Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Lucas MOUROT

Deep Learning for Skeletal Character Animation: Topology Editing, Retargeting and Cleaning

Thèse présentée et soutenue à Rennes, le 4 mai 2023
Unité de recherche : INRIA Rennes - Bretagne Atlantique et Interdigital, Inc

Rapporteurs avant soutenance :

	Alexandre MEYER	Maître de conférence, Université Claude Bernard Lyon 1
	Damien ROHMER	Professeur, École Polytechnique

Composition du Jury :

Président :		
Examinateurs :	Alexandre MEYER	Maître de conférence, Université Claude Bernard Lyon 1
	Damien ROHMER	Professeur, École Polytechnique
	Maud MARCHAL	Professeur, INSA
	Ronan BOULIC	Senior Scientist, EPFL
Co-dir. de thèse :	Ludovic HOYET	Chargé de Recherche, INRIA
	Pierre HELLIER	Principal Scientist, Interdigital

Invité(s) :

Enc. de thèse :	François LE CLERC	Senior Scientist, Interdigital

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AdaIN	adaptive instance normalisation
AE	autoencoder
AUC	area under the curve
BCE	binary cross-entropy
BoS	base of support
BN	batch normalisation
BiLSTM	bidirectional LSTM
CNN	convolutional neural network
CoP	centre of pressure
DCGAN	deep convolutional GAN
DL	deep learning
DNN	deep neural network
DOF	degree of freedom
ELU	exponential linear unit
ERD	encoder-recurrent-decoder
FC	fully-connected
FCRBM	factored CRBM
FK	forward kinematics
GAN	generative adversarial network
GCN	graph convolutional network
GRF	ground reaction force
GRU	gated recurrent unit
GSL	Gram-Schmidt-like
IK	inverse kinematics

IMU inertial measurement unit

KNN k-nearest neighbours

LSTM long short-term memory

LReLU leaky rectified linear unit

MAD median absolute deviation

MLP multilayer perceptron

MSE mean squared error

MSLE mean squared logarithmic error

MPJPE mean per joint position error

MPJVE mean per joint velocity error

NF normalising flows

OPA ordinary Procrustes analysis

OT optimal thresholds

PD proportional-derivative

PFNN phase-functioned neural network

PCKh percentage of correct key points normalised with head size

RBM restricted Boltzmann machine

RMSE root mean squared error

RNN recurrent neural network

ReLU rectified linear unit

SVD singular value decomposition

VAE variational autoencoder

vGRF vertical ground reaction force

ViT vision transformer

WGAN Wasserstein generative adversarial network

XCA cross-covariance attention

1

INTRODUCTION

Humans and their representations are ubiquitous in all cultures. Since the beginning of times, our ancestors have painted or sculpted myriads of works of art often with human representatives portrayed like the parietal wall paintings in Lascaux Cave about 20 thousands years ago, the Venus de Milo 2 thousands years ago or Michelangelo's David a few centuries ago. In the past decades, digital technologies have truly revolutionised the way humans can create and access culture. More and more tools have appeared to assist all kind of artists including designers and animators, especially to increase their creative capabilities and the realism of their artworks while reducing production costs. For instance, synthetic characters have very convincingly brought to life non-human fictional creatures like the Na'vi in *Avatar* or even visual identity of long time deceased actors like Grand Moff Tarkin in *Rogue One* in 2016 portrayed by the British actor Peter Cushing even though he passed away in 1994, as illustrated in Figure 1.1.

Figure 1.1 – Grand Moff Tarkin, a fictional character in the *Star Wars* franchise, first played by Peter Cushing in *Star Wars: Episode IV – A New Hope* in 1977 (left) and synthesised in *Rogue One: A Star Wars Story* in 2016 (right), 22 years after Peter Cushing passed away.

In that context, character animation has a great importance for the overall quality of the artworks. The way humans move is very diverse and conveys a lot of information about the activity performed, the character's state of mind, such as intentions, moods or emotions, the character's identity, such as morphological and biological characteristics, etc., which all have to be consistent together with other visual aspects. In addition, Newton's second law of motion inherently makes human motion a dynamic process, while our understanding of biomechanics is far from being comprehensive. For these reasons, animating human characters is still very challenging, even though huge progresses have been made recently. Therefore, compromises have to be made to balance between production costs, realism, amount of manual work, and animators' level of expertise, depending on the expected quality. For instance, the gaming industry already relies on computer-assisted character animation but animation studios still resort to a lot of manual work, e.g. in the post-production of high-budget films. The research area of character animation has been active for decades to mitigate these compromises and make animation more accessible, starting from pioneering works such as editing and deforming motion examples (e.g. Witkin et al. [188]), retargeting motions to new characters (e.g. Gleicher [44]), controlling characters using motion graphs (e.g. Kovar et al., Min et al. [85, 121]), etc.

THE EMERGENCE OF DEEP LEARNING

Over the last decade, deep learning has emerged as a powerful means to enhance the performance and capabilities of character animation. It has shown an unprecedented ability to address complex tasks in a wide variety of domains not restricted to animation, such as computer vision, natural language processing and many more. Humans are outperformed by artificial intelligence algorithms in a growing number of tasks such as classifying images [53] or playing Go [155]. This is notably due to the fact that deep neural networks (DNNs) are powerful function approximators, able to learn sophisticated patterns in complex real-world phenomena from data, which is increasingly easily captured, stored and processed upstream. Indeed, many of the recent great advances like GANs or transformers have been accompanied with an increase of data volume

needed to train those state-of-the-art models. Still, once the training phase is complete, training data is discarded, leaving compact models that are able to meet performance requirements of real-time applications or to scale to embedded systems.

In animation, deep learning (DL)-based approaches attempt to handle the human motion complexity and provide promising perspectives for cheaper and faster animation techniques with more and more fidelity and creative capabilities. However, this kind of approaches struggle to reach the quality of work obtained by skilled animators, and are not widely deployed in animation studios which still largely rely on hand-crafted animations. To imitate recent breakthroughs of deep learning in other domains, deep animation might need more high-quality motion data to feed recent data-greedy models like transformers. Indeed, large public unified databases of motion data are much more scarce than in other domains such as image databases in computer vision, and less easily aggregated. Often, high-quality motion capture systems are expensive, require controlled environment preventing in-the-wild captures, and differ in which joints or body markers are captured. Research in deep learning could have a role to play, either by improving ways to unify existing human motion databases, or by facilitating high-quality motion data acquisition e.g. through human pose estimation. For these reasons, deep learning is nowadays central to research in character animation, as well as to this thesis, from the gathering of large motion databases to the design and learning of powerful models.

SKELETAL CHARACTER ANIMATION

Traditional animation of 3D virtual humans typically uses 3D rigged skeletons with skinned meshes to provide a good trade-off between quality and complexity. Rigging offers convenient ways to manipulate 3D models as strings do for a puppet, while skinning handles the visual identity by binding coloured and textured 3D meshes to animated characters. Characters are then brought to life by animating their underlying skeleton, the topic of interest in this thesis, called *skeletal character animation* hereafter. This topic can be divided into three main categories:

— Motion synthesis, to generate new motion sequences with desired characteristics.

— Character control, to continuously drive and actuate characters from specific (typically user-driven) controls.

— Motion editing, to combine, transfer, enrich, enhance, clean, etc. existing motion sequences.

Motion synthesis strives to create novel perceptually plausible motion sequences generally in an offline manner from low-level parameters such as latent variables or higher-level parameters e.g. desired style, affective variations, or a set of key sparse poses to be semantically interpolated or extrapolated. In contrast, character control refers by definition to online control and actuation of virtual characters responding to user input flows. Typical use cases of character control include video games, where high-level user controls drive a virtual character expected to seamlessly move into and adapt to the game's environment while performing different actions such as grabbing and manipulating objects. Finally, motion editing focuses on processing existing motion sequences rather than producing novel motion sequences through control or synthesis. It regroups different tasks including the manipulation of high-level features, such as transferring the style from one motion sequence to another, or transferring the motion of one character to another with different morphology or structure. Motion editing also includes all kind of approaches to enhance existing motion sequences, e.g. by removing artefacts such as noise.

About a decade ago, methods relying on deep learning appeared and have continuously grown since then in all these topics. In the last few years, these approaches have punctually begun to reach traditional animation quality. The next steps are then to fully outperform traditional animation pipelines, by upgrading all components of animation systems to synthesise, control, and process the motion of virtual characters, with the aim of reaching, and eventually outperforming skilled animators' manual work quality which is valuable especially for the tedious manual tasks including little creativity (e.g. background characters).

OBJECTIVES

Revolving around motion editing, the goals of this thesis are twofold. Firstly, to identify and tackle current obstacles in DL-based skeletal human character animation preventing to reach skilled animators' quality of work. And secondly, to investigate promising state-of-the-art methods in deep learning to tackle those obstacles and bring new tools in the community of skeletal character animation to enhance motion data processing.

In particular, we first explored the bottleneck constituted by the limited availability of motion data from two different angles: first, the acquisition of motion data to increase the volume and diversity of high-quality in-the-wild motion data. To this end, we considered the use of deep generative models to improve and enrich motion sequences estimated from video with prior knowledge. Second, the unification of existing motion databases, e.g. to leverage both specific aspects of small databases (e.g. style labels) and the diversity and volume of larger databases. To alleviate current issues to mix motion data from different sources (mainly due to structural and morphological differences of characters' skeleton), we also investigated a novel versatile architecture to abstract out pure motion features from skeleton structure and morphology features. Finally, we also delved into the enhancement of existing motion sequences, especially toward solving the long-standing problem in character animation known as *footskate*. In short, footskate is a family of artefacts present in motion sequences when foot movements are not consistent with the ground, which easily appear in neural networks outputs and strongly hurt the quality of a number of approaches in all topics in skeletal character animation and prevents from building upon such imperfect methods.

