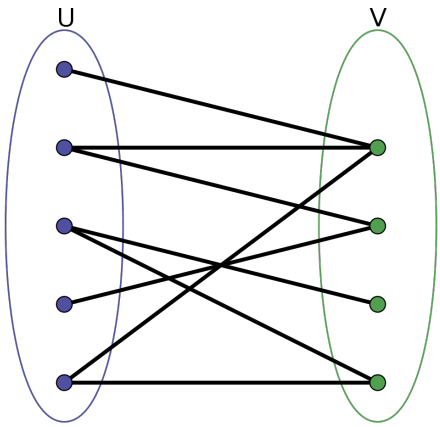


Figure 4.2: Example of Bipartite graph. U and V are in different colors
source: https://fr.wikipedia.org/wiki/Graphe_biparti

4.3 Bipartite Graph Diffusion Model

4.3.1 Framework Overview

Our goal is to generate a human motion interaction $x^{1:N}$ given an arbitrary condition c. Let us consider $x^{1:N}=\{x^1, \ldots, x^N\}$ an arbitrary sequence of joints that compose the two skeletons, $x^i \in \mathbb{R}^{k\times 3\times 2}$, where k is the number of joints. The motion generation is formulated as a reverse diffusion process that requires sampling a random noise $x_t^{1:N}$ from noise distribution to generate a motion sequence. While the forward process requires successively corrupting the motion sequence $x_t^{1:N}$ by adding the noise to the motion sequence for T timesteps in Markov fashion. We propose Transformers to learn the denoising function and a bipartite graph to represent the relationship between the joints of the skeleton. The proposed Transformer learns not only the denoising function but also the bipartite graph. See Fig. 4.3 for an overview.

4.3.2 Diffusion for Motion Generation

The diffusion model consists of two separate processes called forward diffusion and reverse diffusion. During the forward diffusion process, we add to real data a small amount of Gaussian noise repeatedly until the data becomes Gaussian noise. Formally, the forward process

on a real sample from a real data distribution $x_0^{1:N} \sim q(x^{1:N})$ consists in a Markov chain that gradually adds noise following a variance schedule β_t to obtain the posterior $q(x_{1:T}^{1:N}|x_0^{1:N})$ with $x_1^{1:N}$ to $x_T^{1:N}$ the latent data:

$$\begin{aligned} q(x_{1:T}^{1:N}|x_0^{1:N}) &:= \prod_{t=1}^{T} q(x_t^{1:N}|x_{t-1}^{1:N}), \\ q(x_t^{1:N}|x_{t-1}^{1:N}) &:= \mathcal{N}(x_t^{1:N}; \sqrt{1-\beta_t}x_{t-1}^{1:N}, \beta_t \mathbf{I}). \end{aligned} \tag{4.7}$$

Eventually with $T \to +\infty$ the distribution will be close to $\mathcal{N}(\mathbf{0}, \mathbf{I})$. This formulation implies that the forward process is recursive but this can be avoided by using [59] formulation:

$$q(x_t^{1:N}|x_0^{1:N}) = \sqrt{\overline{\alpha_t}}x_0^{1:N} + \epsilon\sqrt{1-\overline{\alpha_t}}, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4.8}$$

with $\overline{\alpha_t} = \prod_{i=0}^{t} \alpha_i$ and $\alpha_t = 1 - \beta_t$. With this formulation, we can sample a noise ϵ and directly generate any $x_t^{1:N}$. Forward diffusion does not require any training but only gradually adds noise to real data. To generate motions, we need to be able to obtain clean data from noisy data to reverse the forward process.

The reverse diffusion process, $p_\theta(x_{0:T}^{1:N})$, is a Markov chain that eliminates the noise from $x_T^{1:N}$ recursively until we obtain $x_0^{1:N}$. With $p(x_T^{1:N}) = \mathcal{N}(x_T^{1:N}; \mathbf{0}, \mathbf{I})$:

$$\begin{aligned} p(x_{0:T}^{1:N}) &:= p(x_T^{1:N}) \prod_{t=1}^{T} p_\theta(x_{t-1}^{1:N}|x_t^{1:N}), \\ p_\theta(x_{t-1}^{1:N}|x_t^{1:N}) &:= \mathcal{N}(x_{t-1}^{1:N}; \mu_\theta(x_t^{1:N}, t, c), \Sigma_\theta(x_t^{1:N}, t, c)). \end{aligned} \tag{4.9}$$

During the denoising process the goal is to estimate $\mu_\theta(x_t^{1:N}, t, c)$ and $\Sigma_\theta(x_t^{1:N}, t, c)$. However, if we use Eq. (4.8) formulation and [59] method then we can set $\Sigma_\theta(x_t^{1:N}, t, c) = \sigma_t^2 \mathbf{I}$ with σ_t a constant and replace $\mu_\theta(x_t^{1:N}, t, c)$ as follow:

$$\mu_\theta(x_t^{1:N}, t, c) = \frac{1}{\sqrt{\alpha_t}}(x_t^{1:N} - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha_t}}}\epsilon_\theta(x_t^{1:N}, t, c)), \tag{4.10}$$

this means that we only need to estimate $\epsilon_\theta(x_t^{1:N}, t, c)$ to be able to denoise the latent data since we can recover $x_{t-1}^{1:N}$ using:

$$x_{t-1}^{1:N} = \frac{1}{\sqrt{\alpha_t}}(x_t^{1:N} - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha_t}}}\epsilon_\theta(x_t^{1:N}, t, c)) + \sigma_t \gamma, \tag{4.11}$$

with $\gamma \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In our model we set $\sigma_t = \log(\beta_t \frac{1-\alpha_{t-1}}{1-\alpha_t})$ following [59] recommendation. To estimate $\epsilon_\theta(x_t^{1:N}, t, c)$ we will train a Bipartite Graph Interaction Transformer (defined in Sec. 4.3.3) to minimize the loss:

$$\begin{aligned} L :=& E_{t\in[1,T], x_0^{1:N}\sim q(x_0^{1:N}), \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\|\epsilon - \epsilon_\theta(x_t^{1:N}, t, c)\|^2] \\ :=& E_{t\in[1,T], x_0^{1:N}\sim q(x_0^{1:N}), \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\|\epsilon - \epsilon_\theta(\sqrt{\overline{\alpha_t}}x_0^{1:N} \\ &+ \epsilon\sqrt{1-\overline{\alpha_t}}, t, c)\|^2]. \end{aligned} \tag{4.12}$$

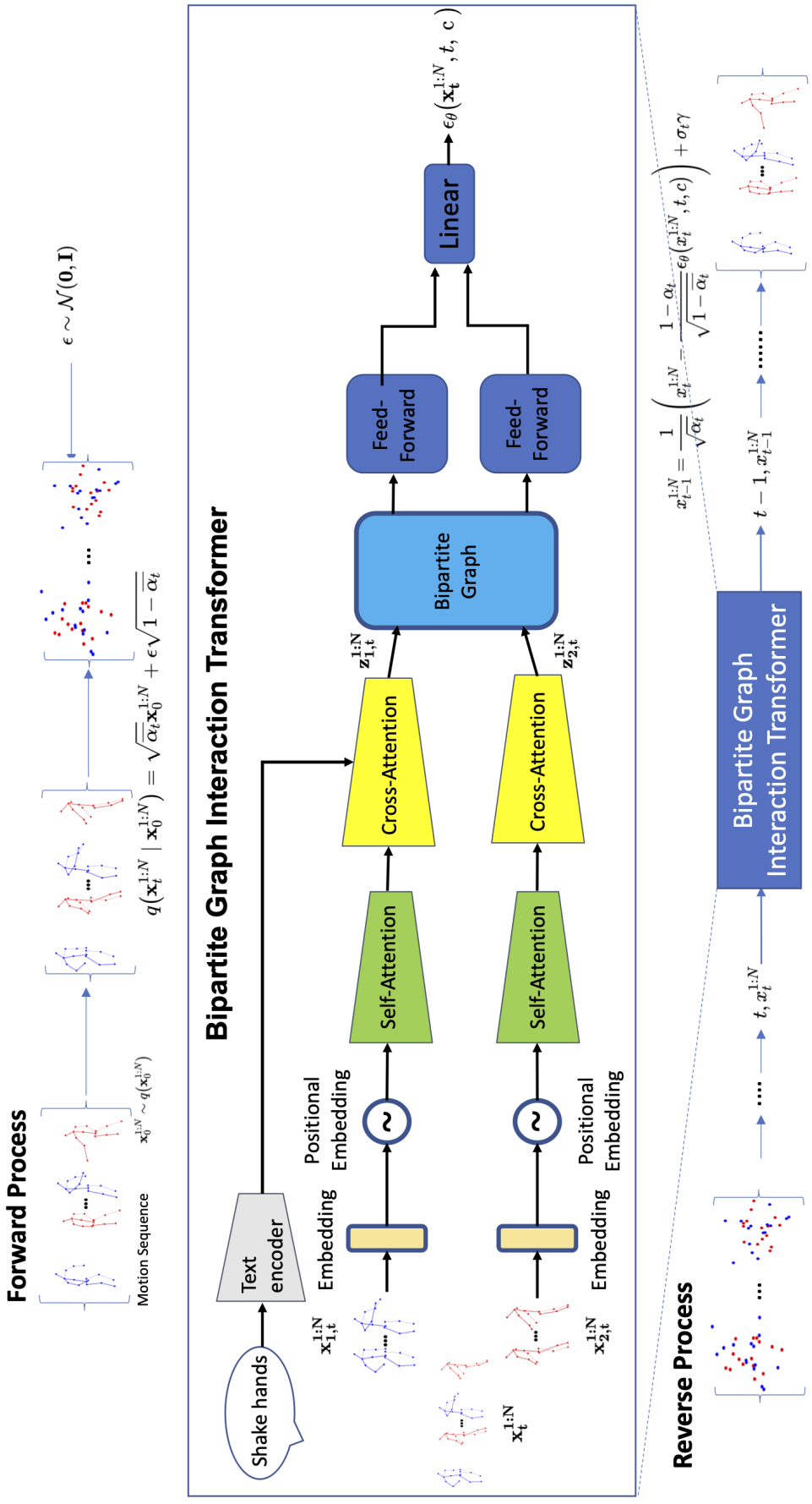


Figure 4.3: **BiGraphDiff overview.** Top: the forward diffusion process to add noise to the motion sequence. Middle: the proposed Bipartite Graph Interaction Transformer to learn the denoising function. Bottom reverse diffusion process to generate motion sequence from noise.

4.3.3 Bipartite Graph Interaction Transformer

The Bipartite Graph Interaction Transformer used by BiGraphDiff is based on the original Transformer [146]. It is composed of a text encoder, embedding and positional encoding layers, self-attention modules, cross-attention modules, a Bipartite graph module, feed-forward modules, and a final linear layer. We input $x_t^{1:N}$ and c to obtain $\epsilon_\theta(x_t, t, c)$.

Text Encoder. The text encoder is used to encode c the class label. We use a simple four layers Transformer encoder as described in [146] that uses multi-head self-attention. To avoid training the encoder from scratch, we initialize the weight with those of CLIP [124].

Motion Decoder. The motion decoder uses $x_t^{1:N}$ and the output of the text encoder to obtain $\epsilon_\theta(x_t, t, c)$. First we split $x_t^{1:N}$ into $x_{1,t}^{1:N}$ and $x_{2,t}^{1:N}$ which represent the first and second skeleton, respectively. Each skeleton passes through an embedding layer followed by a positional encoding layer introduced by [146] that encodes the temporal information from each frame of the sequence.

Self-Attention and Cross-Attention Then the data goes through self-attention and cross-attention layers. Attention is used to find correlations within the data and is defined as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}, \tag{4.13}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value matrices that have the same size as $x_{1,t}^{1:N}$ and $d{=}k * 3$ the dimension of one frame from $x_{1,t}^{1:N}$. $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are defined for self-attention for the first skeleton as

$$Q = x_{1,t}^{1:N}\mathbf{W}_q, \qquad K = x_{1,t}^{1:N}\mathbf{W}_k, \qquad V = x_{1,t}^{1:N}\mathbf{W}_v, \tag{4.14}$$

and for cross-attention

$$Q = x_{1,t}^{1:N}\mathbf{W}_q, \qquad K = c_{emb}\mathbf{W}_k, \qquad V = c_{emb}\mathbf{W}_v, \tag{4.15}$$

where $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$ are the weight matrices for the projection and c_{emb} the text embedding from the text encoder. This type of attention is also used in the text encoder. The issue with using this attention is its complexity. Indeed the complexity is N^2d. This means that long sequences take a very long time to be processed on top of taking a lot of memory space. To solve this issue and following the similar observation from [165], we use efficient attention instead. Efficient attention was introduced by [132] to have a linear complexity on attention by calculating a global feature map instead :

$$\begin{aligned} \mathbf{F} &= \text{softmax}(\mathbf{K}^T)\mathbf{V}, \\ \text{Attention} &= \text{softmax}(\mathbf{Q})\mathbf{F}. \end{aligned} \tag{4.16}$$

This simple modification allows us to get a complexity of $d_h^2 N h$ for the attention with h the number of heads and d_h the dimension of each head. d_h being a fixed value and much smaller than N, the complexity is much lower than standard attention. These heads are a part of the multi-head attention, a concept introduced by [146] where we split the inputs into smaller parts of size d_h. Each part is fed to a head that contains its own attention module.

The output of all the heads is then concatenated. We use 8 heads for both self-attention and cross-attention. Each attention layer (self and cross) is followed by a stylization block. This module, introduced by [165] allows the generative process to keep track of the current diffusion timestep t improving the generation. The output of this module is added to the input of the attention through a residual connection.

Bipartite Graph. Following the self-attention and cross-attention module, both skeletons, $z_{1,t}^{1:N}$ and $z_{2,t}^{1:N}$ go through the bipartite graph module. The proposed bipartite graph aims to capture the long-range cross relations between the two skeletons $S_a{=}z_{1,t}^{1:N}$ and $S_b{=}z_{2,t}^{1:N}$ in a bipartite graph via GCNs. Each node in S_a is connected to all the nodes in S_b. Firstly, S_a and S_b are separately fed into two encoders to obtain the feature F_a and F_b, respectively.

We then reduce the dimension of F_a with the function $\varphi_a(F_a){\in}\mathbb{R}^{C\times D_a}$, where C is the number of feature map channels, and D_a is the number of nodes of F_a. Meanwhile, we reduce the dimension of F_b with the function $\theta_b(F_b){=}H_b^\intercal{\in}\mathbb{R}^{D_b\times C}$, where D_b is the number of nodes of F_b. Next, we project F_a to a new feature V_a in a bipartite graph using the projection function H_b^T. Thus we have:

$$V_a = H_b^\intercal \varphi_a(F_a) = \theta_b(F_b)\varphi_a(F_a), \tag{4.17}$$

where both functions $\theta_b(\cdot)$ and $\varphi_a(\cdot)$ are implemented using a $1{\times}1$ convolutional layer. This results in a new feature $V_a{\in}\mathbb{R}^{D_b\times D_a}$ in the bipartite graph, which represents the cross relations between the nodes of the skeleton F_b and the skeleton F_a.