STRUCTURE

First, we provide an extensive review of the literature relevant to this thesis in Chapter 2. Focused on skeletal character animation based on deep learning, we begin with an overview of low-level concerns encountered when processing human motion data with DNNs, including motion representations and databases, as well as spatial

and temporal features learning. Then, we cover state-of-the-art methods in motion synthesis, character control and motion editing in separate sections. Other specific related works, like state-of-the-art deep models important to parts of this thesis, are treated within each corresponding chapter.

Then, we present in Chapter 3 a novel approach intended to improve quality and level of details of human pose estimation, which strives to capture human body with limited equipment (often a single monocular camera). Since many approaches to 3D human pose estimation broke down this problem into 2D pose estimation followed by inverse mapping from 2D to 3D, we propose an additional intermediate step to this common pipeline to complete and upsample joints of estimated 2D pose sequences, hopefully to enhance, enrich, disambiguate 2D pose sequences before solving the ill-posed problem of inverse mapping from 2D to 3D. To this end, we learn a deep representation of 2D pose sequences, exploit it afterwards by projecting incomplete input pose sequences with a lower number of joints into this representation and finally reconstruct corresponding 2D pose sequences with completed and upsampled joints, as well as improved temporal consistency.

In Chapter 4, we propose a novel versatile deep representation to unify motion sequences with different skeleton topologies and morphologies. Given the variations in locations, number and interconnections of joints across animation systems, pipelines and databases, our goal is to increase their interoperability by abstracting pure motion features out from structural and morphological features of the skeleton. To reach this goal, we design a dedicated autoencoder architecture based on transformers which handles input and output motion sequences with variable number of joints (potentially different) and condition both encoding to and decoding from its latent space on the skeleton, enabling to embed together motions with different skeleton topologies and morphologies. Moreover, this framework has multiple other applications, including the resampling of joints or the transfer of a motion from a source skeleton to a target skeleton, i.e. *motion retargeting* (see Section 2.4.2).

Finally, in Chapter 5 we tackle the long-standing problem of footskate, ubiquitous in character animation. Footskate artefacts are easily introduced during any motion processing step e.g. motion retargeting, are highly detrimental to the perceived real-

ism [138], and require foot contact labels to be cleaned up. As traditional approaches typically rely on manually annotated foot contacts, our first contribution here is a novel and unique database of motion data synchronised with pressure insoles data, serving as a physical ground truth of foot contacts. Then, we leverage this database to train a deep neural network to detect foot contacts, a necessary step in the resolution of footskate. The main novelty is the use of ground reaction forces – which are estimated from motion sequences – as a proxy representation to better capture the physical interactions between feet and ground, from which foot contacts are computed. We demonstrate that our approach outperforms traditional heuristics-based solutions. Finally, we propose a fully automatic workflow to remove footskate artefacts from motion sequences. Following recent approaches, footskate is removed by enforcing foot constraints during detected contact phases beforehand using an inverse kinematics (IK) algorithm which iteratively optimises joint angles to satisfy the constraints. We additionally leverage our deep model to preserve ground reaction forces during the optimisation and keep motion dynamics globally consistent.

For improved global consistency and readability, the order in which the chapters are presented in this thesis does not strictly follow the chronological order of publication. In particular, Chapter 4 is based on works done towards the end of the thesis and not yet published at the time of writing (see contributions below).

CONTRIBUTIONS

This thesis is based on the following contributions:

Article 1: **Lucas Mourot**, Ludovic Hoyet, François Le Clerc, François Schnitzler and Pierre Hellier, "A Survey on Deep Learning for Skeleton-Based Human Animation", *Computer Graphics Forum* 41.1, 2021, pp. 122-157, DOI: 10.1111/cgf.14426.

Article 2: **Lucas Mourot**, François Le Clerc, Cedric Thébault and Pierre Hellier, "JUMPS: Joints Upsampling Method for Pose Sequences", *Proceedings of 2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 1096-1103, DOI: 10.1109/ICPR48806.2021.9412160.

— Presented at the *25th International Conference on Pattern Recognition (ICPR)*, Virtual / Milano, Italy, 2021.

Article 3: **Lucas Mourot**, Ludovic Hoyet, François Le Clerc and Pierre Hellier, "Transformers-based Unified Deep Motion Representation", *article under preparation*.

— Code repository to be publicly released.

— Patent application to be prepared.

Article 4: **Lucas Mourot**, Ludovic Hoyet, François Le Clerc and Pierre Hellier, "UnderPressure: Deep Learning for Foot Contact Detection, Ground Reaction Force Estimation and Footskate Cleanup", *Computer Graphics Forum* 41.8, 2022, pp. 195-206, DOI: 10.1111/cgf.14635.

— Database and code repository publicly released, available at https://github.com/InterDigitalInc/UnderPressure.

— Patent application: **Lucas Mourot**, François Le Clerc and Pierre Hellier, "UnderPressure: Deep Learning of Foot Contacts and Forces", 2022.

— Presented at the *24èmes journées du Groupe de Travail Animation et Simulation (GTAS)*, Vannes, France, 2022.

— Presented at the *21th ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)*, Durham, UK, 2022.

2

LITERATURE REVIEW

Chapter abstract

Human character animation is often critical in entertainment content production, including video games, virtual reality or fiction films. To this end, deep neural networks drive most recent advances through deep learning. In this chapter, we propose a review of the literature relevant to the rest of the thesis, i.e. state-of-the-art methods in skeletal human character animation based on deep learning. First, we introduce motion data representations, most common human motion datasets and how basic deep models can be enhanced to foster learning of spatial and temporal patterns in motion data. Second, we cover approaches creating new motion sequences, divided into motion synthesis and character control. Finally, we provide an overview of motion editing, a topic at the heart of the thesis. The content of this chapter mainly stems from our survey article on deep learning for skeletal character animation (see Article 1). In particular, we restricted the content to the literature relevant to the thesis, and completed and updated whenever novel works have been published since then.

2.1 INTRODUCTION

As stated in the introduction of the thesis, human characters are ubiquitous in culture and digital technologies have revolutionised in the past decades how we create and access artworks. In entertainment content production, e.g. video games, virtual reality or fiction films, the quality of human character animations is critical. Moreover, most of the recent advances to enhance the performance and capabilities in this area rely on deep learning, which outperforms statistical or other classical approaches that were used about up to a decade ago. In this chapter, we review the literature of the recent growing trend of deep learning in skeletal character animation, mainly focused on humanoid characters. *Skeletal* here means using a representation derived from a skeleton, as commonly used in the movie and game industries in combination with 3D skinned meshes (see Section 2.2.1).

We begin with an overview of low-level concerns encountered when processing human motion data with DNNs, presenting pose representations (Section 2.2.1) and human motion datasets (Section 2.2.2) frequently used in the literature, as well as how to efficiently and successfully learn spatial (Section 2.2.3) and temporal (Section 2.2.4) features.

Then we cover in Section 2.3 the general task of generating new motion sequences. We distinguish here two different families of applications interested in creating new motion sequences which are motion synthesis (Section 2.3.1) and character control (Section 2.3.2). Their main difference is that character control strives to dynamically generate motions, continuously responding to user inputs while motion synthesis is interested in synthesising new motion sequences in an offline fashion with all the parameters known in advance.

Finally, Section 2.4 gathers motion editing approaches at large, i.e. methods aiming to process or transform any aspects of existing motion data. Motion cleaning (Section 2.4.1) enhances motion data, e.g. by removing noise or filling in missing information such as marker or joint positions. Then, retargeting (Section 2.4.2) strives to transfer the motion from a source character to a target character, while motion style transfer (Section 2.4.3)

edits the style of a motion segment while preserving the action performed and the character.

2.2 HUMAN MOTION REPRESENTATION, DATA AND MODELLING

Choices of input and output spaces for DNNs are often impactful in deep learning on the effectiveness of the learning phase and on what specific aspects of the data will be retained. When dealing with human motion, the pose representation mainly determines these input and output spaces. Moreover in DNNs, the computational workflow can be structured around spatial and temporal aspects of motion. In this section we explore the different human pose representations commonly encountered in deep animation and their key strengths and weaknesses (Section 2.2.1), commonly used datasets (Section 2.2.2), as well as DNN architectures structured with respect to spatial (Section 2.2.3) and temporal (Section 2.2.4) domains.

2.2.1 *Pose Representations*

Traditional animation approaches typically use 3D rigged skeletons with skinned meshes, which provides a good trade-off between quality and complexity. Rigging offers convenient ways to manipulate 3D models as strings do for a puppet, while skinning is the process of binding actual 3D meshes to animated characters. In that framework, human motions are usually represented as sequences of poses separated by constant time intervals whose rate generally ranges from 30 to 250 hertz. At each time step, the state of the human body is then represented as a set of links (i.e. bones) connected by joints. This skeletal representation is a good compromise for the complexity and the diversity of human movements that can be represented. When bone lengths are kept constant over time, the degrees of freedom (DOFs) are the orientations of the bones, commonly expressed relative to their parent. In the following, we call such a representation an *angular pose representation*, as opposed to a *positional pose*

representation where the skeleton DOFs are the coordinates of the joint positions, which does not explicitly constrain bone lengths to remain constant over time.

Positional Pose Representations

In a positional pose representation, each joint is directly represented by its position, sometimes expressed at each time step in the body's local coordinate system, which allows the decomposition of the whole motion into the local movements of limbs with respect to the body itself and the global movement of the body with respect to its environment. Although different coordinate systems could be formulated to embed the set of joint positions, positional pose representations almost always rely on the Cartesian coordinate system. It has neither discontinuities nor singularities and constitutes a convenient space for interpolation, visualisation and optimisation. Moreover, within the framework described here, positional pose representations do not present some of the limitations inherent to angular representations presented in the following section. However, it suffers from some limitations related to the structure of human motions. For instance, joint positions do not encode the information of bone orientations around themselves which is often needed for concrete applications in animation, in order to display more natural mesh deformations. Positional representations also do not explicitly constrain bone lengths to remain constant over time, therefore requiring reliance on additional constraints to ensure that the skeleton does not break apart. For these reasons, communities closer to animation, such as computer graphics, rarely use purely positional pose representations, while on the opposite, communities more commonly involved in deep learning, such as computer vision, are more prone to employ these representations.

Angular Pose Representations

Angular representations have been widely used in animation mainly because their hierarchical nature allows straightforward orientation of any joint together with all of its descendants, while keeping bone lengths constant. Indeed, the position of each joint is described with respect to its parent as a 3D rigid transformation, often decomposed into a variable rotation and a fixed translation, corresponding to the joint

orientation and the bone dimensions respectively. Main differences among angular pose representations are determined by the parameterisation of the rotations, however there are also representations working at the level of rigid transformation parameterisations.

Formally, the set of all rotations of $\mathbb{R}^3$ equipped with the composition is the 3D rotation group often denoted $\mathrm{SO}(3)$, standing for special orthogonal group of dimension 3. $\mathrm{SO}(3)$ can be identified with the group of orthogonal 3×3 matrices with determinant 1 under the matrix multiplication. Similarly, the 3D special Euclidean group whose elements are proper 3D rigid transformations (i.e. excluding reflections) is $\mathrm{SE}(3) = \mathrm{SO}(3) \times \mathbb{R}^3$. Both $\mathrm{SO}(3)$ and $\mathrm{SE}(3)$ are Lie groups, i.e. differentiable spaces that locally resemble Euclidean space. Furthermore, a Lie algebra is associated to every Lie group, called $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$ for $\mathrm{SO}(3)$ and $\mathrm{SE}(3)$, respectively. Lie algebras are vector spaces tangent to their Lie group at the identity element completely capturing its local structure, making them compelling as representation spaces.

EULER ANGLES. The most intuitive parameterisation of $\mathrm{SO}(3)$ is probably *Euler angles*, that represents a 3D orientation as three successive rotations around different axes, e.g. yaw, pitch and roll. However, it suffers from the well-known gimbal lock when two of the three rotation axes align, causing a DOF to be lost. Gimbal lock can be avoided only if at least one rotation axis is limited to a range smaller than 180°, which is not always possible in practice. As a result, Euler angles are unsuitable for IK, dynamics and spacetime optimisation [47]. Moreover, they do not work well for interpolations since the space of orientations is highly nonlinear [47]. Finally, multiple conventions exist for the axes considered, including their order, which requires to define them formally in each application to avoid any ambiguity. For these reasons, this representation is inappropriate for a lot of applications.

LIE ALGEBRAS. A popular angular pose representation in skeletal character animation consists in representing each joint rotation as an element of $\mathfrak{so}(3)$, where the direction and magnitude of the vector correspond to the axis and angle of the rotation [47], respectively. Since such a vector is an element of $\mathfrak{so}(3)$, the parameterisation is defined by the exponential map from $\mathfrak{so}(3)$ to $\mathrm{SO}(3)$ which can be efficiently computed

with the Rodrigues' formula [143]. This pose representation is often called *exponential map*, and is sometimes confused with the so-called *axis-angle* representation that is equivalent but separates the vector into a unit vector and a scalar describing the axis and the magnitude of the rotation.

Since Lie algebras are locally linearised versions of their Lie group, $\mathfrak{so}(3)$ is a compelling space to work with elements of $\mathrm{SO}(3)$. However, as all parameterisations of $\mathrm{SO}(3)$ in $\mathbb{R}^3$, the exponential map representation has singularities [47] leading to losing a DOF in some parts of the representation space, even though these are located on the spheres of radius $2k\pi$ for $k \in \mathbb{N}^+$, since a rotation of 2π about any axis is equivalent to no rotation. Therefore, this representation is often well-suited in animation since control and simulation deal with small time steps and thus with small rotations that stay inside the sphere of radius 2π, far from the singularities. It has been employed in early works in deep animation (e.g. [167, 168]), and broadly exploited for motion synthesis (e.g. [8, 28]), as well as in other topics (e.g. [9, 58, 69, 118]). However, as quality needs increase, long-term correlations are more and more important and inevitably imply larger rotations, getting close to the singularities of the parameterisation.

Similarly to the exponential map representation which uses $\mathfrak{so}(3)$ to represent joint orientations with respect to their parents, Liu et al. [108] proposed a pose representation using $\mathfrak{se}(3)$ to represent rigid transformations of each joint with respect to its parent. The main motivation for choosing such a representation is to explicitly encode both geometric constraints (i.e. bone lengths) and actual DOFs (i.e. joint orientations) together. Nevertheless, it still has the same singularities as the exponential map representation. As we will see in the following paragraphs, other parameterisations in higher-dimensional spaces than $\mathbb{R}^3$, i.e. over-parameterised representations, are able to prevent such singularities.

ROTATION MATRICES. In computer graphics, rotation matrices are widely used to represent 3D rotations. The corresponding parameterisation is the identity since elements of $\mathrm{SO}(3)$ are 3×3 matrices. Rotation matrices have no singularity and can be integrated together with joint translations into a 4×4 homogeneous matrix, which is elegant and effective when involved in computations like composition or inverse.

However, such a representation is particularly difficult to work with when its parameters must be estimated since the representation is over-parameterised. Indeed, not all 3×3 matrices belong to $\mathrm{SO}(3)$. By definition, a matrix $R \in \mathbb{R}^{3\times3}$ must satisfy $R^\top R = I$ and $det(R) = 1$ to be a valid 3D rotation. For instance, predicting the orientation of a joint in matrix representation would require to solve a constrained optimisation problem to ensure the validity of the rotation, which can be tedious.

UNIT QUATERNIONS. A more compact representation than rotation matrices are unit quaternions. Lying in $\mathbb{R}^4$, they are free of singularities, suitable for interpolation [47], numerically stable and computationally efficient [135]. Like $\mathfrak{so}(3)$, the space of unit quaternions has the same local geometry and topology as $\mathrm{SO}(3)$ [47]. However, unit quaternions are also over-parameterised, but have only four parameters (in comparison to nine parameters for rotation matrices). Thus, they must be constrained to remain on the unit 4-sphere. This angular representation has been popularised in DL-based animation with *QuaterNet*, a quaternion-based framework for human motion prediction proposed by Pavllo et al. [135, 136]. In that framework, Pavllo et al. introduced a penalty term in the loss function for all quaternions predicted by the network that minimises their divergence from the unit length. It encourages the network to predict valid rotations and leads to better training stability. Moreover, the predicted quaternions are also normalised after computing the penalty to enforce their validity. According to the authors, the distribution of predicted quaternion norms converges to a Gaussian with mean 1 during the training, suggesting that the model actually learns to represent valid rotations. Since Pavllo et al. [135] showed promising results using quaternions, their use is gaining popularity.

GRAM-SCHMIDT-LIKE. Zhou et al. [209] recently pointed out that all representations in $\mathbb{R}^n$ with $n \leq 4$ have discontinuities which can be unfavourable for DNNs training. They therefore introduced a continuous representation of n-dimensional rotations $\mathrm{SO}(n)$ with $n^2 - n$ dimensions. The mapping from $\mathrm{SO}(n)$ to the representation space simply drops the last column vector of the input $n \times n$ matrix. The inverse mapping back to $\mathrm{SO}(n)$, called Gram-Schmidt-like (GSL) process, is a Gram-Schmidt process over

the $n-1$ column vectors followed by the computation of the last column vector by a generalisation to n dimensions of the cross product. In the case of $\mathrm{SO}(3)$, this GSL representation gives a 6D representation. Zhou et al. [209] also provided a method to further reduce the dimensionality from 6D to 5D while still keeping a continuous representation using a stereographic projection combined with normalisation. However, they empirically found that nonlinearities introduced by the projection can make the learning process more difficult. A few of the most recent methods in skeletal character animation [98, 131, 139] make use of the 6D GSL pose representation, which tends to confirm the interest of the aforementioned appealing properties.

HIERARCHICAL REPRESENTATION LIMITATIONS. In skeletal character animation, a hierarchical modelling approach is used in conjunction with angular pose representations, i.e. the representation of the joint orientations relative to their parents. In that context, positional errors over proximal joints (e.g. the shoulder) are propagated and accumulated down the kinematic chains. This is problematic in optimisation-based frameworks such as deep learning since equally distributed joint orientation errors will result in growing joint position errors along the kinematic chains as depicted in

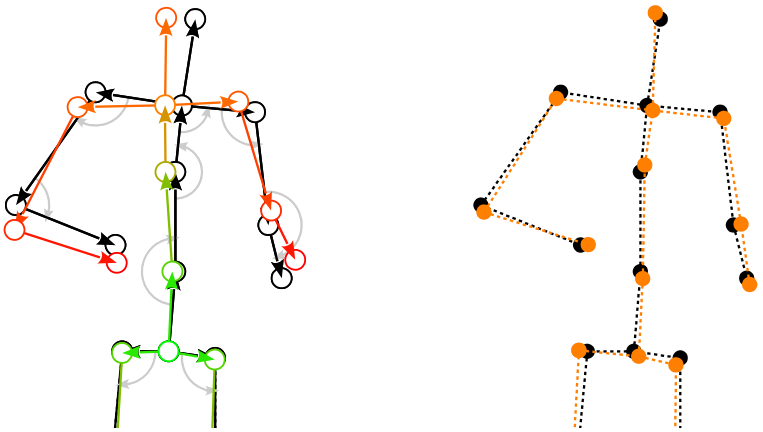


Figure 2.1 – Illustration of the error accumulation problem with angular representations: even small angular errors along the kinematic chain can lead to large accumulated joint positioning errors (left-hand stick figure, see error colour gradient). This is problematic in optimisation frameworks such as deep learning when penalising joint orientation deviations. This is not the case with positional representations (right-hand stick figure) where joint positions are directly optimised.