After projection, we employ a fully connected bipartite graph with adjacency matrix $A_a{\in}\mathbb{R}^{D_b\times D_b}$. We then use a graph convolution to learn the long-range cross relations between the nodes from both skeletons, which can be represented as:

$$M_a = (\mathrm{I} - A_a)V_a W_a, \tag{4.18}$$

where $W_a{\in}\mathbb{R}^{D_a\times D_a}$ denotes the trainable edge weights. We use Laplacian smoothing [25, 88] to propagate the node features over the bipartite graph. The identity matrix I can be viewed as a residual sum connection to alleviate optimization difficulties. We randomly initialize both the adjacency matrix A_a and the weights W_a and then train them by gradient descent.

After the cross-reasoning process, the new updated feature M_a is mapped back to the original coordinate space for further processing. Next, we add the result to the original feature F_a to form a residual connection, as follows:

$$\tilde{F}_a = \phi_a(H_b M_a) + F_a, \tag{4.19}$$

where we reuse the projection matrix H_b and apply a linear projection $\phi_a(\cdot)$ to project M_a back to the original coordinate space. Therefore, we obtain the feature $\tilde{F}_a$, which has the same dimension as the original one F_a.

Similarly, we can obtain the new feature $\tilde{F}_b$. Overall, the proposed method reasons the cross relations between feature maps of different skeletons using a bipartite graph.

Feed-Forward Network. After the bipartite graph module, the data of each skeleton goes through a feed-forward network. It is composed of linear projections, dropout, and GELU

Table 4.1: Classification score on NTU-26.

Method	GT	ACTOR [118]	MotionDiffuse [165]	BiGraphDiff
	Classification Accuracy ↑			
Punching	76.0%	1.0%	43.0%	**49.0%**
Kicking	86.0%	14.0%	61.0%	**86.0%**
Pushing	97.0%	77.0%	**86.0%**	74.0%
Pat on back	88.0%	4.0%	72.0%	**80.0%**
Point Finger	83.0%	0.0%	52.0%	**76.0%**
Hugging	97.0%	59.0%	90.0%	**97.0%**
Giving object	91.0%	34.0%	68.0%	**86.0%**
Touch pocket	93.0%	35.0%	81.0%	**84.0%**
Shaking hands	89.0%	16.0%	80.0%	**90.0%**
Walking toward	93.0%	72.0%	98.0%	**99.0%**
Walking apart	95.0%	**90.0%**	**90.0%**	**90.0%**
Hit with object	44.0%	8.0%	23.0%	**28.0%**
Wield knife	50.0%	7.0%	31.0%	**41.0%**
Knock over	85.0%	4.0%	**61.0%**	**61.0%**
Grab stuff	74.0%	0.0%	57.0%	**62.0%**
Shoot with gun	57.0%	1.0%	**46.0%**	44.0%
Step on foot	89.0%	5.0%	85.0%	**90.0%**
High five	90.0%	4.0%	75.0%	**78.0%**
Cheers and drink	90.0%	16.0%	69.0%	**92.0%**
Carry object	96.0%	**98.0%**	92.0%	95.0%
Take a photo	87.0%	19.0%	63.0%	**80.0%**
Follow	94.0%	68.0%	**90.0%**	81.0%
Whisper	83.0%	0.0%	72.0%	**79.0%**
Exchange things	88.0%	6.0%	65.0%	**78.0%**
Support somebody	94.0%	**100.0%**	94.0%	92.0%
Rock paper scissor	91.0%	6.0%	75.0%	**91.0%**
Average	84.6%	30.7%	70.0%	**77.0%**

activation functions. It is followed by a stylization block to ensure that the information about the current timestep is not lost. The output is added to the input of the feed-forward network thanks to a residual connection.

Linear Transformation. The Motion decoder described above contains 8 identical layers and the input of layer m is the output of layer $m-1$. Following those 8 layers the data of the two skeletons is concatenated and goes through a final linear projection to obtain $\epsilon_\theta(x_t, t, c)$ that we can use in our loss and to retrieve $x_{t-1}^{1:N}$.

4.4 Experiments

4.4.1 Datasets

There are few 3D motion two-persons interaction datasets. Therefore we focus on two complementary datasets. The NTU RGB+D 120 dataset [94], among its 120 classes, contains 26 classes labeled as "Mutual Actions / Two Person Interactions" which show two persons performing simple interaction motions. We take this 26 classes subset that we call NTU-26 and split each class randomly to obtain our training and testing set. The testing set contains 2,600 samples (100 per class) and the training set is 19,787 samples. The second dataset is DuetDance [83], which contains five classes of two persons dance motions for a total of 406 sequences. The motions are more complex than the one from NTU-26 and harder to classify, even for a human observer. The original dataset contains motions with great variations in lengths from 100 frames to more than 4,000. The average length is 483 frames with a median of 360 frames. While our model can generate very long motions it causes a problem when obtaining quantitative results and lower the quality of the generation due to the limited presence of some sequence of certain lengths. We decided to split the sequences into subsequences of 300 frames or less. This increases the number of samples to train our network with. This increased number of samples will also help the diffusion model since it needs a lot of data. This leaves us with 698 training samples and 125 test samples (25 per class randomly selected).

4.4.2 Implementation Details

We train our model on an NVIDIA A100 80Go GPU with PyTorch with a batch size of 128 for NTU and 64 for DuetDance. We train on NTU for 1,500 epochs and for 30,000 epochs on DuetDance.

4.4.3 Baselines

We compare BiGraphDiff to two methods from the state-of-the-art, i.e., MotionDiffuse [165] and ACTOR [118]. MotionDiffuse, a recent Diffusion and Transformer based architecture, generates a single-person motion from the text. For our experiments, the code provided by the author and recommended parameters are used. Due to the similarity with our method, however, we take the same batch size and number of epochs as for our method. ACTOR, a Transformer VAE method, generates a single-person motion. We use the code provided by the authors and retrain it on our datasets. We use the recommended parameters to run the model without SMPL [97] loss function. SMPL loss function is deactivated because SMPL is not available in NTU RGB+D and DuetDance datasets.

4.4.4 Evaluation Metrics

Classification accuracy To evaluate the generated sequence we use a classifier made of a simple Transformer encoder followed by a MLP. The classifier is trained and tested on the same training and testing sets as the generative methods. We look at the percentage of correctly classified samples in each class and the average over the entire testing set.

FVD for Fréchet Video distance, adapt the Fréchet Inception distance (FID) [58] for video sequences [144]. With FVD we compute the distance between the generated data distribution and the ground truth using deep features.

$$\text{FVD} = |\mu_{gt} - \mu_{gen}|^2 + \text{tr}\left[\mathbf{C}_{gt} + \mathbf{C}_{gen} - 2\left(\mathbf{C}_{gt} * \mathbf{C}_{gen}\right)^{1/2}\right], \tag{4.20}$$

where μ_{gt}, μ_{gen} and $\mathbf{C}_{gt}$ and $\mathbf{C}_{gen}$ are the means and covariance matrices of the deep features from ground truth and the generated samples respectively, tr($\cdot$) is the trace. We obtain the deep features from one of the last MLP layers from the classifier used to get the classification accuracy.

Multimodality Multimodality is defined as the average deep features distance of the samples generated by a method compared to the average deep features distance of the ground truth on a specific class. Multimodality allows us to see if the samples we generate are different from each other, it corresponds to intra-class diversity. To compute the average deep features distance we split the set of features of each class into two equal sets and compare the euclidean norm between the pairs formed by a member of each set and compute the average over the size of the subsets. The average deep features distance for multimodality is defined as follow:

$$dist = \frac{1}{cm} \sum_{j=1}^{c} \sum_{i=1}^{m} ||F_j i^A - F_j i^B||_2, \tag{4.21}$$

where c is the number of classes in the dataset $F_j i^A$ and $F_j i^B$ the i^{th} features of the subset A and B of class j. The multimodality is then calculated as :

$$score = 100 \times \frac{|dist_{gt} - dist_{gen}|}{dist_{gt}}, \tag{4.22}$$

with $dist_gt$ and $dist_gen$ the deep features distance of the ground truth and of the considered method respectively. The lower the multimodality the better as it means that we are close to the multimodality of real data.

4.4.5 Quantitative Results

NTU-26. Table 4.1 shows that our method outperforms the two the-state-of-art methods in terms of average accuracy. BiGraphDiff outperforms MotionDiffuse by 7.0% and ACTOR by 46.3%. We are also very close to the accuracy of the classifier on the ground truth. This shows that the sequences generated by our method are realistic and correspond to the input class. In more detail, we can see that we outperform or equate the other methods on 22 classes out of 26. MotionDiffuse and ACTOR are both better in 2 classes. However, we

Table 4.2: FVD and Multimodality on NTU-26.

Method	FVD↓	Multimodality↓
ACTOR [118]	25298.73	34.91
MotionDiffuse [165]	1292.32	14.94
BiGraphDiff	**1048.13**	**11.28**

Table 4.3: Classification score on DuetDance.

Method	GT	ACTOR [118]	MotionDiffuse [165]	BiGraphDiff
	Classification Accuracy ↑			
Cha-cha	28.0%	**36.0%**	32.0%	32.0%
Jive	52.0%	16.0%	**20.0%**	16.0%
Rumba	56.0%	16.0%	48.0%	**68.0%**
Salsa	88.0%	0.0%	64.0%	**76.0%**
Samba	52.0%	**80.0%**	32.0%	52.0%
Average	55.2%	29.6%	39.2%	**48.8%**

Table 4.4: FVD and Multimodality on DuetDance.

Method	FVD↓	Multimodality↓
ACTOR [118]	2641.08	67.79
MotionDiffuse [165]	1133.51	12.24
BiGraphDiff	**997.92**	**4.33**

can see that ACTOR results being actually better is debatable as some classes have very low accuracy, down to 0%. We can also see that the classes in which we perform the worse (i.e., "Hit with object" 28%, "Wield knife" 41%, and "Shoot with gun" 44%) are the ones where the results are also low for the ground truth. Those are classes where the main difference is the object used which is something we can not see using 3D skeleton data. Table 4.2 shows the FVD and multimodality results. In terms of FVD and Multimodality, our method also outperforms the two other methods indicating that our method produces sequences closer to the real data. One issue when using the NTU dataset is that it is very noisy (see the ground truth in the qualitative results). This means that it is harder to generate noiseless sequences but also that a method that generates samples without noise might be disadvantaged in the quantitative results since they are compared with the ground truth and the classifier is trained on the noisy data.

DuetDance. Table 4.3 shows the classification results on the DuetDance dataset. We can see that, as on NTU-26, we have the best performance on average. The accuracy of our method is 9.6% higher than on MotionDiffuse and only 6.4% lower than on the ground truth. We can note that the accuracy for the ground truth is much lower than for NTU-26. This is due to the nature of the motion in DuetDance. The dance motions are much harder to recognize even for a human and also longer so it is not surprising that the results are worse. ACTOR, on the

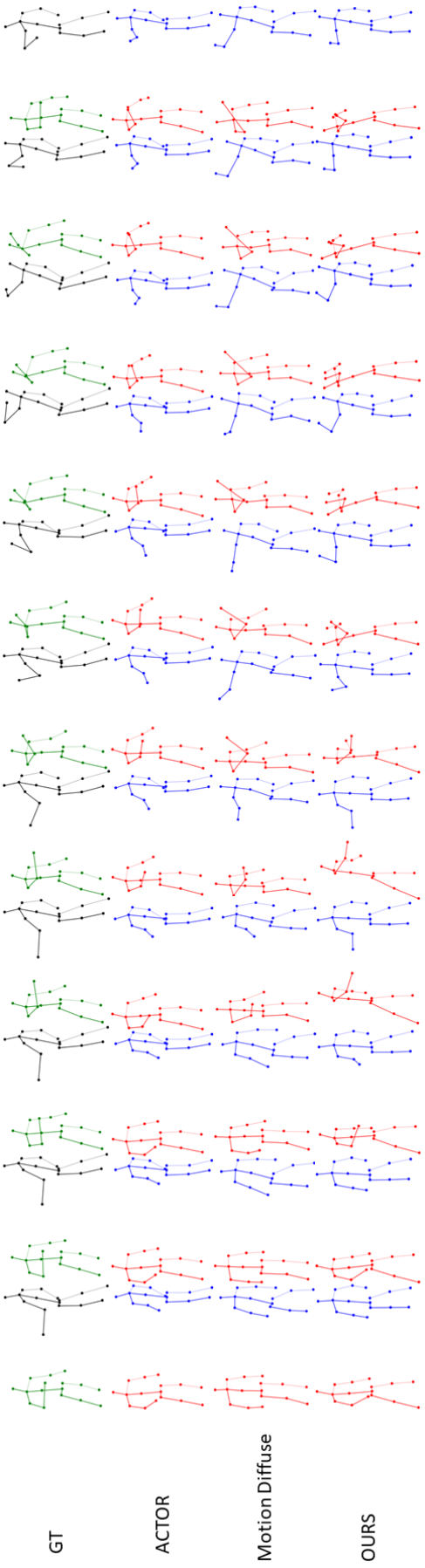


Figure 4.4: Examples of diverse motion generation for a given text prompt “Cheers and Drink” action from NTU-26.

other only achieves results slightly higher than chance (20%) we will see in the qualitative results that on DuetDance, ACTOR does not produce any motion, we will discuss this in detail in the qualitative results. In the “jive” class, all methods only achieve chance level or lower accuracy. But the results are not so low for the ground truth with means that all methods have trouble generating motion of the “jive class”. In Table 4.4, we show that we outperform the other methods on both metrics meaning that the results of our method are more realistic.

4.4.6 Qualitative Results

NTU-26. Figure 4.4, 4.5, and 4.6 shows visuals of sequences generated for the “Cheers and drinks”, “High-five", and “Kicking" classes respectively. The ‘Cheers and drinks” class of motion is more complex than others because it is composed of two separate motions “cheers” and “drink”. All methods generate a proper motion but ACTOR shows a low intensity for “cheers” and does not really generate the “drink” motion. MotionDiffuse generates a good motion with both “cheers” and “drink” but there is some noise and the arm length grows over time. Our method generates the proper motion with the two steps and does not produce the noise that is present in the ground truth. In our case, one character drinks while grabbing the glass with one hand while the other uses both hands showing the diversity in the generated motions. For “High-five" ACTOR also generate a low-intensity motion and both characters raise their hand but do not perform a high-five. Both MotionDiffuse and our method generate a high-five but MotionDiffuse shows noise and the hands of both characters stay far from each other. The ground truth once again contains noise that is not present in our generation. For the “Kicking" class ACTOR does not generate any motion for either character. MotionDiffuse generates the red character as being kicked but does not generate the blue person kicking. Our method on the other hand generates both the kicking motion and the other character being kicked like the ground truth. In the ground truth, we can see that the leg is never fully extended during the kick this is common for this class. The NTU-RGB+D dataset is captured using Kinect camera and has difficulties capturing the legs due to the positioning of the camera and occlusion during interactions. this shows again the kind of noise present in the original data. Overall we see that our motion is more realistic, temporally, and spatially coherent and manages well to keep the interaction coherent.