Figure 2.1, making it difficult to accurately handle end effectors. This is especially true in motions sequences involving fast or ample movements, e.g. running.

To solve this problem, Pavllo et al. [135, 136] performed forward kinematics (FK) to convert quaternion-based poses predicted by their DNN into 3D joint positions, and then penalised absolute position errors instead of angular errors. Since FK is a differentiable operation with respect to joint orientations, they can train their network end-to-end using a positional loss.

Hybrid Representations

As mentioned above, both angular and positional approaches for representing human poses have advantages and drawbacks that sometimes depend on the application or viewpoint, dividing researcher communities. For this reason, several works have proposed hybrid representations with the goal of mitigating drawbacks while keeping benefits of both types of representations.

Aberman et al. [3] presented a novel data-driven approach for retargeting motions between homeomorphic skeletons (see Section 2.4.2) along with an interesting and elegant representation of human motion illustrated in Figure 2.2. In this work, both

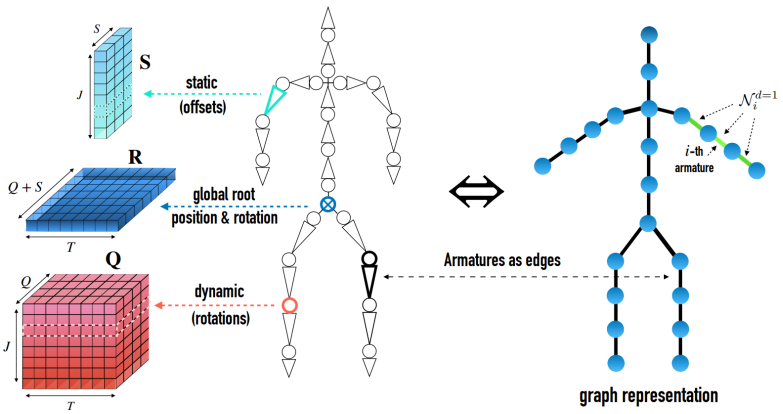


Figure 2.2 – Representation of skeletal motion data as a graph proposed by Aberman et al. [3]. The nodes of the graph correspond to joints and the edges to armatures. Each of the J armatures holds a time-varying tensor Q modelling the temporal sequence of rotations at its corresponding joint, and a time-independent vector S modelling the bone offset to the parent joint. The global motion of the root joint R is processed separately. Illustration in courtesy of Aberman et al. [3].

angular and positional information are combined: a static component S consisting of a set of 3D positional offsets describes the skeleton in some arbitrary pose (similar to a T-pose but specific to a pose sequence), while a dynamic component Q specifies the sequence of orientations of each joint along time (with respect to S) represented using unit quaternions. The separation between static and dynamic partial representations enables the authors to design an architecture such that each component is processed in a separate branch. In continuation of previous work of Pavllo et al. [135, 136], Aberman et al. [3] penalised errors in the positional space after performing FK, while also penalising errors in the quaternion space. Following this work, Shi et al. [150] addressed the reconstruction of 3D kinematic skeletons from 2D key points estimated from monocular video while dividing the motion representation into static bone lengths and dynamic joint orientations. To this end, a DNN called *MotioNet* learns to map 2D key points to a symmetric static skeleton, represented by its bone lengths and a dynamic sequence of joint rotations (quaternions) which are then combined through FK to get a full kinematic skeleton.

Finally, the success of multiple recent methods in skeletal character animation mixing different pose representations suggests that DNNs benefit from redundant pose information. In addition to joint orientations, researchers often feed their models with joint positions [93], joint positions and velocities [58, 106, 118, 161, 163, 207] or even joint positions and linear and angular velocities [61].

2.2.2 *Human Motion Datasets*

Another crucial aspect of data-driven approaches is the choice of dataset, from which a DL-based model will learn a deep representation of human motion. In particular, large amounts of high-quality motion data are necessary to constitute so-called benchmark datasets and to provide robust assessment procedures. In this section we introduce a selection of human motion datasets that are the most relevant for skeleton character animation. Although many datasets have been proposed, only a small number of them have been repeatedly exploited and even fewer can be considered as standard bench-

Table 2.1 – Summary of the main datasets presented in Section 2.2.2.

Dataset	URL	Size & Framerate			Data			Availability
		Frames	Framerate	Hours	Joints	Repr.	Misc. Data	
CMU [170]	</>	3.9×10^6	120 Hz	9.1 h	29	angular		Public
HDM05 [127]	</>	3.6×10^5	120 Hz	0.8 h	31	angular		Public
Human3.6M [66]	</>	3.6×10^6	50 Hz	20.0 h	32	angular	RGB+D	On request
Holden et al. [59]	</>	6.0×10^6	120 Hz	13.9 h	21	positional		Public
NTU RGB+D [149]	</>	4.0×10^6	30 Hz	37.0 h	25	positional	RGB+D+IR	On request
NTU RGB+D 120 [107]	</>	8.0×10^6	30 Hz	74.1 h	25	positional	RGB+D+IR	On request
3DPW [114]	</>	5.1×10^4	30 Hz	0.5 h	23	angular	RGB	Public
AMASS [111]	</>	1.8×10^7	60–250 Hz	41.5 h	52	angular	Body mesh	Public
Mixamo [5]	</>	2.7×10^5	30 Hz	2.5 h	52	angular	Body mesh	Public

marks. Table 2.1 provides relevant information about these datasets most commonly used in the works presented in this chapter.

The two most widely used databases in skeleton-based deep human animation are CMU [170] and Human3.6M [66]. Both are standard large-scale human motion datasets for learning and evaluation. Despite the fact that the CMU dataset was released about a decade earlier than Human3.6M, they have a comparable size (see Table 2.1). The main advantage of Human3.6M over CMU is the presence of RGB+D videos synchronised with human pose sequences, making it sometimes more suitable for tasks closer to computer vision such as motion prediction, even though CMU is also often leveraged.

Beyond these two standard human motion databases, a few others are noticeable, e.g. for the types of motion they contain, for the annotations that are included, or even for the environment in which they were captured. A few years after the release of CMU, Müller et al. [127] proposed HDM05, a public well-documented database of systematically recorded motion capture data. Complementary to CMU which contains a large number of diverse motion sequences, HDM05 is composed of a limited number of specific motion sequences (one hundred) which were executed from 10 to 50 times by five actors. As an example, a cartwheel starting with the left hand has been performed 21 times. More recently, Shahroudy et al. [149] proposed NTU RGB+D, one of the largest datasets for 3D skeleton-based action recognition. It contains 60 action classes as well as RGB and infrared (IR) videos and depth map sequences (D) synchronised with motion sequences. Later on, Liu et al. [107] added another 60 classes to constitute NTU RGB+D 120, roughly doubling the size of the dataset. However, the motion

sequences in these two datasets are represented only by joint positions which limits their use in skeleton-based human animation. Since motion capture systems need dedicated environments e.g. for a multicamera setup, human motion datasets are mostly captured in the lab, resulting in a lack of in the wild data. To this end, Marcard et al. [114] proposed a pose estimation method, leveraging inertial measurement units in addition to a hand-held camera, accurate enough to capture a new dataset called 3DPW consisting of human pose sequences in the wild synchronised with RGB videos. It contains challenging sequences including walking in the city, going up-stairs, or taking the bus for a total of more than 51000 frames.

A special need of data occurs from the use-case of retargeting (see Section 2.4.2), which consists in transferring the movements of a character to another one with a different morphology, i.e. motion data with diversity among morphologies. Unfortunately, most human motion datasets feature only a very limited number of subjects with minor morphological differences. As a result the so-called Mixamo Dataset [5] is often used to train or evaluate models for retargeting. Indeed Mixamo is a company developing services for 3D character animation including downloadable animation sequences performed by numerous 3D character models with varied morphologies. These animations were created using motion capture and cleaned up by key frame animators [1].

Finally, existing datasets were also gathered to build larger human motion databases. Holden et al. [59] constructed a dataset by collecting CMU [170], HDM05 [127], MHAD [130] and Xia et al. [190] in addition to internal motion capture sequences. This data was retargeted to a uniform skeleton structure and resampled to 120 frames per second. More recently, Mahmood et al. [111] proposed AMASS, which unifies 15 different optical marker-based motion capture datasets (including CMU [170], HDM05 [127], SFU [171], HumanEva [154]). The size of AMASS, initially around 42 hours of data, is still increasing. Motion sequences in AMASS are parameterised using the Skinned Multi-Person Linear (SMPL) model [110], a learnt model of human body shape and pose that provides a parameter space from which the skeleton, the joint orientations and the body mesh can be computed.

1. https://en.wikipedia.org/wiki/Mixamo

2.2.3 *Learning Spatial Features*

Besides the pose representation and the dataset, the network architecture can also have a significant impact on the deep representation learned. In this section, we present different types of architectures leveraged to learn spatial correlations in motion data. While most DNNs designed to address tasks related to skeletal character animation do not exploit the prior knowledge we have about geometric and structural aspects of the human skeleton, e.g. its symmetry or its hierarchical structure, a few methods proposed various clever architectures to benefit from this prior knowledge. We present them in the following sub-sections, divided into three categories: spatially-structured architectures, convolutional neural networks (CNNs) and graph convolutional networks (GCNs).

Spatially-Structured Architectures

A first group of approaches to help DNNs learn spatial correlations rely on network architectures structured around the human skeleton, such that the function computed by the network intrinsically encodes human skeleton characteristics. These approaches split the skeleton into body parts and process the corresponding data either in parallel network branches or hierarchically.

In parallel approaches, the main difference is usually related to the targeted task, which conditions the architecture of individual branches. Wang and Neff [182] extracted deep motion signatures with an independent autoencoder (AE) for each branch (i.e. limb or torso) and concatenated the outputs. Guo and Choi [49] relied instead on fully-connected (FC) layers for each branch, whose outputs are merged using a shared layer to predict the next frame. Nakada et al. [128] similarly divided the body into separate modules responsible for controlling muscle activations of body parts from preprocessed common visual information. Both Goel et al. [45] and Men et al. [120] fed long short-term memories (LSTMs) with human body parts in their adversarial motion synthesis framework. Finally, Jain et al. [68] predicted future poses with one recurrent neural network (RNN) per body part, whose inputs are the predictions of neighbouring RNN at the previous timestamps as well as their own previous predictions.

Hierarchical approaches can be either top-down or bottom-up. Wang et al. [178] proposed a spatial encoder that separates the human pose into five body parts, then encodes and merges them two by two recursively. Both Li et al. [102] and Bütepage et al. [23] used a similar top-down approach with a finer human pose split at the input. In contrast, Aksan et al. [7] proposed a bottom-up scheme where the human pose is predicted step by step from the root joint (e.g. pelvis) to the end effectors, i.e. the root is first predicted and then the other joints are recursively predicted using neighbouring predictions as additional inputs.

Convolutional Neural Networks

Another type of architecture sometimes employed to model spatial correlations relies on 2D convolutions, i.e. in the spatial and temporal domains. To this purpose, the skeleton graph is flattened along the spatial dimension. CNNs are particularly efficient at learning spatial correlations in data whose structure is regular such as images. However, learning the spatio-temporal dynamics of human joints remains challenging with CNNs because the graph structure of the human skeleton cannot be meaningfully flattened along a single dimension. To capture the spatial correlations of joints from different limbs, Li et al. [95] proposed to enlarge the convolutional kernels in the spatial domain. More recently, Zang et al. [203] proposed to adaptively model the spatial correlations with deformable convolutional kernels whose relative positions of the entries are learned. The problem of learning patterns in irregular data structures like human poses with CNNs can be overcome by extending convolutions to graph-structured data, as we will see next.

Graph Convolutional Networks

To leverage CNNs while properly handling the graph-structure typically used in animation to represent skeletons, GCNs, an extension of CNNs, have been recently considered in different frameworks working on human motion data. GCNs come in two different flavours [21]. Spatial approaches map neighbourhoods of each node in the graph to Euclidean patches on which a convolution is applied. Spectral approaches operate in the Fourier domain of the feature signals sampled on the graph, which

depends on the graph Laplacian operator [153]. By analogy with the convolution theorem, convolution filters are defined as spectral coefficients that are multiplied by the Fourier transforms of the signals.

Aberman et al. [3] resorted to a simple implementation of spatial GCNs. The supports of convolution kernels around each joint are defined as d-ring neighbourhoods on the skeleton graph in the spatial dimension and extended to the temporal axis to obtain 2D skeleto-temporal convolutions. The operation of such GCNs is limited to skeletons sharing the same topology. However, the motion retargeting network they proposed includes skeletal pooling/unpooling layers, based on the fusion/duplication of the signals of adjacent edges. The pooling layers bring the input topology to a common primal skeleton on which the core processing is performed. The result is transformed back to the original topology by the unpooling layers. Such a network can cope with any topology that is homeomorphic to the primal skeleton.

Other methods leveraging GCNs relied on the spectral approach proposed by Kipf and Welling [84]. Here, the output F^{l+1} of the graph convolutional layer l fed with a feature signal F^l is formulated as $F^{l+1} = \sigma(\hat{A}F^lW^l)$ where W^l is the tensor of learnable convolution filter weights, $\hat{A}$ depends only on the graph adjacency matrix and σ is a non-linear activation function. Most of the time, the weights of the adjacency matrix are learnt in addition to the convolution filter. Mao et al. [113] built their adjacency matrix from a fully connected graph of joints and thereby simultaneously learnt the motion correlations between joints that are physically connected and joints that are far apart but whose motion are dependent, e.g. hands and feet during walking. Cui et al. [30] objected that this scheme may result in unstable training and separately learnt two graph adjacency matrices, one in which the weights of non-connected joints in the skeleton are forced to zero and another with full joint connectivity.

2.2.4 *Learning Temporal Features*

The temporal dimension of motion data is informative of the nature of the action being performed as well as the way it is performed. In the following sub-sections we

review the approaches taken to model human dynamics, most of which rely either on RNNs or CNNs.

Recurrent Neural Networks

RNNs are neural networks designed to process each timestep of time series one after another, and can thereby handle variable length sequences. They maintain an internal state that captures the temporal context of the signal. RNNs are most of the time based on LSTM or gated recurrent unit (GRU).

LONG SHORT-TERM MEMORY. A common LSTM is composed of a memory cell that remembers values over arbitrary time intervals, and three gates – an input gate, an output gate and a forget gate – to regulate the flow of information into and out of the cell and to avoid a common problem with RNNs known as the vanishing (or exploding) gradient problem. In general, the problem is that the gradients used to update the network weights can become extremely small (or large), either preventing the network from further learning or making the network diverge, respectively. In the case of RNNs, the backpropagation through time heavily relies on the chain rule to compute gradients which exponentially decrease (vanishing problem) or increase (exploding problem) if any weight is greater or smaller than one, respectively.

The memory cell remembers values over arbitrary time intervals, making LSTM effective at capturing both short-term and long-term temporal dependencies. Indeed, LSTMs have proven to be powerful for learning temporal dependencies by achieving state-of-the-art performance in key applications e.g. natural language processing and machine translation. In skeletal character animation, LSTMs have been broadly employed, e.g. in motion synthesis [52, 55, 180, 184], in character control [93, 183, 185], as well as in motion editing [181].

GATED RECURRENT UNIT. As an alternative to LSTMs, GRUs have also been widely used, such as in motion synthesis [10, 19, 202] or in motion editing [69, 176]. GRUs rely on a gating mechanism similar to LSTMs in order to avoid the vanishing gradient problem but have only two gates, a reset gate and an update gate. As a result, GRUs use

fewer parameters and therefore less memory, are computationally less expensive [49] and thus train faster than LSTMs. They can process entire motion datasets [116] instead of training action-specific models. However, as shown by Weiss et al. [186], LSTMs are strictly stronger than GRUs as they can easily perform unbounded counting, while GRUs cannot. Thus, LSTMs seem more accurate than GRUs on longer sequences. In summary, the choice between LSTMs and GRUs depends on the processed data and the considered application.

Besides pure LSTMs or GRUs, extensions [166] or combinations of both [177] have been used to model human dynamics. Bidirectional LSTMs (BiLSTMs) stack two LSTMs running forward and backward, respectively. As a result, temporal information is processed and preserved in both directions, i.e. past and future, which is helpful on certain tasks. For instance, BiLSTMs have been leveraged to synthesise motion sequences [45, 193] or even to refine 3D motion data [99, 101].

Transformer

One of the limitations of RNNs is the difficulty to learn to model correlations in time series occurring between arbitrarily distant pieces of information. Indeed, during training, their recurrent nature imposes to gradients to flow through very long computational paths to model distant correlations. This results in eventually falling back to the vanishing gradient problem initially intended to be mitigated by recurrent connections. In natural language processing, researchers identified this issue and introduced attention mechanisms in recurrent architectures to face the vanishing gradient problem in a first step. In contrast with classical RNNs, attention-based RNNs additionally model correlations between all intermediate input and output hidden states instead of only passing the final hidden states. These skip connections effectively create shortcuts for gradients. In a second time, researchers found even more efficient to solely rely on attention mechanisms and dispense recurrent connections [172], resulting in the transformer architecture, recently becoming very popular.

A typical transformer network is composed of a stack of multi-head attention blocks. Each block starts with a self-attention layer which first maps a sequence of input tokens to three values usually referred to as query Q, keys K and values V, in analogy with

retrieval systems. Then attention scores are computed by matrix multiplication of Q and K and used to weight V. In a multi-head attention layer, this process is done several time in parallel with different set of learnable parameters, and the corresponding results are summed up. Then, attention is followed by a token-wise feed-forward layer to better fit the input for the next attention layer. Finally, both attention and feed-forward layers are residual and layer normalisation [16] is applied on their outputs. Moreover, since transformers contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, information is injected about the relative or absolute position of the tokens in the sequence, e.g. with position encoding.

In skeletal character animation, transformer-based models started to gather attention from researchers essentially in the last three years. In particular, this kind of models have been explored towards motion in-betweening [36, 131, 139] (see Section 2.3.1) and upper-body gestures style transfer [89] (see Section 2.4.3).