DuetDance. Figure 4.7 shows examples of motion generation of the “salsa” class. The dance motions are more complex and the sequences are longer than NTU-26 sequences. ACTOR does not produce motion. We believe this to be due to the great variability of motions from the same class. ACTOR converges to a mean and finds that an unmoving pair of skeletons is the best generation for its losses. We see that MotionDiffuse produces a dance motion without noise. This is because there is less noise in DuetDance than in NTU-26. Our method also generates a dance motion but is better than MotionDiffuse, we reproduce the motion of characters changing sides that is present in the ground truth and the interaction is better as the arm of both characters does not overlap.

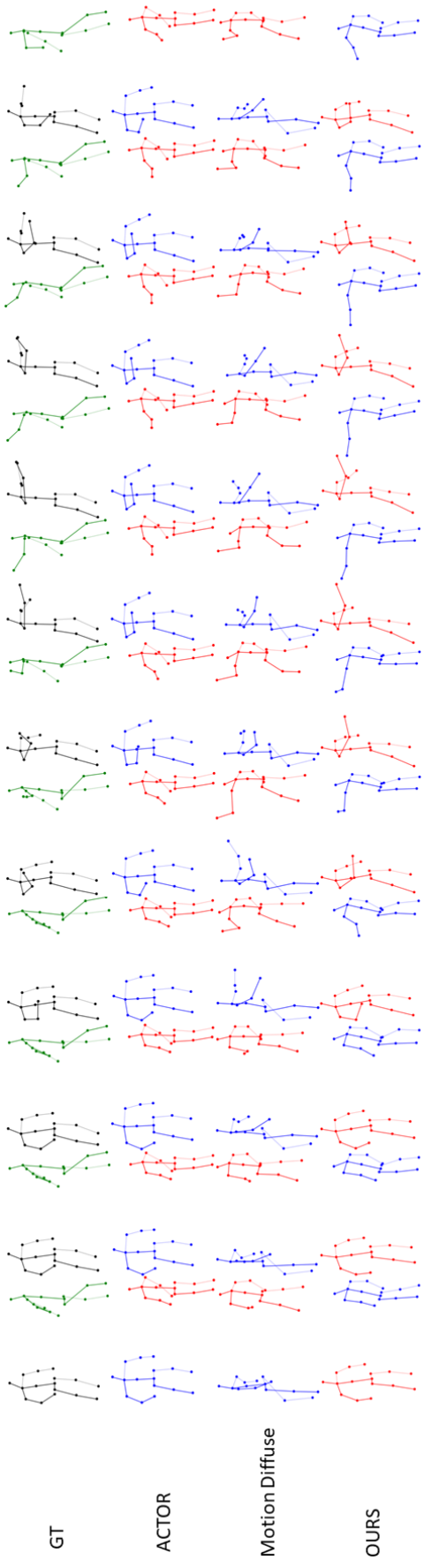


Figure 4.5: Examples of diverse motion generation for a given text prompt “High-five" action from NTU-26.

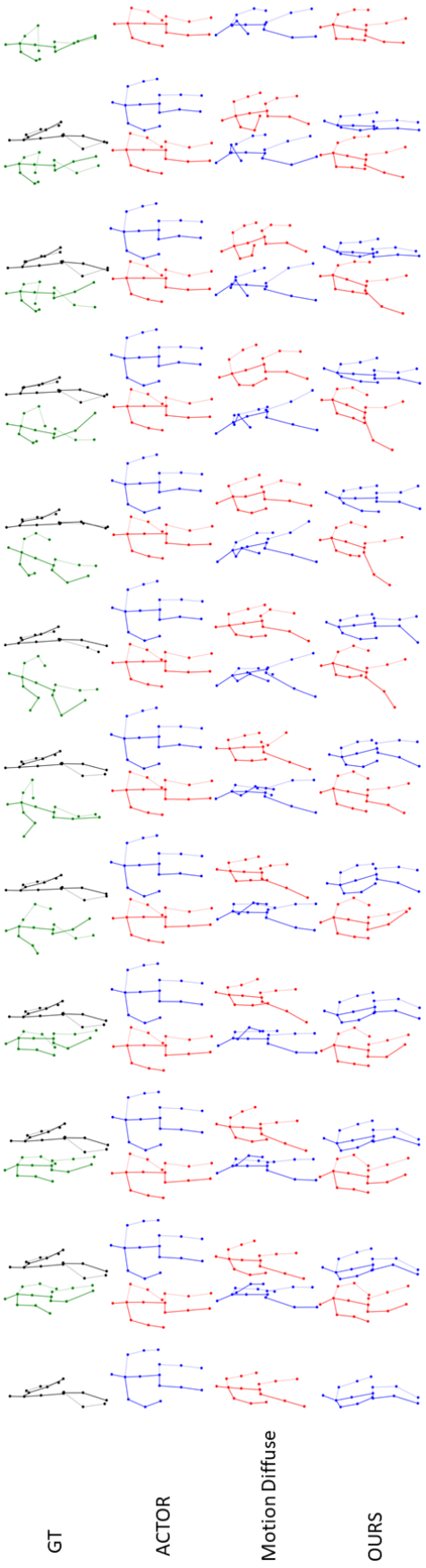


Figure 4.6: Examples of diverse motion generation for a given text prompt “Kicking" action from NTU-26.

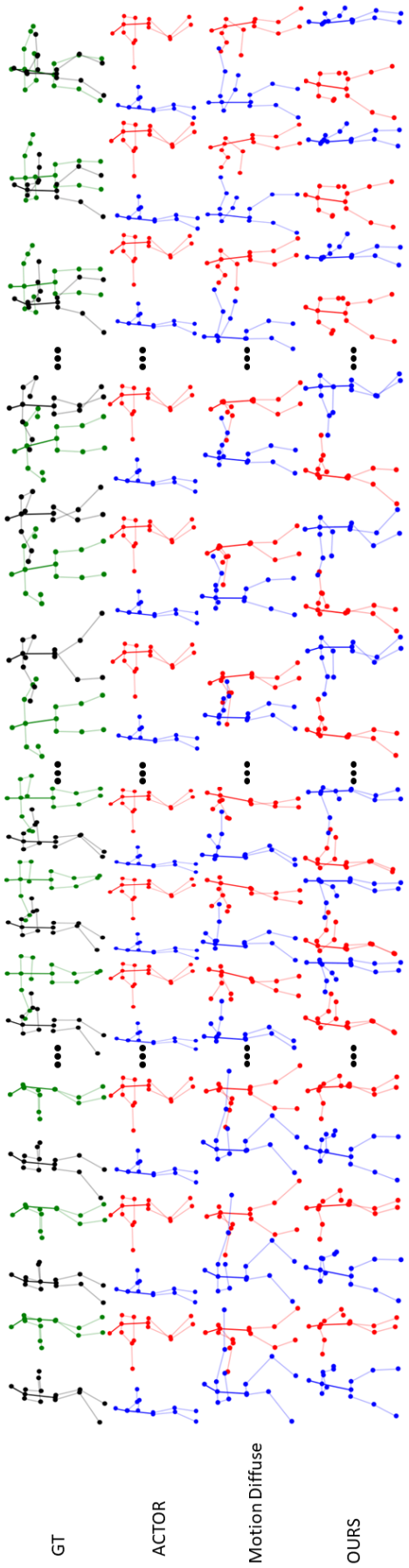


Figure 4.7: Examples of diverse motion generation for a given text prompt "Salsa" action from DuetDance.

Table 4.5: User study results (%).

Method	NTU-26		DuetDance	
	Q1	Q2	Q1	Q2
ACTOR [118]	6.1	7.8	5.6	5.9
MotionDiffuse [165]	22.4	24.3	21.8	23.7
BiGraphDiff	**31.6**	**32.7**	**28.5**	**30.1**
GT	39.9	35.2	44.1	40.3

4.4.7 User Study

The user study compared BiGraphDiff with two leading methods (i.e., ACTOR [118], MotionDiffuse [165]) and the ground truth sequence. For both datasets, we randomly select 20 samples for each class from the test data. For each comparison, 30 participants are asked to answer two questions, i.e., 'Q1: Which skeleton sequence is more realistic?', and 'Q2: Which skeleton sequence matches the input text better?'. The numbers indicate the preference percentage of users who favor the results of the corresponding methods or the GT skeleton sequence. The results highlight the quality of the sequence generated by our method.

4.4.8 Ablation Study

We report ablation results in Table 4.6 on the NTU-26 dataset. We compare a simple two-stream Transformer (S1), a two-stream Transformer in a diffusion process (S2), a two-stream Transformer in a diffusion process with a simple GCN (S3), and finally our method with bipartite graphs (S4).

The results of S1 are extremely bad. It is explained by the fact the Transformer is a deterministic method and has a low generation diversity which explains the very high FVD. Furthermore, the noisy data from the NTU dataset makes it even harder to provide well-generated sequences. S2 provides much better results both in classification accuracy and FVD, the results are similar to the results obtained by MotionDiffuse. With S3 the simple GCN helps enhance the generation leading to better accuracy and FVD. This highlights the ability of the GCN to model more accurately the spatio-temporal dependencies from each skeleton. Adding a bipartite graph network in S4 provides a stronger increase in performance. It shows that modeling the interactions between the two skeletons is more important than trying to refine the interactions inside each skeleton like S3 did. It validates the use of the bipartite graph network in BiGraphDiff architecture.

4.5 Very Long Generation

Long-term motion generation plays an important role in real-world applications. Our method is able to generate longer sequences as shown in Figure 4.8. We train the network on the

Table 4.6: Ablation study on NTU-26.

Method	Classification↑	FVD↓
S1: Two Stream Transformer	3.9%	21215.21
S2: Two Stream Transformer + Diffusion	69.3%	1406.09
S3: S2 + Simple GCN	73.2%	1123.88
S4: S2 + Bipartite Graph	**77.0%**	**1048.16**

original DuetDance dataset with a maximum sequence length of 4050 frames. We use 376 samples for training and 40 (8 per class) for testing. Figure 4.8 shows an example of 1580 frames from the "rumba" class. We can see that we generate dance-like motion for the entire duration of the sequence. However, it is very noticeable that we generate better motion for the first few hundred frames, we see that the motion quality around 300 frames is good but then around 600 frames we see deterioration that gradually becomes worse. This is due to the length of the sequences in the DuetDance dataset distribution which are usually not very long (average: 483 frames, median: 360 frames).

4.6 Conclusion, Limitations and Future Work

We introduce the first approach for 3D human motion interactions based on denoising diffusion models. Both quantitative and qualitative evaluations show that BiGraphDiff outperforms state-of-the-art methods. The proposed BiGraphDiff method generates coherent human motion sequences that are longer and more diverse than the results of previous approaches.

The proposed BiGraphDiff suffers however from the common limitations of diffusion models: the need for large datasets and the long training and testing duration. The method is also still slightly sensitive to noise in the training data and can sometimes generate deformed skeletons. This is due in part to the quality of the data used but also because we do not set any constraint related to the input data, e.g., bone length or relative position of joint for 3D skeletons. This means that BiGraphDiff can be used for tasks other than human interaction generation. As long as the input data can be split into two sets and has a temporal or positional component BiGraphDiff can be used for generation. Diffusion models are improving very quickly and some of these improvements could be used to further improve our results. To reduce the training time, early stop diffusion could be used [101]. To improve the performance of graphs in our model we could leverage diffusion models specifically designed to use graphs [48].

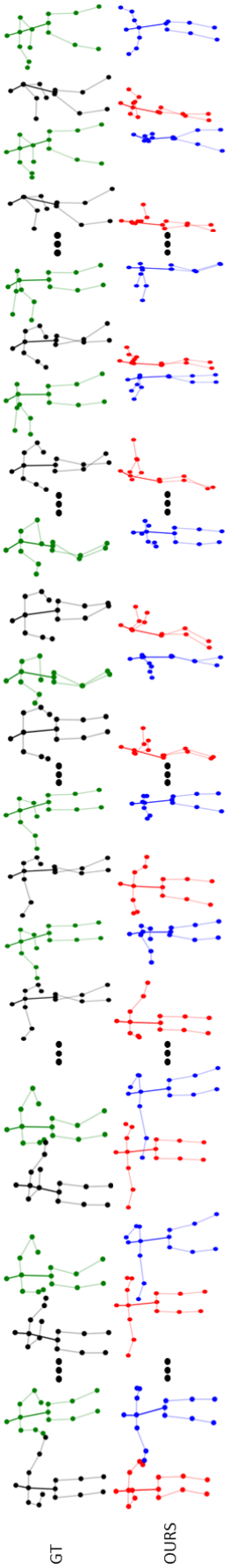


Figure 4.8: Example of generation of very long sequences on the "rumba" class from DuetDance. Under each skeleton the frame number.

Part V

Application to an interactive agent in virtual reality

CHAPTER 5

APPLICATION TO AN INTERACTIVE AGENT IN VIRTUAL REALITY

Interaction with virtual agents has been studied through several modalities: expression recognition and generation [39], action recognition and interaction with objects [36], multimodal input and output [39]. Some methods can even be used for both virtual agents and robots[62]. However, the challenge of building an interactive virtual agent differs from that of building an interactive robot. Robots deal with real persons and objects and do not need to have a human appearance, the focus is more on the accuracy of their motion than on their realness. Virtual agents on the other hand look very human and increasingly so, as 3D modeling software allows for more realistic virtual humans. They lack the mechanical constraints of robots and their behavior can be more realistic than that of robots. In this thesis, we build a virtual agent that reacts to communicative gestures using non-intrusive data acquisition devices. We want the behavior of our virtual agent to be as realistic as possible.