Temporal Convolutions

Convolutional neural networks (CNNs) also constitute an alternative to RNNs for learning temporal patterns in motion data. They can be either 1D along the temporal dimension, or use 2D spatio-temporal convolutions. Stacking several convolutional layers can efficiently capture both short and long range temporal patterns since lower and higher layers will capture dependencies between nearby and distant frames, respectively. CNN-based approaches are more computationally efficient than RNN-based ones because they process whole motion segments at once rather than frame by frame, allowing greater parallelism. However, CNN-based architectures often contain elements that do not allow variable length inputs (e.g. a few FC layers after convolutions) limiting their use in practice. As a result, they are more prone to be used in tasks where fixed-length motion sequences are suitable, e.g. motion editing [3, 4, 33, 60, 81, 104, 109, 150, 152, 211] (see Section 2.4).

Miscellaneous

Other approaches to better model the temporal flow of human motions include motion phase representation, spectral decomposition of motion data and spatio-temporal

attention. For instance, several authors investigated the representation and learning of the phase of movements in the context of kinematic character control, which is detailed in Section 2.3.2. In the character controller network proposed by Holden et al. [58], the network weights are computed as a spline function of the phase, whose control points, representing network weights configurations during the human locomotion cycle, are learned. Other works [106, 161, 163, 207] use a gating network instead of the cyclic phase to blend expert weights, resulting in a mixture-of-experts scheme.

2.3 MOTION SYNTHESIS AND CHARACTER CONTROL

Generating new motion sequences is of strong practical interest to the media and entertainment industry. Besides creating realistic and diverse sequences, a key challenge is to be able to control various aspects of the motion with high-level parameters. In this section we distinguish motion synthesis and character control. The former focuses on synthesising new motions both consistent with the distribution of samples in a reference training dataset and sufficiently diverse to capture its variations, optionally conditioned on semantic cues typically pertaining to the character trajectory or motion style. Here we restrict ourself to general-purpose approaches and exclude works targeting application-specific contexts such as the synthesis of gestures from speech or dance animations from music. The latter also strives to generate motions but in an online fashion, dynamically responding to user inputs while satisfying environmental constraints. The needs of responsiveness and the fact that user inputs are not known in advance, unlike conditioning parameters in motion synthesis, make character control approaches very different from motion synthesis.

2.3.1 *Motion Synthesis*

We define motion synthesis as the process of creating perceptually plausible motion sequences with a desired style or expressed emotion for instance. Motion synthesis models are thus capable of generating different motions depending on inputs, e.g.

low-level parameters such as latent variables or high-level parameters such as trajectory. In the following sub-sections, we group approaches to motion synthesis based on the framework they build on, all but one being deep generative models. Indeed, in the scheme proposed by Holden et al. [59] the synthesis process is purely deterministic and driven by high-level cues. Finally, we conclude this section with an additional sub-section treating of a particular case of motion synthesis known as *in-betweening*, where synthesised motions ought to be interpolated between given sparse poses referred to as *keyframes*. We summarise the methods presented in this section in Table 2.2.

Table 2.2 – Summary of the methods presented in Section 2.3.1. Miscellaneous data includes hand-crafted, synthetic, proprietary, unspecified or other public datasets.

Reference	URL	Dataset							Architecture									Repr.
		CMU [170]	Human3.6M [66]	HDM05 [127]	Holden et al. [59]	NTU RGB+D[149]	LAFAN1 [52]	Miscellaneous	FC	CNN	GCN	RNN	Transformer	AE	RBM	VAE	GAN	NF	
Holden et al. 2016 [59]	</>				×					×				×					3D positions
Toyer et al. 2017 [169]						×		×				×							2D positions
Barsoum et al. 2018 [19]	</>		×			×						×					×		3D positions
Yan et al. 2018 [195]	</>		×					×				×				×			3D positions
Du et al. 2019 [35]		×		×				×		×						×			3D positions
Wang et al. 2019 [185]		×						×				×					×		Unknown
Yan et al. 2019 [194]						×		×			×						×		Unknown
Aliakbarian et al. 2020 [10]	</>	×	×									×				×			Quaternions
Henter et al. 2020 [55]	</>	×		×	×		×	×				×						×	3D positions
Wang et al. 2020 [180]	</>							×				×		×			×		Euler Angles
Wang et al. 2020 [184]	</>		×			×						×					×		Unknown
Goel et al. 2022 [45]	</>							×				×					×		3D positions
Li et al. 2022 [98]	</>									×	×						×		6D GSL
Men et al. 2022 [120]								×				×					×		3D positions
Habibie et al. 2017 [51]								×		×		×				×			3D positions
Yu et al. 2019 [200]								×	×										Unknown
Kaufmann et al. 2020 [80]					×			×		×				×					3D positions
Xu et al. 2020 [193]		×						×				×					×		Euler Angles
Harvey et al. 2020 [52]	</>		×				×	×				×					×		Quaternions
Duan et al. 2021 [36]	</>						×	×		×			×						3D pos. & quat.
Oreshkin et al. 2022 [131]	</>						×	×					×						3D pos. & 6D GSL
Tang et al. 2022 [165]			×				×					×				×			3D pos. & rot.
Qin et al. 2022 [139]							×						×						6D GSL

Motion prediction/extrapolation, consisting in extrapolating given motion sequences into the future could also have been considered as part of motion synthesis. However the community around it is oriented toward time series and computer vision, as well as a bit farther from animation, since the analogous problem with image sequence

extrapolation is also of interest in those fields. For this reason, motion prediction is out of the scope of the thesis. Therefore, we do not cover it in the following (see Article 1 for a comprehensive review of DL-based motion prediction in the context of skeletal character animation).

Early Works

Pioneer works [8, 28, 167, 168] in motion synthesis based on deep learning notably relied on restricted Boltzmann machines (RBMs). Initially proposed by Smolensky [158], RBMs build on a parametric expression of the probability density function of the data distribution from which new samples are to be generated. These early works are now outperformed by the more recent approaches described below e.g. variational autoencoders (VAEs) and generative adversarial networks (GANs), and have bee abandoned.

Variational Autoencoders

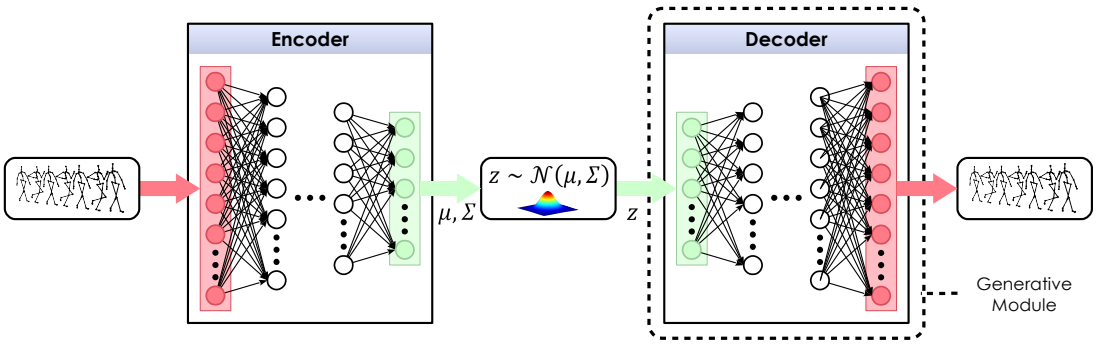


Figure 2.3 – In a variational autoencoder (VAE), a motion sequence is mapped by an encoder to a random latent code that is constrained to follow a prior Gaussian distribution. This code is mapped back by a decoder to the input motion data. Once the full network is trained, feeding the decoder with samples drawn from the prior generates stochastic motion sequences.

Structured as an encoder followed by a decoder, an autoencoder is a DNN whose input and output represent the same data. The encoder output provides an intermediate latent code with a dimension often lower than the input data. Thus, autoencoders provide a scheme for non-linear dimensionality reduction. As illustrated in Figure 2.3, the distribution of the latent codes in a variational autoencoder (VAE) [83] is further

constrained to follow a predefined prior distribution, typically a multivariate normal distribution, endowing the autoencoder with a generative capability. Thus, VAEs provide a convenient way to embed a stochastic component into the generation of human motion.

Various approaches have been proposed to extend the VAE framework to the modelling of temporal sequences. Toyer et al. [169] relied on a Deep Markov Model, essentially a VAE in which the latent code is conditioned on its value at the previous time step. Habibie et al. [51] combined a VAE and an RNN. Motion generation is driven by the random latent code samples, as well as by control variables that constrain the trajectory and velocity of the character. The RNN is conditioned on encodings on these control variables. At inference time, its cell state is initialised with the latent code value. The concatenation of cell state outputs at each time step is fed to the decoder to produce the synthesised motion sequence.

Du et al. [35] built on the motion graph framework proposed by Min and Chai [121], in which motion sequences are represented as a graph of motion primitives. The segmentation of motion sequences into primitives is dependent on the type of motion and typically hand-crafted. Autoencoders learn embeddings for each primitive, and the latent codes for these embeddings are further encoded by conditional VAEs trained on dataset samples for the considered primitives. The conditioning of the primitive-specific VAEs ensures that they reproduce the style of the input motion, which is encoded as a Gram matrix in the embedding space, following prior work in motion style transfer [59] (see Section 2.4.3). To synthesise a motion sequence, a path is determined in the motion graph based on user-defined trajectory controls, and motion primitives are generated along this path using the VAEs.