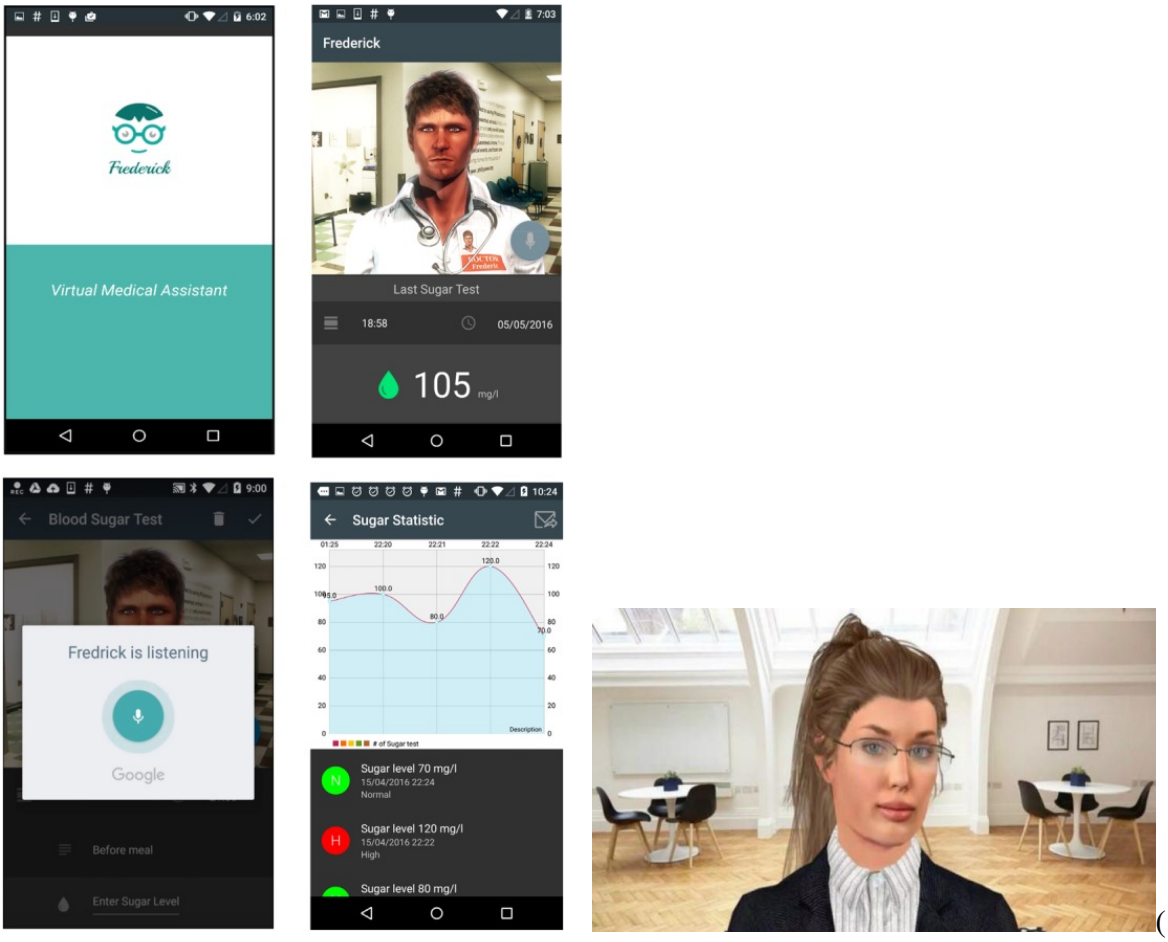
5.1 Virtual Agent Application

Virtual agents have been increasingly used for many purposes, especially with the recent growth of virtual reality applications. We will describe some possible uses of virtual avatars in several domains.

Healthcare. Virtual agents can have a wide range of uses in healthcare. For example, they can be used to help patients feel more comfortable during telehealth sessions by giving practitioners more presence. Virtual agents can be part of health surveillance applications to ensure that people follow their treatments (Fig.5.1a). Or they can be used to help with therapy for example by using virtual agents to "train" people with communication disorders to have a better understanding of human communication. This could help people improve their social skills and reduce the inconvenience that their disability can cause in everyday life. Furthermore, virtual agents could be used as companions for long stays in hospitals. We can also reverse the setting and use an avatar to train medical students, in this case the avatar will play the role of a patient.

Virtual reality environment. Entertainment is another obvious use for virtual agents, especially with virtual reality. In these virtual worlds, the quality of graphics, interactions with the environment, and sound design has continuously increased. Virtual agents on the other hand, despite having been part of the experience for a long time, still suffer from very limited interactions with the users that make them feel artificial. For example, many modern virtual agents allow communication by speech, much less by facial expressions, and even less by bodily gestures although the latter two are also very important means of communication for humans. Having more realistic avatars that can use several channels to communicate with users would enhance the quality of the experience in this virtual world. With the recent rise in popularity of metaverses, keeping users interested in the virtual world, the issue of creating realistic interaction with virtual agents will become even more important and so will providing avatars capable of communicating through multiple modalities.

Virtual assistants. Some customer services have already been replaced by simple artifi-



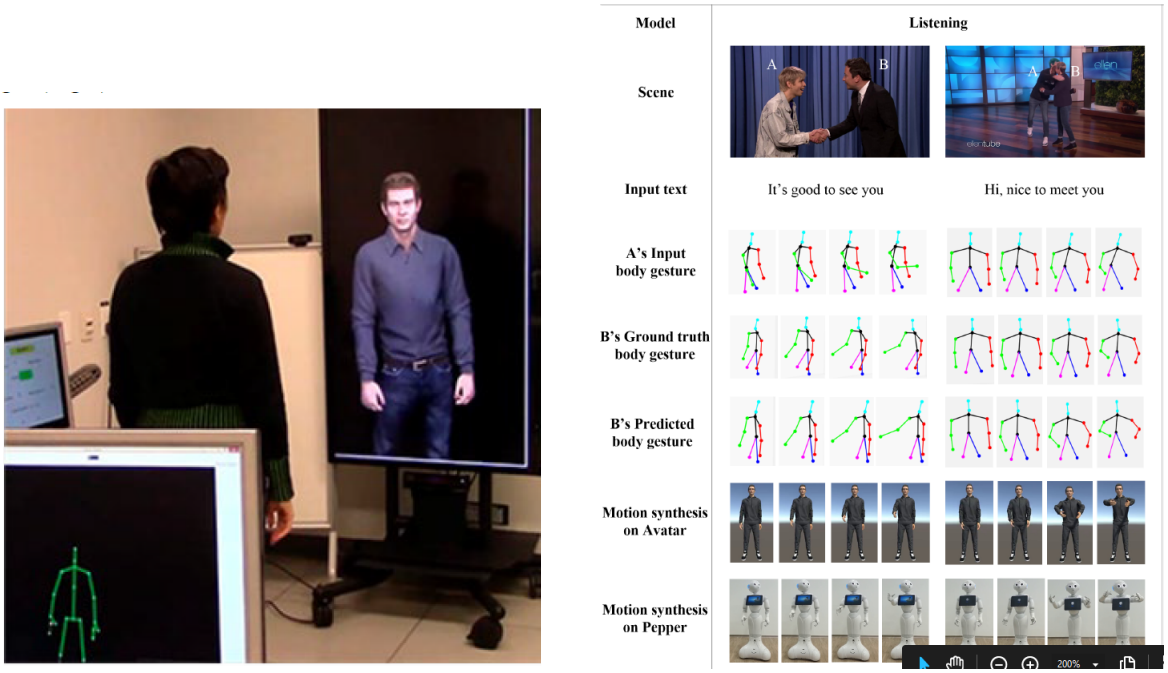
(a) Frederick, an assistant to help people with diabetes

(b) Vera a virtual recruiter which conduct interviews for companies

(a) source: [110] https://www.youtube.com/watch?v=foQIVavHA4Y.
(b) source: https://www.jobboardfinder.com/news/robot-vera/

Figure 5.1: Examples of avatar uses

cial intelligence *e.g* after-sales services replaced by chatbots for simple issues. As these may seem impersonal and people might lose patience faster than with a real person, the obvious enhancement will be to replace the chatbot with an avatar. An avatar has the advantage of looking like a human. If its behavior is realistic enough, the customer will consider it more like a person and be much more patient when interacting with it and explaining problems. In some cases, humans and less immersive artificial intelligence have already been replaced by avatars. For example, recruitment AI is given a human appearance to make the interview feel more natural (Fig.5.1b). However, like in virtual environments they often only use the speech modality sometimes coupled with facial expression recognition [39].



(a) Multimodal avatar by [39] that use many modalities to communicate with the user: object detection, facial expression recognition, skin conductance...

(b) [62] Method allow to generate motion in an interaction both for a virtual agent and a robot

Figure 5.2: Existing interactive avatars

5.2 Virtual Agent Application Requirements

We want to build an interactive virtual agent that reacts to a person's gestures. It is an application to our work on human motion prediction. We apply several additional constraints to the virtual agent as well as the environment in which it evolves to improve the realism of the interaction:

1. Realistic avatar appearance: we want the avatar to look like a human, this means that it must be detailed and textured enough so that users may interact with it like they would with another human.

2. Believable setting: the environment in which the avatar evolves must also be realistic and the setting linking the avatar to the environment must be believable *e.g* we do not want the avatar to stand still while waiting for the user to make gestures.

3. Non-intrusive data collection: we need to collect data from the user (motion, facial expression...) but we do not want to use captors that the user needs to put on the body (*e.g* skin conductance captors, motion capture marker...). We wish to only use cameras (video, depth...) to build an environment that the user can simply enter and instantly start to interact with the virtual agent.

4. Related and accurate reaction: the avatar must react in a way that its reaction is coherent with the gesture performed by the user (*e.g* the gesture for being cold causes

the avatar to close a window). Furthermore, the reaction motion must be realistic and smooth to give the avatar a more human behavior.

5. Timing of the reaction motion: to look realistic the timing at which the motion is performed is important. In particular, the reaction must be performed quickly enough so that the user does not feel as if the avatar did not understand the motion.

5.3 Architecture Choices

To respect the constraints we set for our virtual agent application, we must first be able to react quickly to the user gesture. This implies that, in a short time, we must understand the motion the user is performing and produce a reaction to this motion. In these conditions, waiting for the user to complete the motion is not ideal. Therefore, we choose to start by predicting the end of the user motion based on its start. Then we classify the predicted motion and send the information to the avatar that will react accordingly by selecting the corresponding prerecorded reaction motion. The result of the classification will let the system decide which reaction to perform. However, as we will see in Section.5.4, all the gestures we choose indicate discomfort for the user. Depending on the degree of said discomfort, a reaction might not be needed. To take this possibility into account, we add a facial expression recognition module to decide if the virtual agent has to react. Fig.5.3 shows the overall architecture of the application.

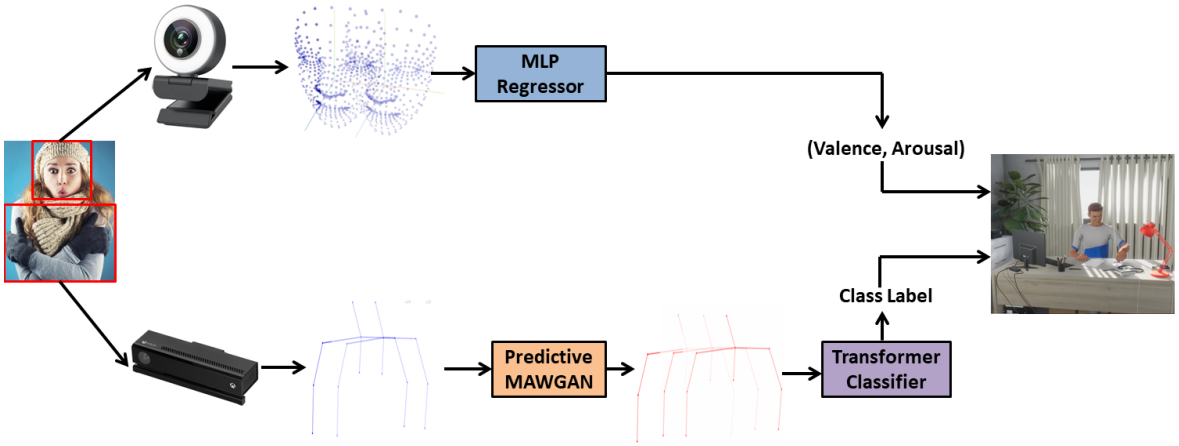


Figure 5.3: Overview of the virtual agent application architecture

One last issue to solve is the need for data. While facial expression databases are plenty and we do not need data to generate the reaction from the virtual agent, we still need data to train networks to predict and recognize the gesture of the user.

5.4 A New Motion Database

There is currently no database containing the kind of communicative gestures that we are looking for when interacting with the avatar (e.g “I’m cold”, “My ears hurt”...). Most motion

Scenario	Motions	number of samples MoCap	number of samples Kinect
I am cold	rub both arms with hands	115	109
I am hot	wave one or both hands near the neck/ mouth	115	115
This is too loud	cover both ears with hands	113	99
This smells bad	pinch nose with one hand or cover mouth and nose with both hands	115	110
It is too bright	cover eyes with one or both hands	111	109
Hello	wave with one or both hands	100	121
Idle	random and meaningless motions	59	65
Total		728	728

Table 5.1: Content of the gesture dataset

databases contain motions that represent action performed alone (walking, kicking...) or in interaction with other persons (exchanging an item, fighting...). In the latter, the interactions require physical contact between the two parties, either directly or through an object. Gesture databases are also available, however, the gestures included are gestures that do not require a reaction from another person. Furthermore, the gesture databases that could be used to make another person react are culturally coded, the motion performed might not have the same meaning for a French person or even have no meaning at all. To get data that we could use in our scenarios we build a database of communicative motions.

We show in Table 5.1 the different scenarios and motions used in our database as well as a description and the number of samples recorded. Since different motions can be used for each of the scenarios given to the participants, they were also instructed to avoid certain types of motion (*e.g* they were not allowed to pull the collar of their shirt during the "I am hot" scenario). We added an additional class of motions containing random idle motions to train our networks. It is used to recognize when no motion of particular significance is being performed. By doing that, we can avoid the misclassification of random motions as meaningful ones leading to an unwanted reaction from the avatar. The scenarios were selected from situations where people might perform a gesture to indicate their mental state but that can also be performed to elicit a response from another person (e.g. the gesture of “coming here” oblige observers to modify their behavior. In this case, people move toward the executor). Mental state gestures are gestures that communicate a feeling (e.g. I am cold). In all our scenarios the motion indicates a discomfort and the avatar must react by helping the user.