Yan et al. [195] and Aliakbarian et al. [10] relied on similar network architectures for stochastic motion prediction. Pose sequences are mapped to lower-dimensional features by an encoder and transformed back to motion data by a decoder. The VAE operates on the encoded features, its output is fed to the decoder to synthesise the motion clips. Yan et al. [195] processed small pose sequences called *motion modes* that capture short-term motion features. During training, their VAE maps a pair of (past, future) mode features to a random latent code (encoder) then to a prediction of the

future mode feature (decoder). It thereby captures the transition between the two modes. At inference time, the VAE and RNN decoders generate a stochastic prediction of the future mode, given a past conditioning mode and a draw of the VAE random latent code. Aliakbarian et al. [10] argued that stacking the past sequence information and the random generating seed in a vector and feeding it to the decoding network leaves the possibility that the stochastic component is assigned low weights during training and is thus effectively ignored by the network. This concern is confirmed by experimental evidence. To avoid this, they proposed a *mix-and-match perturbation* strategy and formed a vector by replacing randomly selected components of the past sequence feature by corresponding components of the VAE latent code. Feeding this vector to the decoder forces it to account for both the past context and the stochastic input.

Generative Adversarial Networks

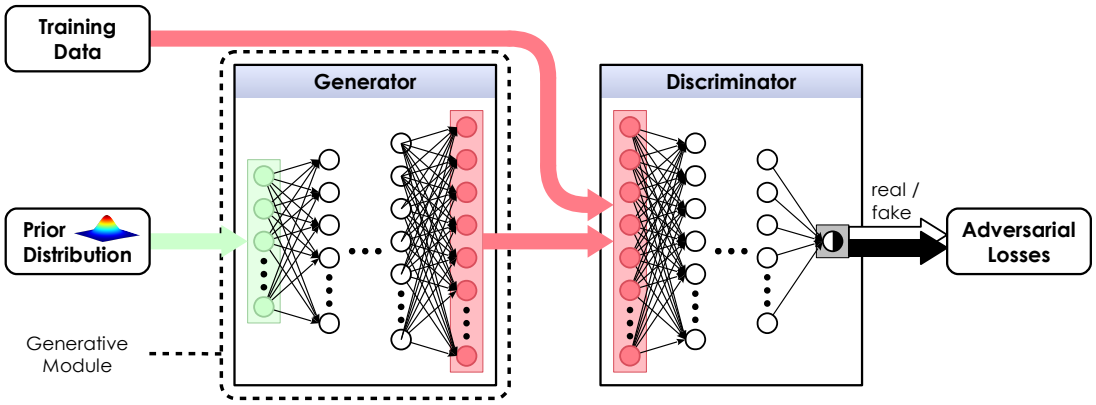


Figure 2.4 – A generative adversarial network (GAN) consists of two networks: the generator produces synthetic motion samples from random seeds, the discriminator tries to differentiate them from real ones drawn from the training dataset. The two modules are trained jointly, aiming at an equilibrium where the generator outputs cannot be distinguished from the training data.

Generative adversarial networks (GANs) are a popular alternative to VAEs for generating samples from random seeds. In a GAN, new samples are synthesised by a generator network that operates in conjunction with a discriminator network (see Figure 2.4). The generator transforms a random seed drawn from a known prior distribution to a

sample which aims to be similar to the contents of the training dataset, the discriminator assesses this similarity. The two networks are trained jointly with adversarial losses, the generator being driven to produce samples that the discriminator ultimately should not be able to distinguish from the training dataset samples. In motion synthesis, Barsoum et al. [19] pointed out that relying on a GAN has intrinsic advantages. GANs do not suffer from the temporal error accumulation issues of RNNs and will produce one instance of all possible outputs instead of averaging these possible outputs. Moreover, GANs also natively embed a stochastic component. Any motion generation network can be wrapped into a GAN by adding a discriminator network to improve the plausibility of the generated sequences [19, 52, 185].

Most other works focused their contribution on optimising the architecture of the GAN generator network to facilitate and improve its training. Wang et al. [180] proposed an adversarial autoencoder architecture [112] where a GAN enforces a prior on the distribution of the latent code of an autoencoder. Yan et al. [194] fed the generator with a sequence of random samples drawn from a Gaussian process with a predetermined covariance function. Through a series of purely convolutional modules made up of a spatio-temporal upsampling layer followed by a graph convolution on the skeleton features, they gradually increased the spatial and temporal resolution of their output to produce a pose sequence. Wang et al. [184] split their generator into separate spatial and temporal sub-networks: the lower layer maps the input random seed and conditioning action label via an RNN to a sequence of low-dimensional latent codes that model temporal transitions. The upper layer decodes each latent code into a skeletal pose. The generator is regularised by a helper action classifier network through a cycle consistency constraint: the conditioning action label fed to the generator should agree with the result of the classification of the motion sequence it produces. Finally, both Men et al. [120] and Goel et al. [45] proposed a GAN-based approach for close interaction synthesis, i.e. synthesising the motion of a character (output) reacting to another character's motion (input). Both approaches rely on an LSTM-based generator with *seq2seq* attention mechanism [17] and fed with separated human body parts, together with an LSTM-based binary and multi-class discriminator which tries to distinguish between fake and real samples as well as between different classes of interaction.

Different from most other approaches, Li et al. [98] learn a GAN-based motion synthesis model from a single motion sequence instead of a large motion dataset. To avoid overfitting the single training motion sequence, the receptive field of the discriminator is limited, following *Patch*-GAN [67, 96]. Moreover, following the progressive approach of Karras et al. [79] with images, motion sequences are generated progressively by upsampling them at each level. Finally, their architecture rely on skeleton-aware operators [3] as a backbone (see Section 2.2.3).

Normalising Flows

Generative models based on normalising flows (NF) [32] synthesise samples by applying a composition of invertible elementary transforms to a latent variable drawn from a known prior distribution. Unlike in GANs or VAEs, owing to the form of the elementary transforms, the probability density function of the generated samples can be computed in closed form. Thus, the generative network can be trained by maximum likelihood optimisation. Henter et al. [55] adapted this framework to the generation of motion sequences. Each transform layer in the generator network is conditioned by a control signal that encodes the past trajectory of the root joint and holds an LSTM unit whose hidden state captures temporal dependencies.

Deterministic Generation

Holden et al. [59] conditioned their motion generation approach on purely deterministic constraints that specify the trajectory and velocity of the character. They leveraged an autoencoder operating on temporal chunks of poses to learn a latent manifold of human motion. A separate feedforward network maps the autoencoder embedding to high-level, semantically interpretable controls of the motion. Generating high-dimensional motion embedding samples from a set of low-dimensional control parameters is severely under-constrained. To disambiguate the generation to the largest possible extent, the scope of the approach is restricted to human locomotion, and a comprehensive set of trajectory and foot contact constraints is imposed.

In-betweening

The task of interpolating motion between sparse character poses, i.e. keyframes, is referred to as in-betweening. In early works [144, 187], spacetime constraints and inverse kinematics were used to interpolate motion between keyframes. Then, probabilistic models including MAP optimisers [25, 122] or Gaussian processes [179] have also been used for in-betweening. However, as pointed out by Harvey et al. [52], these models are specific to action and actors, making combinations of actions looking scripted and sequential. Recent works relied on the scalability and expressiveness of DNNs to tackle this issue.

Yu et al. [200] proposed a simple scheme for interpolating a motion sequence from starting and ending positions of end-effector joints. Their main objective is computational efficiency, in order to generate terrain-adaptive character motion in real time for video games. They cascaded two FC networks: the first interpolates the trajectories of the end effectors in time, the second infers full poses at each frame.

Then multiple works focused on generating plausible and diverse motion over long transitions bounded by a starting and an ending keyframe. Kaufmann et al. [80] leveraged a deep convolutional denoising autoencoder, in which pooling layers ensure large receptive fields to capture long-range spatial and temporal joint correlations. A curriculum learning scheme feeds the encoder with sequences containing increasingly large temporal gaps to improve the training. Xu et al. [193] proposed a temporally hierarchical scheme in which the transition segment is split into equal length subsegments. Trajectory constraints are provided at each sub-segment endpoint. The transition sequence is initialised by sampling a motion clip from the training dataset for each subsegment. Next, the style of each clip is changed to match the style of a reference sequence throughout the whole transition sequence. This stage leverages a motion autoencoder with separate content and style embeddings, that is trained in an unsupervised way. Style transfer is performed by linearly combining the content latent code of the initial clips and the style latent code of the reference style sequence. Finally, transitions between the endpoints of consecutive sub-segments are generated using forward and backward LSTM networks, and the plausibility of the generated sequence is

enhanced by combining the generation network with a discriminator in an adversarial framework. Harvey et al. [52] pointed out that in-betweening between distant keyframes may result in stalling or teleportation artifacts if the temporal evolution of the motion is not monitored during the generation process. To deal with this issue, Harvey et al. [52] relied on an encoder-recurrent-decoder (ERD) architecture [41] that is fed with the pose representation deltas between the current and the target frame, in addition to the character pose at the current timestep and the end pose. The time-to-arrival is encoded in a sine wave and added (similar to positional encoding), rather than stacked, to the latent code that is input to the RNN, forcing the network to take this piece of information into account during training. Later on, Tang et al. [165] proposed a real-time approach to in-betweening, by learning a deep representation of the low-level short-horizon motion dynamics through a conditional VAE modelling the distribution of frames conditioned on the preceding frame, as well as on the next frame hip velocity to disambiguate the next frame. Moreover, a conditional mixture-of-experts scheme is used in the decoder to achieve multi-modal frame transitions. Then, an additional network is learned to sample next frame hip velocity and latent code from previous and target frames and target duration. This enables in-betweening by iteratively conditioning the VAE on the previous frame. According to the authors, 50-frame sequences are too long for the learning to converge, and training sequences were therefore restricted to 25 frames. Another limitation to this approach is that the target frame is not guaranteed to be reached.