The motion capture was performed at the Equipex Continuum using the facilities of the Fédération de Recherche "Sciences et Cultures du visuel" at Tourcoing, France. We recruited 11 healthy volunteer participants among the students of the lab team (8 males, 3 females; age between 20-60 yrs). Participants did not receive financial compensation for their participation in the study. Participants were asked to stand in front of the Kinect at a place marked by a cross on the ground. Fig.5.6 shows the view participants had. Their initial position was then adjusted depending on their height (to allow the capture of all markers by the motion capture system). They were asked to perform the gestures from Table.5.1. The motions from class “Idle” were only performed by 2 participants as they are random, meaningless, motions. The participants performed each motion at least 10 times. Each participant spent between 30 and 45 minutes to complete the entire recording session. The motions are captured on two different systems. The first is a high-precision motion capture system with 6 Qualisys motion capture cameras and nine markers on the participant’s body: one on the forehead between the eyebrows, one at the base of the neck, and one above the navel, for each arm we have

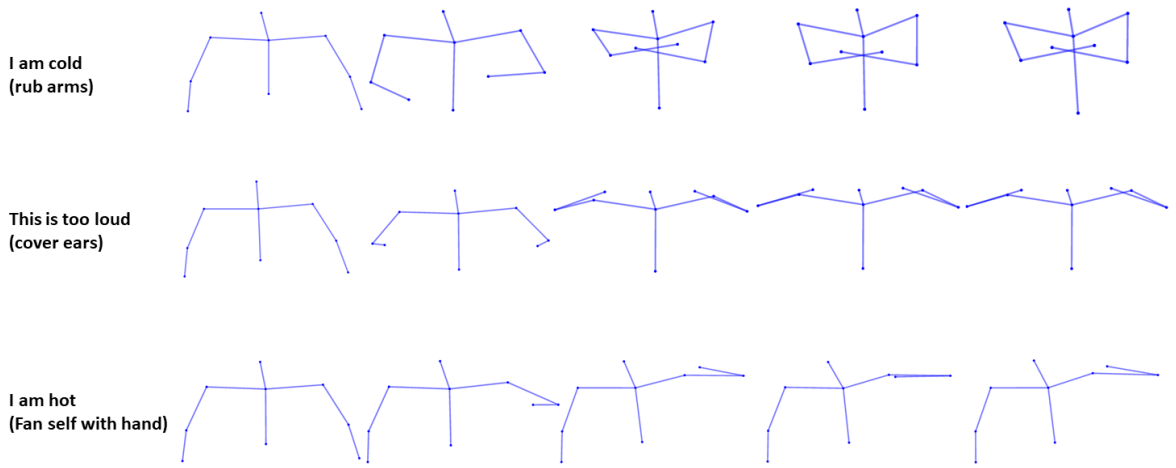


Figure 5.4: Examples of motion from the new database (motion capture). We show three different classes. Visualization obtained after motion sequences were cut and resampled to have a duration of $3s$ (see Sec.5.5.2)

markers on the top of the shoulder, on the elbow and on the outer side of the wrist. Fig.5.5 shows the position of the markers on a participant. The motion capture system is calibrated to have an average positional error of less than 2 millimeters. Each sample is recorded for exactly 5 seconds regardless of the actual duration of the motion at 100 fps.

The second capture system we use consists of a single Kinect V2 camera. We decided to also capture the motion with a Kinect camera since the motion capture system would give us extremely precise data that we will not be able to obtain with the system we have put in place for our application. The Kinect allows us to capture real-world data of the motion present in the motion capture database which will help our models become more resilient to the noise there is in real-world data. With the Kinect, we capture joint positions of the entire body but then only keep the nine joints corresponding to those from the Motion capture. Due to the difficulty of setting up synchronous capture between the Kinect and the motion capture system, the capture from the Kinect is started and stopped by an experimenter following the instruction from the experimenter working with the motion capture system. The capture setup is illustrated in Fig.5.6. Due to this, the duration of the Kinect capture varies between 73 and 530 frames with a median of 155 frames and a mean of 157 frames with a framerate of 30 fps which means that on average the motion duration average is close to 5s. Samples that were too noisy or where joint positions were lost for too long were removed. When a joint position was lost for a short amount of time we estimate its position to be the last known position. All recorded samples were checked by a human curator before being used in the database. This amount to 728 motion samples for the MoCap data and 728 samples for the Kinect data.

Figure 5.5: Motion capture markers position

Figure 5.6: Motion capture and Kinect setup. Point of view of the captured subject.

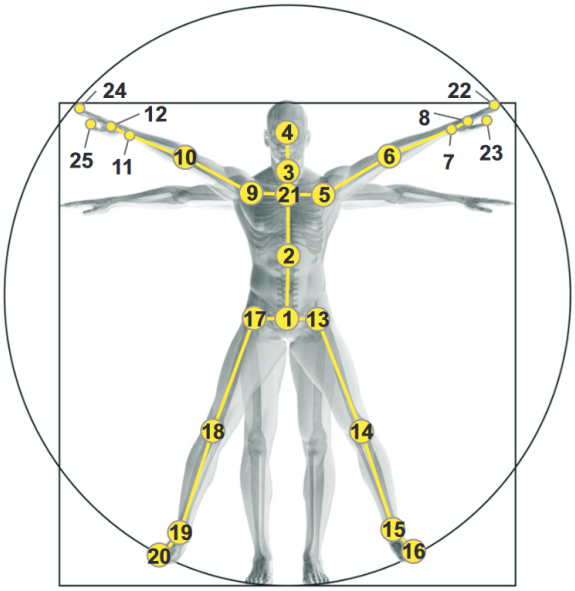
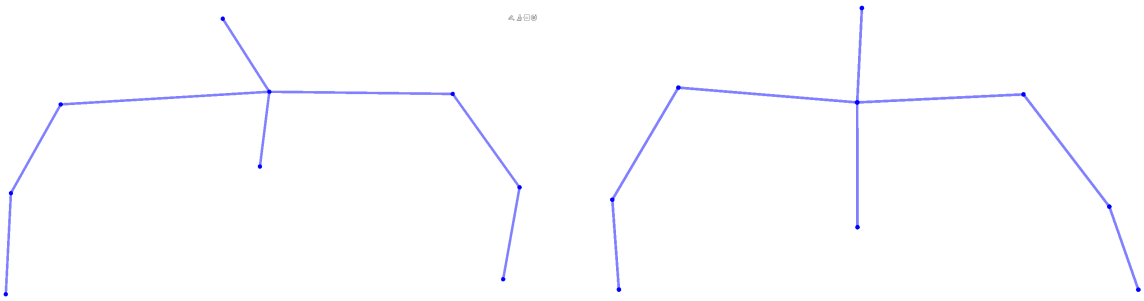


Figure 5.7: Figure from [94] showing the joints captured by the Kinect camera



(a) Kinect (b) Motion capture

Figure 5.8: Captured skeletons example

5.5 Virtual Agent Application

5.5.1 Data Capture

To avoid having to wear captors when using the application, we can capture the motion data using depth cameras or RGB cameras. Both allow for the capture of skeletons through different toolkits. For example, OpenPose [21] for RGB camera or the Kinect SDK for the depth cameras. We tried a RGB camera with BlazePose[14], Kinect depth camera, and Intel real sense depth camera with NuiTrack SDK[1]. The result led us to choose the Kinect camera. Blazepose, while good for 2D joint detection, fails to correctly estimate the depth coordinates and causes the recorded data to be noisy. The intel real sense with NuiTrack often fails completely to detect the joint position of the arm and the recorded motion looks

[1] https://nuitrack.com/

nothing like the actual motion. On the other hand, Kinect gives an accurate recording of the motion even if it is quite sensitive to occlusion. Fig.5.9 illustrate the results of each method.

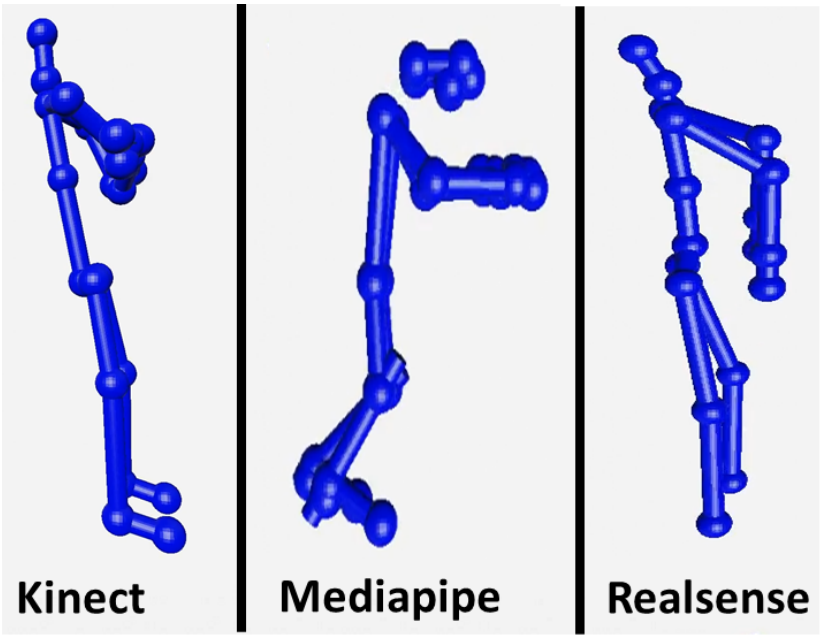


Figure 5.9: Comparison of different 3D skeleton capture methods for an arm rubing motion

For the face data, we use a RGB camera with facial landmark detection software. Out of the many libraries that exist (OpenFace [2], dlib[3]...) we choose to use Mediapipe FaceMesh[4] which uses a facial detector [13] coupled with a face mesh predictor [73]. We choose to use this method due to the speed and light weight of the model but also because of its accuracy and higher resistance to occlusion. Even if it uses a face mesh instead of proper landmarks, the results on expression recognition are similar to other landmarks detectors. Fig.5.10 shows the facial mesh from Mediapipe.

5.5.2 Action Prediction Module

To predict the motion of the user we simply use the Predictive Manifold aware GAN from chapter 2. This means that we must encode the motion data as SRVF before sending it to the prediction network and then retrieve skeleton data from the predicted SRVF which, as shown in Chapter 2 can easily be done. We have also shown that the prediction speed is fast enough to be used in real time which is mandatory to be used in our application. In chapter 2 we predicted the same number of frames as what we were given as input. Here, however, we take an input of 25 frames ($1s$) to predict 50 frames ($2s$). The Predictive Network is trained on our dataset, where we mix the Kinect and Mocap data. The motion sequences were cut and resampled to have a duration of $3s$.

[2] https://github.com/TadasBaltrusaitis/OpenFace
[3] http://dlib.net/
[4] https://google.github.io/mediapipe/solutions/face_mesh

Figure 5.10: Figure from [73] showing the face mesh from MediaPipe

5.5.3 Action Recognition Module

To classify the predicted motions we use a Transformer encoder that uses spatial attention and the skeleton adjacency module presented in Chapter.3. The Transfomer encoder is followed by a simple multi-layer perceptron to classify the motions. Fig.5.11 shows the architecture of the action recognition network. We then train the network on $3s$ of motion composed of $1s$ of real data and the $2s$ of motion predicted by the prediction module. We train with predicted data instead of real data because the recognition module will always receive predicted data. This module estimates the class of the concatenation of the recorded motion and the predicted motion. Table.5.2 shows the accuracy of the network on our dataset. For this experiment, we split the dataset between train and test. Out of the 11 subjects we randomly choose subject 2 and subject 7 to be the test subjects. For class "Idle", since the samples are not connected to a particular subject we randomly take 11 samples from the Kinect data and 11 samples from the Mocap data to create the test set. With this, we have 1192 training samples and 264 test samples. We compare the result of $3s$ of captured data, $1s$ of captured and $1s$ of captured data completed by $2s$ of predicted data. For each modality, we train the network on the corresponding data. As for the prediction network we use both the Mocap and Kinect data. We see that using $3s$ of data the network correctly classifies 63.6% of action. Some actions of our dataset like "Hello" and "I am hot" (see Table 5.1 for the detailed motions) are similar and often confused which can explain the relatively low score. If we use only $1s$ of data we see a decrease in performance down to 47.7% accuracy. If we use our prediction network to predict $2s$ of motion and combine it with the $1s$ of input data to get a $3s$ sequence we increase the results to 60.2% close to the original 63.6%. This highlights the advantage of using our prediction network. We can are able to classify motion with only one-third of the data nearly as accurately. In practice this allows the avatar to react thrice as fast since the prediction process only takes a few milliseconds.

Table 5.2: Recognition accuracy on our dataset. Comparison between $1s$ of motion captured by the Kinect, $3s$ of motion captured by the Kinect, and $1s$ of motion captured by the Kinect concatenated to $2s$ of predicted motion

	Capture $1s$	Capture $3s$	Capture $1s$ + Prediction $2s$
Accuracy	47.7%	63.6%	60.2%

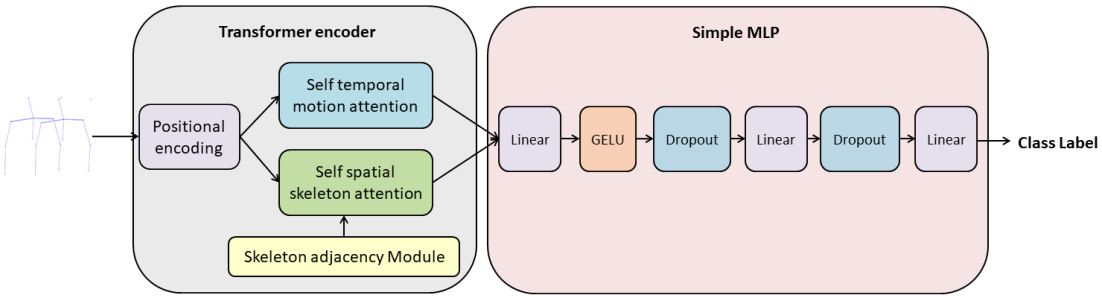


Figure 5.11: Architecture of our Interformer-based motion classifier

5.5.4 Facial Expression Recognition Module

To avoid reacting to random motion or communicative motion that does not require a reaction from the avatar, we add a facial expression recognition module that estimates the valence and arousal value of the user expression. We use a simple MLP consisting of 4 Dense layers with dropout that we train on the AFEW-VA[81] dataset using landmark data captured with Mediapipe. While the network might seem extremely simple, it is efficient. On the AFEW-VA dataset, with valence and arousal values comprised between -10 and 10, we obtain an error of only 1. State of the art method might perform even better but since our goal is only to set a threshold for discomfort while being fast we do not need as much accuracy as the state-of-the-art methods. We want to prevent the avatar from reacting when the user is in a neutral or positive mood since the motion performed might not indicate discomfort in that case. To do so we ignore motions performed while $valence > 0$ as it correlates with positive feelings. To avoid reacting when the user is in a neutral mood, we decide to only consider motion performed while $valence < -3$ and $|arousal| > 3$. We capture the face of the user with a second camera independent from the Kinect. The Kinect must be far enough to see the entire upper body and can not be simultaneously used to capture a detailed face.

5.5.5 Avatar Module

Our avatar module is a Unity application where our avatar evolves in a scene consisting of a room with furniture as shown in Figure 5.12. The room contains a desk behind which the avatar sits, the avatar can interact with several objects in the scene: cigarette, window, curtains, HiFi system, and a keyboard. We choose a simple appearance for the avatar with a tee shirt so that users will not focus on its appearance but rather on its reaction. When reacting to the user gesture the avatar will stand up (except for the"This smells bad" scenario)

Scenario	Motions
I am cold	If the window is open: stand up, close the window, sit down.
I am hot	If the window is closed: stand up, open the window, sit down.
This is too loud	if the volume is high: stand up, turn the volume down on the hi-fi, sit down system
This smells bad	if the cigarette is lit: put out cigarette
It is too bright	if the curtain is open: stand up, close the curtain, sit down
Hello	Stop idle motion and turn to face and look at the user
Before turning attention to the user	look at the computer screen while typing on the keyboard. Sometimes turns its head to look at the notebook.
Attention state	look at the user and sometimes at objects on the desk.

Table 5.3: Avatar reaction to the gestures described in Table 5.1

and perform an action behind the desk (See Table 5.3 for a complete description of each reaction). This module can also work in virtual reality and the entire room is modelized in unity as seen in Figure 5.13. When using the virtual reality version, since the user needs to wear a virtual reality helmet we can not use facial expression recognition. In that case, the output of the module is ignored.

Figure 5.12: Our avatar in its resting state.

5.5.6 Full Application

The full application combines the aforementioned modules. The motion data is captured with a Kinect V2 camera and the face data with any RGB camera such as a webcam. From the motion data from the Kinect, we keep only the 3D skeleton data that we send to our prediction network every 25 frames. In the prediction network, the data will first be normalized and transformed into a SRVF. Then we predict the 50 future frames of the motion in SRVF format and rebuild the corresponding skeleton sequence. The predicted sequence is concatenated with the sequence recorded by the Kinect for a total length of 75 frames. This concatenated sequence is fed to the action recognition network which outputs a class label. Simultaneously 25 frames of the face data are sent to the facial expression recognition module where we extract the facial landmark from the RGB data with MediaPipe's face mesh. The landmarks

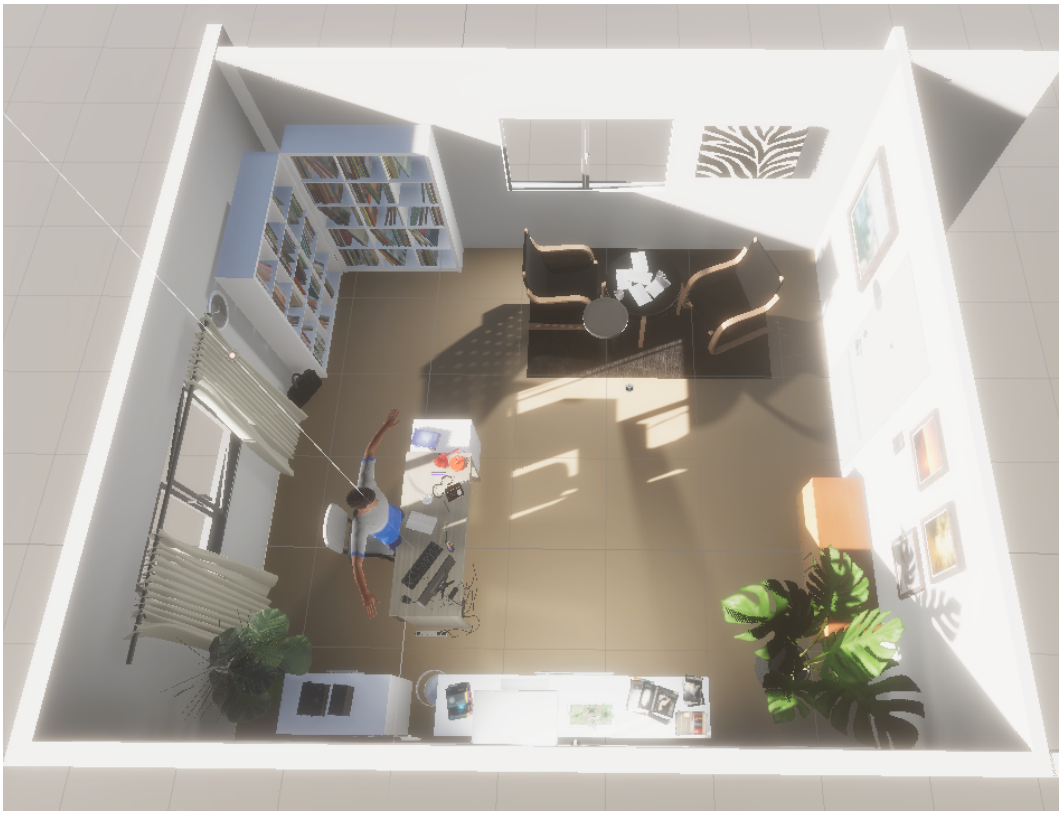


Figure 5.13: Aerial view of the scene in the unity builder (ceiling is removed). The entire room is modelized for virtual reality.

are treated by the network which outputs arousal and valence values. Our scenarios are all negative ones, the user is bothered by something and wants the avatar to do something about it. We choose valence and arousal intervals that contain expressions that users may perform in these scenarios. If the values obtained by the network are outside these intervals we consider that the motion performed by the user does not show a discomfort needing a reaction or that it is unrelated to the scenarios. In these cases, we prevent the avatar from reacting. Depending on the output of the motion recognition and of the facial expression recognition network the application will decide if the avatar needs to react and if so start the appropriate motion. If no reaction motion is required because the predicted class is 'Idle' or due to the valence and arousal values, the avatar stays in its 'Attention' state. This cycle is repeated until the application is closed. When using virtual reality users can move the camera with their heads and can move around in the room. However, the position of the avatar remains unchanged and a change in user position too drastic will interfere with the body data capture from the Kinect. Figure. 5.3 describes the full application.

Part VI

Conclusion

CHAPTER 6

CONCLUSION

6.1 Summary of Contributions

The goal of this thesis was to develop novel generative models for the generation of human motion. We present a new human motion prediction method in which we represent the motion as a trajectory. Our method outperforms the state of the art on two commonly used datasets, especially for long-term prediction. We propose a Transformer architecture to generate a human reaction motion conditioned on an action motion. To the best of our knowledge, this work is the first to investigate this challenge. We outperform state-of-the-art methods for two-person interaction prediction and single-person motion prediction. We show our ability to generate longer and more complex reactions on a dance dataset. We propose a diffusion two stream Transformer with bipartite graphs to generate two-person interaction motions. We outperform state-of-the-art methods for motion generation on simple interaction and complex two-person dance motion. We show an application of our human motion prediction method by developing an interactive virtual agent that reacts to user motions. To ensure that the system works in real-time, we predict the end of the user's motion before classifying it to select the proper reaction for the agent. To train the networks used with our application, we build a new 3D motion database with 7 classes and 1456 samples captured either by a motion capture system or by a Kinect camera. Taken together, the entire work carried out in this thesis shows several methods to deal with human motion generation and their possible uses in real-life applications. Nevertheless, the proposed approaches still suffer from some limitations.

6.2 Future works

6.2.1 Avatar Interaction

The virtual agent in its current form, while performing adequately for the specifications set, suffers from several limitations that reduce its realism. For example, when using a virtual headset we can not use the facial expression recognition module. Some headsets however contain an eye-tracking device. In future works, we could use eye tracking or gaze detection to alleviate the lack of facial expression recognition, for example, by having the avatar reacts only when the gaze of the user is directed toward it. Eye-tracking uses specific hardware to follow the motion of the eyes while gaze detection uses RGB or infrared cameras to guess what people are looking at. Some methods are specifically designed to work with virtual reality helmets [24]. Fig.6.1 shows an example of an existing gaze detection method designed for virtual reality headsets. We also would like to permit more varied interactions with the virtual agent. To do this, we need a larger database with more types of motion from interactive scenarios. Adding new scenarios also means adding more reaction motion from the avatar which would require making new handmade reaction animations. In our virtual agent application, the reaction motions are prerecorded. While this allows us to have very detailed animations we have no diversity in the reaction motions. Human motion generation can increase the diversity of the avatar's motions and give and make the behavior of the avatar more realistic. Using SMPL 3D model [97, 116] we could have an agent that uses generated motion with good quality. Fig.6.2 shows an example of a body generated with SMPL. Also,

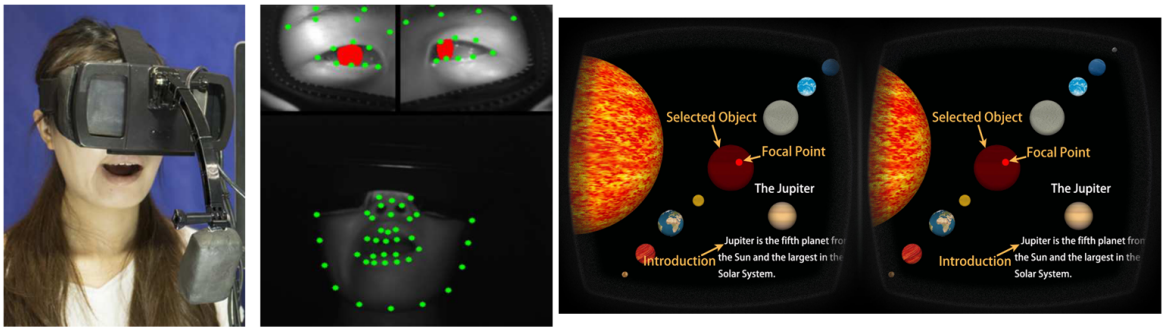


Figure 6.1: Figure from [24]. Example of gaze detection method for virtual reality headsets. **Left:** this method detects the gaze of the user with an infrared camera. **Right:** the method uses the gaze to describe the planet the user is looking at.

the avatar reactions sometimes involve grabbing an object *e.g* putting out a cigarette. These motions require a modelization of the fingers and of the object being manipulated. While more recent methods allow for the generation of the hand volume [116], we would need to generate the finger motions in addition to the rest of the skeleton.

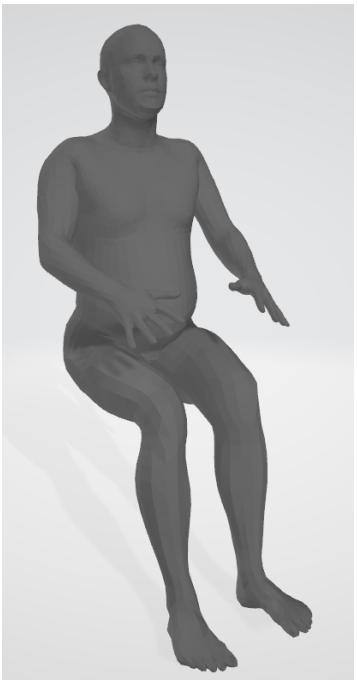


Figure 6.2: Example of volume generated by SMPL [97] of a person sitting.

The virtual agent we use also has a limited number of facial expressions. Recent methods are able to generate high-quality sequences of facial meshes [113]. Such methods could be used to generate the 3D facial expression of the virtual agent during the interaction. We could also model the user's facial expression more accurately. For example, the method for gaze detection shown in Figure.6.1 is also able to detect facial landmarks (green dots on the left picture). [24] uses the landmarks to generate the facial mesh of the user's face. Other methods such as [69] provide more realistic textured meshes that can be used to realistically

reproduce the user's face and facial expression in the virtual environment (Fig.6.3). This could be a way to further improve the experience of our virtual application. By giving the user a personal avatar to control we would increase the feeling that the virtual environment is real.

Figure 6.3: Figure from [69]. Example of 3D face generated from a person wearing a virtual headset using the method from [69].

6.2.2 Human motion prediction

In this thesis, we have focused on making a single prediction based only on the historical sequence. Recent works have explored the possibility of predicting several future motions for a single historical sequence [9, 102]. Predicting several future motions would let us choose the most suitable one. To predict very long motion and to manage changes in motion that are not predetermined by the historical motion we will need to consider additional conditioning inputs. When we move, we do so with a goal and are limited by our environment. These are the main things that we can take into account to enhance the performance of motion prediction methods. Some methods already tried to do this by using the trajectory of the character[159]. To provide better predictions and work on classes where the trajectory does not provide useful information, we need to take more information into account (furniture, wall, object, other people...). To provide more accurate predictions, some works have tried to take the environment into account [33].

However, there are very few datasets that contain information like the room layout that would help us to model the environment in which the character evolves. The existing datasets can still be used thanks to methods like [158] that generate an environment based on the motion of one or more 3D characters. To be able to extrapolate our motion prediction model to unseen classes we will need to build a method that looks at the local motion (body parts) and global motion (entire body). That way we hope that on unseen classes despite not knowing

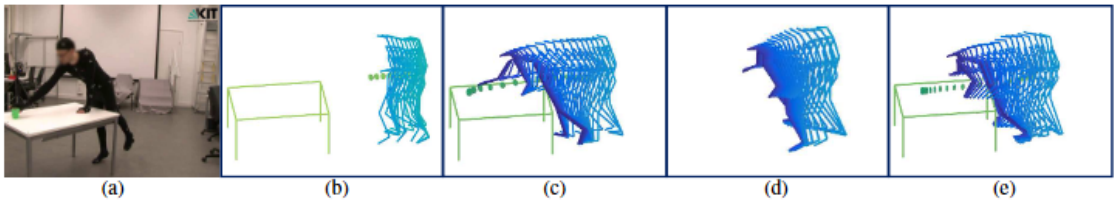


Figure 6.4: Figure from [33]. Example of motion prediction that takes the environment into account. (a) video from Human3.6M[64], (b) historical sequence, (c) future ground truth, (d) prediction of a standard prediction method, (e) prediction of [33]. In green are the table and the cup.

the global motion the model will be able to leverage its knowledge of local motion to predict an accurate motion.

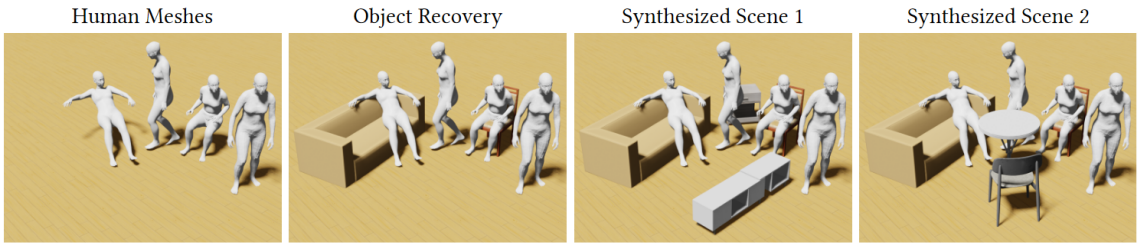


Figure 6.5: Figure from [158]. Environnement generation from human motion

6.2.3 Human Motion Generation

These improvements could also improve human reaction generation as adding information about the environment would lead to more realistic reactions. Additionally, we need to tackle the lack of diversity in our method. Our work on diffusion showed that including that coupling Interformer with a diffusion process could solve the diversity issue. Diffusion requires more data, and databases of annotated action-reaction couples are rare and usually small. We will need to build a database to help with the lack of complex interaction motion datasets. This database needs to contain both simple and complex interactions as well as long interaction motions less abstract than the dance motions from DuetDance. The interactions we consider are only between two persons. The CMU Panoptic dataset [67] is a dataset that contains interaction motion between more than two persons and could be used to generate the reaction of several persons based on the action of one (Fig.6.6). We could also try to generate the reaction of one person based on the action of multiple persons or the reaction of n persons based on the action of m persons. Future works on human interaction generation could also benefit from being able to generate interaction with more than two persons. Also, with more extensive databases we could try to generate more complex interactions that contain several short interactions like it is already done for the text-to-motion task [165]. The current text-to-motion methods are able to generate realistic sequences of action but only for

a single person.

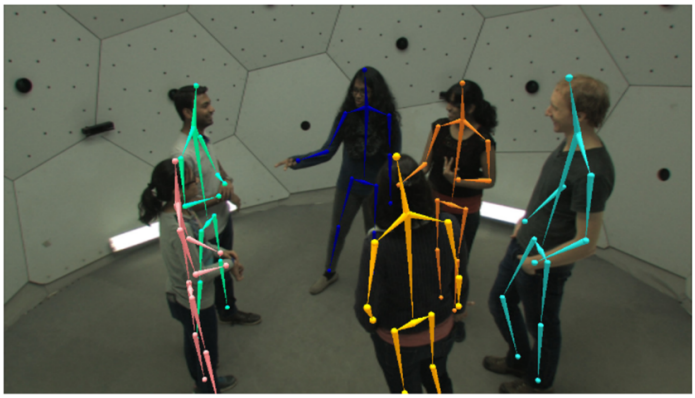


Figure 6.6: Figure from [67]. Example of interaction between multiple persons from the CMU Panoptic dataset.

6.2.4 Extension to Point Cloud and Surfaces

In this thesis, we work exclusively with 3D skeletons but there are other representations of the human body such as 3D meshes or dense point clouds. These representations are much more complex but contain a lot more information about body deformations. Point cloud methods that deal with human motion use dense point clouds to classify human motion [44, 45, 148] while generative point cloud methods focus on generating the 3D shape of objects without motion [155, 99]. There are few works on the generation of human motion through the point cloud. Thus, exploring human motion generation through dense 3D point clouds could be interesting but would require developing a lightweight method due to the huge amount of data from dense point clouds.

BIBLIOGRAPHY

[1] Chaitanya Ahuja et al. “No Gestures Left Behind: Learning Relationships between Spoken Language and Freeform Gestures”. In: *EMNLP*. 2020.

[2] Chaitanya Ahuja et al. “To React or Not to React: End-to-End Visual Pose Forecasting for Personalized Avatar during Dyadic Conversations”. In: *ICMI*. 2019.

[3] Emre Aksan et al. “A spatio-temporal transformer for 3d human motion prediction”. In: *2021 International Conference on 3D Vision (3DV)*. IEEE. 2021, pp. 565–574.

[4] Simon Alexanderson et al. “Listen, denoise, action! Audio-driven motion synthesis with diffusion models”. In: *arXiv preprint arXiv:2211.09707* (2022).

[5] Sadegh Aliakbarian et al. “A Stochastic Conditioning Scheme for Diverse Human Motion Prediction”. In: *ICCV*. 2020.

[6] Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay. “Face aging with conditional generative adversarial networks”. In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 2089–2093. DOI: 10.1109/ICIP.2017.8296650.

[7] Danilo Avola et al. “2-D Skeleton-Based Action Recognition via Two-Branch Stacked LSTM-RNNs”. In: *IEEE Transactions on Multimedia* 22.10 (2020), pp. 2481–2496. DOI: 10.1109/TMM.2019.2960588.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[9] Emad Barsoum, John Kender, and Zicheng Liu. “HP-GAN: Probabilistic 3D human motion prediction via GAN”. In: *CVPR Workshops*. 2018, pp. 1418–1427.

[10] Emad Barsoum, John Kender, and Zicheng Liu. “HP-GAN: Probabilistic 3D Human Motion Prediction via GAN”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 1499–149909. DOI: 10.1109/CVPRW.2018.00191.

[11] M. Baruah and B. Banerjee. “A Multimodal Predictive Agent Model for Human Interaction Generation”. In: *CVPR Workshops*. 2020.

[12] M. Baruah and B. Banerjee. “A Multimodal Predictive Agent Model for Human Interaction Generation”. In: *CVPR Workshops*. 2020.

[13] Valentin Bazarevsky et al. “BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs”. In: *ArXiv* abs/1907.05047 (2019).

[14] Valentin Bazarevsky et al. “BlazePose: On-device Real-time Body Pose tracking”. In: *ArXiv* abs/2006.10204 (2020).

[15] Boulbaba Ben Amor, Jingyong Su, and Anuj Srivastava. "Action recognition using rate-invariant analysis of skeletal shape trajectories". In: *PAMI* 38.1 (2016), pp. 1–13.

[16] Stefano Berretti et al. "Representation, Analysis, and Recognition of 3D Humans: A Survey". In: *ACM Trans. Multim. Comput. Commun. Appl.* 14.1s (2018), 16:1–16:36. DOI: 10.1145/3182179. URL: https://doi.org/10.1145/3182179.

[17] Victoria Bloom, Vasileios Argyriou, and Dimitrios Makris. "Hierarchical transfer learning for online recognition of compound actions". In: *Computer Vision and Image Understanding* 144 (2016). Individual and Group Activities in Video Event Analysis, pp. 62–72. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2015.12.001. URL: https://www.sciencedirect.com/science/article/pii/S1077314215002647.

[18] Nadav Brandes et al. "ProteinBERT: a universal deep-learning model of protein sequence and function". In: *Bioinformatics* 38.8 (Feb. 2022), pp. 2102–2110. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btac020. eprint: https://academic.oup.com/bioinformatics/article-pdf/38/8/2102/45474534/btac020.pdf. URL: https://doi.org/10.1093/bioinformatics/btac020.

[19] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[20] Judith Butepage et al. "Deep representation learning for human motion prediction and classification". In: *CVPR*. 2017, pp. 6158–6166.

[21] Z. Cao et al. "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[22] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 213–229. ISBN: 978-3-030-58452-8.

[23] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *ECCV*. 2020.

[24] Shu-Yu Chen et al. "3D Face Reconstruction and Gaze Tracking in the HMD for Virtual Interaction '". In: *IEEE Transactions on Multimedia* (2022), pp. 1–1. DOI: 10.1109/TMM.2022.3156820.

[25] Yunpeng Chen et al. "Graph-based global reasoning networks". In: *CVPR*. 2019.

[26] Ke Cheng et al. “Skeleton-Based Action Recognition With Shift Graph Convolutional Network”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 180–189. DOI: 10.1109/CVPR42600.2020.00026.

[27] Baptiste Chopin et al. “3D Skeleton-based Human Motion Prediction with Manifold-Aware GAN”. In: *IEEE Transactions on Biometrics, Behavior, and Identity Science* (2022), pp. 1–1. DOI: 10.1109/TBIOM.2022.3215067.

[28] Baptiste Chopin et al. “Human Motion Prediction Using Manifold-Aware Wasserstein GAN”. In: *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*. IEEE. 2021, pp. 1–8.

[29] Baptiste Chopin et al. “Interaction Transformer for Human Reaction Generation”. In: *IEEE Transactions on Multimedia* (2023), pp. 1–13. DOI: 10.1109/TMM.2023.3242152.

[30] Liu Chunhui et al. “PKU-MMD: A Large Scale Benchmark for Continuous Multi-Modal Human Action Understanding”. In: *arXiv preprint arXiv:1703.07475* (2017).

[31] C. Coppola et al. “Automatic Detection of Human Interactions from RGB-D Data for Social Activity Classification”. In: *IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN)*. 2017, pp. 871–876.

[32] C. Coppola et al. “Social Activity Recognition based on Probabilistic Merging of Skeleton Features with Proximity Priors from RGB-D Data”. In: *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 5055–5061.

[33] Enric Corona et al. “Context-aware human motion prediction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6992–7001.

[34] Qiongjie Cui, Huaijiang Sun, and Fei Yang. “Learning Dynamic Relationships for 3D Human Motion Prediction”. In: *CVPR*. 2020.

[35] Rishabh Dabral et al. *MoFusion: A Framework for Denoising-Diffusion-based Motion Synthesis*. 2022. arXiv: 2212.04495 [cs.CV].

[36] Mohamed Daoudi et al. “A New Computational Approach to Identify Human Social Intention in Action”. In: *13th IEEE International Conference on Automatic Face & Gesture Recognition, FG 2018, Xi’an, China, May 15-19, 2018*. 2018, pp. 512–516. DOI: 10.1109/FG.2018.00082. URL: https://doi.org/10.1109/FG.2018.00082.

[37] Maxime Devanne et al. “3-D human action recognition by shape analysis of motion trajectories on Riemannian manifold”. In: *IEEE TC* 45.7 (2014), pp. 1340–1352.

[38] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *NAACL*. 2019.

[39] Steve DiPaola and Özge Nilay Yalçın. “A multi-layer artificial intelligence and sensing based affective conversational embodied agent”. en. In: (), p. 2.

[40] Hassen Drira et al. “3D face recognition under expressions, occlusions, and pose variations”. In: *PAMI* 35.9 (2013), pp. 2270–2283.

[41] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. “Generative Multi-Adversarial Networks”. In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=Byk-VI9eg.

[42] Jesse Engel et al. “GANSynth: Adversarial Neural Audio Synthesis”. In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=H1xQVn09FX.

[43] Patrick Esser, Robin Rombach, and Bjorn Ommer. “Taming Transformers for High-Resolution Image Synthesis”. In: *CVPR*. 2021.

[44] Hehe Fan, Yi Yang, and Mohan Kankanhalli. “Point 4D Transformer Networks for Spatio-Temporal Modeling in Point Cloud Videos”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 14204–14213.

[45] Hehe Fan et al. “PSTNet: Point Spatio-Temporal Convolution on Point Cloud Sequences”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=O3bqkf_Puys.

[46] K. Fragkiadaki et al. “Recurrent Network Models for Human Dynamics”. In: *ICCV*. 2015.

[47] Katerina Fragkiadaki et al. “Recurrent Network Models for Human Dynamics”. In: *ICCV*. 2015, pp. 4346–4354.

[48] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. “Diffusion Improves Graph Learning”. In: Oct. 2019.

[49] Partha Ghosh et al. “Learning human motion models for long-term predictions”. In: *3DV*. 2017, pp. 458–466.

[50] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Fourth. The Johns Hopkins University Press, 2013.

[51] Ian J. Goodfellow et al. “Generative Adversarial Networks”. en. In: *arXiv:1406.2661 [cs, stat]* (June 2014). arXiv: 1406.2661. URL: http://arxiv.org/abs/1406.2661 (visited on 06/16/2020).

[52] Liang-Yan Gui et al. “Adversarial Geometry-Aware Human Motion Prediction”. In: *ECCV*. 2018, pp. 823–842.

[53] Ishaan Gulrajani et al. “Improved training of Wasserstein GANs”. In: *NIPS*. 2017, pp. 5767–5777.

[54] Chuan Guo et al. “TM2T: Stochastic and Tokenized Modeling for the Reciprocal Generation of 3D Human Motions and Texts”. In: *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXV*. 2022, pp. 580–597.

[55] Agrim Gupta et al. “Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks”. In: *CVPR*. 2018.

[56] Ikhsanul Habibie et al. “Learning Speech-driven 3D Conversational Gestures from Video”. In: *ACM International Conference on Intelligent Virtual Agents (IVA)*. 2021. eprint: Todo.

[57] William Harvey et al. “Flexible Diffusion Modeling of Long Videos”. In: *arXiv preprint arXiv:2205.11495* (2022).

[58] Martin Heusel et al. “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium”. In: *NeurIPS*. 2017.

[59] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising Diffusion Probabilistic Models”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. URL: https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab1017Paper.pdf.

[60] Emiel Hoogeboom et al. “Equivariant Diffusion for Molecule Generation in 3D”. In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 8867–8887.

[61] T. Hu, Xinyan Zhu, and W. Guo. “Two-person interaction recognition based on key poses”. In: *Journal of Computational Information Systems* 10 (2014), pp. 1965–1972.

[62] Minjie Hua et al. “Towards More Realistic Human-Robot Conversation: A Seq2Seq-based Body Gesture Interaction System”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 1393–1400. DOI: 10.1109/IROS40897.2019.8968038.

[63] Zhiwu Huang, Jiqing Wu, and Luc Van Gool. “Manifold-Valued Image Generation with Wasserstein Generative Adversarial Nets”. In: *AAAI*. 2019, pp. 3886–3893.

[64] Catalin Ionescu et al. “Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments”. en. In: *PAMI* 36.7 (July 2014), pp. 1325–1339. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2013.248. URL: http://ieeexplore.ieee.org/document/6682899/ (visited on 06/08/2020).

[65] Ashesh Jain et al. “Structural-RNN: Deep Learning on Spatio-Temporal Graphs”. In: *CVPR*. 2016, pp. 5308–5317.

[66] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. "TransGAN: Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 14745–14758. URL: https://proceedings.neurips.cc/paper/2021/file/7c220a2091c26a7f5e9f1cfb099511e3-Paper.pdf.

[67] Hanbyul Joo et al. "Panoptic Studio: A Massively Multiview System for Social Interaction Capture". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[68] Hanbyul Joo et al. "Towards Social Artificial Intelligence: Nonverbal Social Signal Prediction in a Triadic Interaction". In: *CVPR*. 2019.

[69] Amin Jourabloo et al. "Robust egocentric photo-realistic facial expression transfer for virtual reality". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 20323–20332.

[70] Anis Kacem et al. "A Novel Geometric Framework on Gram Matrix Trajectories for Human Behavior Understanding". In: *PAMI* 42.1 (2020), pp. 1–14. DOI: 10.1109/TPAMI.2018.2872564. URL: https://doi.org/10.1109/TPAMI.2018.2872564.

[71] Anis Kacem et al. "A Novel Geometric Framework on Gram Matrix Trajectories for Human Behavior Understanding". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 42.1 (2020), pp. 1–14.

[72] Hermann Karcher. "Riemannian center of mass and mollifier smoothing". In: *Communications on pure and applied mathematics* 30.5 (1977), pp. 509–541.

[73] Yury Kartynnik et al. "Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs". In: *CVPR Workshop on Computer Vision for Augmented and Virtual Reality 2019*. Long Beach, CA, 2019. URL: https://arxiv.org/abs/1907.06724.

[74] Isinsu Katircioglu et al. "Dyadic human motion prediction". In: *arXiv preprint arXiv:2112.00396* (2021).

[75] Will Kay et al. "The kinetics human action video dataset". In: *arXiv preprint arXiv:1705.06950* (2017).

[76] Junbeom Kim et al. "A Bipartite Graph Neural Network Approach for Scalable Beamforming Optimization". In: *IEEE Transactions on Wireless Communications* 22.1 (2023), pp. 333–347. DOI: 10.1109/TWC.2022.3193138.

[77] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR* (Dec. 2014).

[78] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR*. 2015.

[79] H. S. Koppula and A. Saxena. "Anticipating human activities for reactive robotic response". In: *IROS*. 2013, pp. 2071–2071.

[80] Bruno Korbar, Du Tran, and Lorenzo Torresani. "SCSampler: Sampling Salient Clips From Video for Efficient Action Recognition". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[81] Jean Kossaifi et al. "AFEW-VA Database for Valence and Arousal estimation In-The-Wild". In: *Image and Vision Computing* 65 (Feb. 2017). DOI: 10.1016/j.imavis.2017.02.001.

[82] Lucas Kovar, Michael Gleicher, and Frédéric H. Pighin. "Motion graphs". In: *ACM SIGGRAPH Classes*. 2008, 51:1–51:10.

[83] Jogendra Kundu et al. "Cross-Conditioned Recurrent Networks for Long-Term Synthesis of Inter-Person Human Motion Interactions". In: *WACV*. 2020.

[84] Christian Ledig et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[85] Gilwoo Lee et al. "Talking With Hands 16.2M: A Large-Scale Dataset of Synchronized Body-Finger Motion and Audio for Conversational Motion Analysis and Synthesis". In: *ICCV*. 2019.

[86] Hsin-Ying Lee et al. "Dancing to Music". In: *NeurIPS*. 2019.

[87] Chen Li et al. "Convolutional Sequence to Sequence Model for Human Dynamics". In: *CVPR*. 2018, pp. 5226–5234.

[88] Qimai Li, Zhichao Han, and Xiao-Ming Wu. "Deeper insights into graph convolutional networks for semi-supervised learning". In: *AAAI*. 2018.

[89] Rongjie Li et al. "Bipartite Graph Network With Adaptive Message Passing for Unbiased Scene Graph Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 11109–11119.

[90] Ruilong Li et al. "AI Choreographer: Music Conditioned 3D Dance Generation with AIST++". In: *ICCV* (2021), pp. 13381–13392.

[91] Yanran Li et al. "Efficient convolutional hierarchical autoencoder for human motion prediction". English. In: *The Visual Computer* 35.6 (June 2019), pp. 1143–1156. ISSN: 0178-2789. DOI: 10.1007/s00371-019-01692-9.

[92] Kevin Lin et al. "Adversarial Ranking for Language Generation". In: Dec. 2017.

[93] Di Liu et al. "Adaptive Attention Memory Graph Convolutional Networks for Skeleton-Based Action Recognition". In: *Sensors* 21.20 (2021). ISSN: 1424-8220. DOI: 10.3390/s21206761. URL: https://www.mdpi.com/1424-8220/21/20/6761.

[94] Jun Liu et al. "NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding". en. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019). arXiv: 1905.04757, pp. 1–1. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2019.2916873. URL: http://arxiv.org/abs/1905.04757 (visited on 12/02/2019).

[95] Xiaoli Liu et al. "TrajectoryCNN: A New Spatio-Temporal Feature Learning Network for Human Motion Prediction". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.6 (2021), pp. 2133–2146. DOI: 10.1109/TCSVT.2020.3021409.

[96] Zhenguang Liu et al. "Aggregated Multi-GANs for Controlled 3D Human Motion Prediction". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.3 (2021), pp. 2225–2232. DOI: 10.1609/aaai.v35i3.16321. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16321.

[97] Matthew Loper et al. "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34.6 (Oct. 2015), 248:1–248:16.

[98] Thomas Lucas et al. "PoseGPT: Quantization-based 3D Human Motion Generation and Forecasting". In: *European Conference on Computer Vision (ECCV)*. 2022.

[99] Shitong Luo and Wei Hu. "Diffusion Probabilistic Models for 3D Point Cloud Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[100] Kedi Lyu et al. "Learning Human Motion Prediction via Stochastic Differential Equations". In: Oct. 2021, pp. 4976–4984. DOI: 10.1145/3474085.3475630.

[101] Zhaoyang Lyu et al. "Accelerating Diffusion Models via Early Stop of the Diffusion Process". In: *ArXiv* abs/2205.12524 (2022).

[102] Hengbo Ma et al. "Multi-Objective Diverse Human Motion Prediction With Knowledge Distillation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8161–8171.

[103] Tiezheng Ma et al. "Progressively Generating Better Initial Guesses Towards Next Stages for High-Quality Human Motion Prediction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 6437–6446.

[104] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *JMLR* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[105] Mehran Maghoumi and Joseph J. LaViola. "DeepGRU: Deep Gesture Recognition Utility". In: *Advances in Visual Computing*. 2019.

[106] Wei Mao, Miaomiao Liu, and Mathieu Salzmann. "Generating Smooth Pose Sequences for Diverse Human Motion Prediction". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 13289–13298. DOI: 10.1109/ICCV48922.2021.01306.

[107] Wei Mao, Miaomiao Liu, and Mathieu Salzmann. "History repeats itself: Human motion prediction via motion attention". In: *European Conference on Computer Vision*. Springer. 2020, pp. 474–489.

[108] Wei Mao et al. "Learning Trajectory Dependencies for Human Motion Prediction". In: *ICCV*. 2019, pp. 9488–9496.

[109] Julieta Martinez, Michael J. Black, and Javier Romero. "On Human Motion Prediction Using Recurrent Neural Networks". In: *CVPR*. 2017, pp. 4674–4683.

[110] Shaked Nava. "Avatars and Virtual Agents - Relationship Interfaces for the Elderly". In: *Healthcare Technology Letters* 4 (May 2017). DOI: 10.1049/htl.2017.0009.

[111] Will Norcliffe-Brown, Efstathios Vafeias, and Sarah Parisot. "Learning conditioned graph structures for interpretable visual question answering". In: *NeurIPS*. 2018.

[112] Naima Otberdout et al. "Dynamic Facial Expression Generation on Hilbert Hypersphere with Conditional Wasserstein Generative Adversarial Nets". In: *PAMI* (2020), pp. 1–1.

[113] Naima Otberdout et al. "Sparse to dense dynamic 3d facial expression generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 20385–20394.

[114] Brian Paden et al. "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles". In: *T-IV* 1.1 (2016), pp. 33–55.

[115] Niki Parmar et al. "Image Transformer". In: *ICML*. 2018.

[116] Georgios Pavlakos et al. "Expressive Body Capture: 3D Hands, Face, and Body from a Single Image". In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10975–10985.

[117] Mathis Petrovich, Michael J. Black, and Gül Varol. "Action-Conditioned 3D Human Motion Synthesis With Transformer VAE". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 10985–10995.

[118] Mathis Petrovich, Michael J. Black, and Gül Varol. "Action-Conditioned 3D Human Motion Synthesis with Transformer VAE". In: *International Conference on Computer Vision (ICCV)*. 2021.

[119] Chiara Plizzari, Marco Cannici, and Matteo Matteucci. "Skeleton-based action recognition via spatial and temporal transformer networks". In: *CVIU* 208-209 (2021), p. 103219.

[120] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *CVPR*. 2017.

[121] Xiaojuan Qi et al. “3d graph neural networks for rgbd semantic segmentation”. In: *CVPR*. 2017.

[122] Tianwen Qian et al. “Scene Graph Refinement Network for Visual Question Answering”. In: *IEEE Transactions on Multimedia* (2022), pp. 1–1. DOI: 10.1109/TMM.2022.3169065.

[123] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *CoRR* abs/1511.06434 (2016).

[124] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[125] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8748–8763. URL: http://proceedings.mlr.press/v139/radford21a.html.

[126] Aditya Ramesh et al. “Zero-Shot Text-to-Image Generation”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8821–8831. URL: https://proceedings.mlr.press/v139/ramesh21a.html.

[127] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. “Vision Transformers for Dense Prediction”. In: *ArXiv preprint* (2021).

[128] Grégory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. “LCR-Net++: Multi-person 2D and 3D Pose Detection in Natural Images”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Jan. 2019). DOI: 10.1109/TPAMI.2019.2892985.

[129] Chitwan Saharia et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. In: *arXiv preprint arXiv:2205.11487* (2022).

[130] Franco Scarselli et al. “The Graph Neural Network Model”. In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.

[131] Amir Shahroudy et al. “Deep Multimodal Feature Analysis for Action Recognition in RGB+D Videos”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.5 (2018), pp. 1045–1058. DOI: 10.1109/TPAMI.2017.2691321.

[132] Zhuoran Shen et al. “Efficient attention: Attention with linear complexities”. In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2021, pp. 3531–3539.

[133] Theodoros Sofianos et al. “Space-Time-Separable Graph Convolutional Network for Pose Forecasting”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 11189–11198.

[134] Theodoros Sofianos et al. “Space-Time-Separable Graph Convolutional Network for Pose Forecasting”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 11209–11218.

[135] Jascha Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 2256–2265.

[136] Anuj Srivastava and Eric P. Klassen. *Functional and Shape Data Analysis*. Springer, New York, NY, 2016.

[137] Anuj Srivastava et al. “Shape Analysis of Elastic Curves in Euclidean Spaces”. In: *PAMI* 33.7 (2011), pp. 1415–1428. DOI: 10.1109/TPAMI.2010.184. URL: https://doi.org/10.1109/TPAMI.2010.184.

[138] Hao Tang et al. “Bipartite Graph Reasoning GANs for Person Image Generation”. In: *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press, 2020. URL: https://www.bmvc2020-conference.com/assets/papers/0689.pdf.

[139] Hao Tang et al. “Bipartite Graph Reasoning GANs for Person Image Generation”. In: *BMVC*. 2020.

[140] Hao Tang et al. “Bipartite Graph Reasoning GANs for Person Pose and Facial Image Synthesis”. In: *International Journal of Computer Vision (IJCV)* (2022).

[141] Jonathan Tseng, Rodrigo Castellon, and C. Karen Liu. “EDGE: Editable Dance Generation From Music”. In: *CoRR* abs/2211.10658 (2022).

[142] Sergey Tulyakov et al. “MoCoGAN: Decomposing motion and content for video generation”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 1526–1535.

[143] Pavan K. Turaga and Rama Chellappa. “Locally time-invariant models of human activities using trajectories on the Grassmannian”. In: *CVPR*. 2009, pp. 2435–2441.

[144] Thomas Unterthiner et al. “Towards Accurate Generative Models of Video: A New Metric & Challenges”. In: *arXiv preprint arXiv:1812.01717* (2018).

[145] Bazarevsky Valentin et al. “BlazePose: On-device real-time body pose tracking”. en. In: *Proc. CVPR Workshop Comput. Vis. Augmented Virtual Reality 2020*. IEEE, 2020, pp. 1–4.

[146] Ashish Vaswani et al. “Attention is All you Need”. In: *NeurIPS*. 2017.

[147] David Vogt et al. “A Data-Driven Method for Real-Time Character Animation in Human-Agent Interaction”. In: Aug. 2014, pp. 463–476. ISBN: 9783319097664. DOI: 10.1007/978-3-319-09767-1_57.

[148] Yancheng Wang et al. “3DV: 3D Dynamic Voxel for Action Recognition in Depth Video”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[149] Zhenyi Wang et al. “Learning Diverse Stochastic Human-Action Generators by Learning Smooth Latent Transitions”. In: *AAAI*. 2020.

[150] Mao Wei et al. “Learning Trajectory Dependencies for Human Motion Prediction”. In: *ICCV*. 2019, pp. 9488–9496.

[151] Max Welling and Thomas N Kipf. “Semi-supervised classification with graph convolutional networks”. In: *J. International Conference on Learning Representations (ICLR 2017)*. 2016.

[152] Jiajun Wu et al. “Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 82–90.

[153] Liang Xu et al. “ActFormer: A GAN-based Transformer towards General Action-Conditioned 3D Human Motion Generation”. In: *arXiv e-prints*, arXiv:2203.07706 (Mar. 2022), arXiv:2203.07706. arXiv: 2203.07706 [cs.CV].

[154] Ning Xu et al. “Multi-Modal & Multi-View & Interactive Benchmark Dataset for Human Action Recognition”. In: *Proceedings of the 23rd ACM International Conference on Multimedia*. MM ’15. Brisbane, Australia: Association for Computing Machinery, 2015, 1195–1198. ISBN: 9781450334594. DOI: 10.1145/2733373.2806315. URL: https://doi.org/10.1145/2733373.2806315.

[155] Ximing Yang et al. “CPCGAN: A Controllable 3D Point Cloud Generative Adversarial Network with Semantic Label Generating”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.4 (2021), pp. 3154–3162. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16425.

[156] Xin Yang et al. “Exposing GAN-Synthesized Faces Using Landmark Locations”. In: *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*. IHMMSec’19. Paris, France: Association for Computing Machinery, 2019, 113–118. ISBN: 9781450368216. DOI: 10.1145/3335203.3335724. URL: https://doi.org/10.1145/3335203.3335724.

[157] Yanzhe Yang, Jimei Yang, and Jessica Hodgins. “Statistics-Based Motion Synthesis for Social Conversations”. In: *SIGGRAPH*. 2020.

[158] Sifan Ye et al. “Scene Synthesis from Human Motion”. In: *SIGGRAPH Asia 2022 Conference Papers*. SA ’22. Daegu, Republic of Korea: Association for Computing Machinery, 2022. ISBN: 9781450394703. DOI: 10.1145/3550469.3555426. URL: https://doi.org/10.1145/3550469.3555426.

[159] Wenjie Yin et al. *Graph-based Normalizing Flow for Human Motion Generation and Reconstruction*. 2021. arXiv: 2104.03020 [cs.CV].

[160] Youngwoo Yoon et al. “Speech Gesture Generation from the Trimodal Context of Text, Audio, and Speaker Identity”. In: *ACM Trans. Graph.* 39.6 (2020).

[161] Jiahui Yu et al. “Generative Image Inpainting With Contextual Attention”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[162] Xianwen Yu et al. “VAEGAN: A Collaborative Filtering Framework based on Adversarial Variational Autoencoders”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, July 2019, pp. 4206–4212. DOI: 10.24963/ijcai.2019/584. URL: https://doi.org/10.24963/ijcai.2019/584.

[163] Kiwon Yun et al. “Two-person Interaction Detection Using Body-Pose Features and Multiple Instance Learning”. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE. 2012.

[164] Jianjing Zhang et al. “Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly”. In: *CIRP Annals* 69.1 (2020), pp. 9–12. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2020.04.077. URL: https://www.sciencedirect.com/science/article/pii/S0007850620300998.

[165] Mingyuan Zhang et al. “MotionDiffuse: Text-Driven Human Motion Generation with Diffusion Model”. In: *arXiv preprint arXiv:2208.15001* (2022).

[166] Richard Zhang et al. “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric”. In: *CVPR*. 2018.

[167] Rui Zhao, Hui Su, and Qiang Ji. “Bayesian Adversarial Human Motion Synthesis”. In: *CVPR*. 2020.

[168] Chongyang Zhong et al. “Spatio-Temporal Gating-Adjacency GCN for Human Motion Prediction”. In: June 2022.

[169] Yi Zhou et al. “Auto-Conditioned Recurrent Networks for Extended Complex Human Motion Synthesis”. In: *ICLR*. 2018.

[170] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.

[171] Liping Zhu et al. “Dyadic relational graph convolutional networks for skeleton-based human interaction recognition”. In: *Pattern Recognition* 115 (2021).