One of the major issues of the aforementioned methods is that they struggle to capture long-term correlations. Indeed, autoregressive models are known to accumulate prediction errors while temporal horizons in convolutional architectures are limited by the size of the kernels and the number of layers. To tackle this issue, researchers recently investigated transformers which are known to efficiently model long-term dependencies (see Section 2.2.4). Duan et al. [36] first used a transformer encoder wrapped by 1D temporal convolutional layers. Missing frames between keyframes are first interpolated using spherical linear interpolation before being fed to the model, which is expected to refine interpolated frames and bring global consistency. Oreshkin et al. [131] then also relied on a transformer-based architecture and interpolation of

keyframes. Instead of starting from interpolated keyframes, the proposed network models deltas and in-betweening is then obtained by summing up both deltas and interpolated keyframes. Qin et al. [139] argued that keyframes interpolation can produce unnatural transitions which cannot be fully corrected afterwards. Instead, they proposed a two-stage transfer architecture where the first stage, called *context transformer*, replaces interpolation and produces rough transitions which are then refined in the second stage, the *detail transformer*. Evaluated on LAFAN1 dataset, a standard benchmark in motion in-betweening since first introduced by Harvey et al. [52], this approach significantly outperforms other state-of-the-art methods.

2.3.2 *Character Control*

In this section, we explore the task of controlling character motions that react naturally to user inputs, while accounting for environment constraints, which is another challenge involved in creating believable virtual characters. Character control can itself be sorted into kinematic, physical and biomechanical control. Kinematic approaches directly produce motions as joint angles. In contrast, both physical and biomechanical approaches strive to obey the laws of physics, while differing in the actuation model: physical models are actuated by forces and torques, while biomechanical models (a.k.a. musculoskeletal models) are driven by muscle activations. Moreover, both rely most of the time on deep reinforcement learning which is well-suited when dealing with agents (characters) in a simulated environment. This section only treats kinematic character control, the other two being slightly out of the scope of this thesis (see Article 1 for a comprehensive review of physical and biomechanical character control). We summarise DL-based methods presented in this section in Table 2.3.

Kinematic character control approaches typically produce motions as joint angles, based on a set of motion examples and high-level controls (e.g. user inputs, interactions with the environment). Because of the requirement of generating motions in an online fashion when controlling characters in video games or other interactive applications, RNNs and other autoregressive models are often considered to be more appropriate than CNNs, as the future pose is predicted from the previous motion as well as a

Table 2.3 – Summary of the methods presented in Section 2.3.2. Miscellaneous data includes hand-crafted, synthetic, proprietary, unspecified or other public datasets.

Reference	URL	Dataset		Architecture							Character	
		CMU [170]	Miscellaneous	FC	CNN	RNN	AE	RBM	VAE	GAN	Humanoid(s)	Quadruped(s)
Alemi et al. 2017 [9]			×					×			×	
Holden et al. 2017 [58]	</>		×	×							×	
Wang et al. 2017 [183]		×	×			×					×	
Lee et al. 2018 [93]			×			×					×	
Mason et al. 2018 [118]	</>	×		×							×	
Zhang et al. 2018 [207]	</>		×	×								×
Starke et al. 2019 [163]	</>		×	×							×	
Holden et al. 2020 [61]			×	×							×	
Ling et al. 2020 [106]			×	×					×		×	
Starke et al. 2020 [161]			×	×						×	×	
Starke et al. 2021 [162]			×	×							×	
Starke et al. 2022 [160]	</>		×		×		×				×	×
Mason et al. 2022 [117]			×	×	×						×	

control signal. For instance, Lee et al. [93] used a four-layer LSTM model for controlling characters playing basketball and tennis. However such approaches often tend to fail in the long run, as errors in the prediction are fed back into the input and accumulate, eventually either converging to an average pose or introducing high frequency artifacts. According to Starke et al. [163], these models also often suffer from low responsiveness due to the large variation of the memory state in the case of interactive character control, as the internal memory state is high dimensional.

To overcome these limitations, Holden et al. [58] proposed the use of a specialised architecture called phase-functioned neural network (PFNN), which provides the phase variable to represent the progression of the motion. In their seminal work, the phase is defined based on alternating foot contacts, and used to generate the weights of the regression network at each frame. A trade-off between compactness and runtime speed can then be achieved by precomputing the phase-function for a number of fixed intervals, then interpolating the precomputed elements at runtime. One major limitation of PFNN is that phase functions need to be manually defined, which can be in some cases extremely complex [161, 207]. Zhang et al. [207] therefore proposed to rely on a mixture-of-experts scheme to dynamically compute the weights of a motion prediction network (see Figure 2.5 for an illustration of the general concept). In

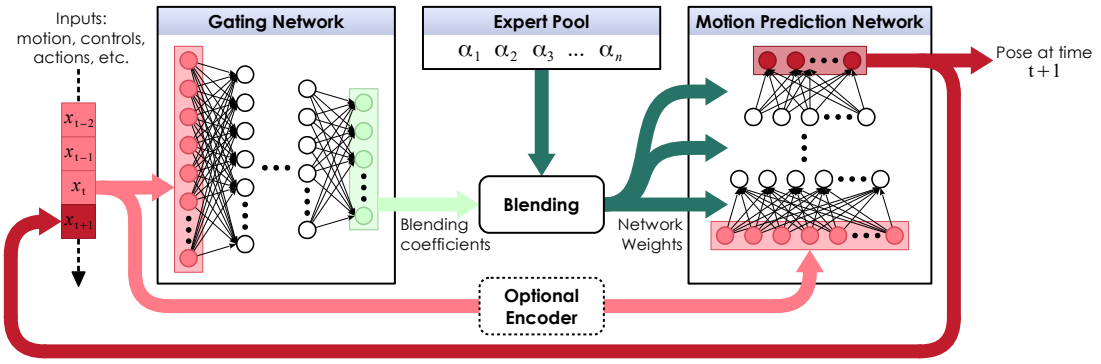


Figure 2.5 – Overall idea of the mixture-of-experts scheme [99, 161, 163, 207]: a gating network first computes blending weights for a number of expert networks, based on a number of features extracted from the current frame x_t. Each expert specialises in a particular movement. They are then blended together to dynamically compute the weights of a motion prediction network, which outputs information relevant to the next frame x_{t+1} including the pose of the animated character. This output is typically fed back into the network (autoregression) at time $t+1$.

their architecture, a gating network first computes blending weights for a number of expert networks, each specialising in a particular movement. This approach was first demonstrated for creating complex quadruped character controllers and then extended by Starke et al. [163] to compute goal-directed series of motions and transitions, while potentially interacting with the environment. The idea was pushed one step further through the use of local motion phases [161], which are defined based on how each body part contacts external objects. Unlike previous approaches where different actions are considered to be synchronised by a single global phase variable, their approach describes each motion by a set of multiple independent and local phases for each bone. It then enables neural networks to learn asynchronous movements of each bone, as well as its interaction with external elements of the virtual environment. Later on, Starke et al. [160] proposed to learn a *periodic autoencoder* from unstructured motion data such that the features of its latent space serve as deep motion phases. Based on a temporal convolutional autoencoder [62], the latent space is further structured by parameterising each of its dimension as a sinusoidal function whose amplitude, frequency, offset and phase shift are learnt, enforcing periodicity encoding. A neural motion controller can then be obtained using a framework similar to the gated mixture-of-experts scheme (see above) [161, 207] but using the deep motion phases as input

to the gating network. Moreover, this approach can also be used in motion matching frameworks, by matching deep motion phases instead of higher-dimensional pose or velocity features of the current character state. Finally, while the previous approaches relied on autoregressive DNNs to generate controlled character motions, Ling et al. [106] demonstrated that a VAE using a similar mixture-of-experts scheme is also viable to produce stable high-quality human motions, while being usable in a deep reinforcement learning context to produce goal-directed motions.

Simultaneously, a few approaches explored the creation of controllable human motions with expressive variations or styles from high-level semantic parameters. For instance, Alemi and Pasquier [9] trained a factored CRBM (FCRBM) on a dataset of motion capture data containing movements from different subjects, expressions, and trajectories. This model can then be used to generate modulated walking movements in real time. Mason et al. [118] explored a similar question from the perspective of generating characters moving in different styles when there is little data available for a new style and proposed a few-shot learning approach. The goal of few-shot learning is to learn (part of) a model able to generalise out of a single or very few examples. In their work, Mason et al. [118] adapted a pre-trained PFNN (modelling style-independent components of the motions), coupled with a set of residual adapters (modelling style-dependent components) learned separately for each new style.

Recently, Mason et al. [117] proposed an improved style modelling system based on the state-of-the-art mixture-of-experts scheme [161] with extended local motion phases. Previously based on feet and hands contacts within the environment, local phases, needed for stylised locomotion, are computed using principal components of cyclic arm movements when contact-free. Then, they introduced a style modulation network which learns to parametrise locomotion style from a stylised reference motion. Style parameters then modulate hidden layers of an autoregressive animation synthesis network, which produces the next frame of motion paced by the local motion phases. Moreover, this framework is trained and evaluated on 100STYLE, a large dataset of stylised locomotion data with about 4 million mocap frames and 100 different styles released alongside. Finally, Starke et al. [162] leveraged a mixture-of-experts with a local motion phases generator into a neural animation layering framework for martial

arts movements. This framework can produce a large variety of novel movements from reference motions and high-level user controls, including combinations of punching, kicking, avoiding as well as close character-interactions. A set of control modules are learnt to predict a set of future body trajectories, which can then be intuitively edited, mixed or layered by animators. The motion generator, fed with those trajectories, will then synthesise a plausible novel animation.

While some of the approaches mentioned above have been demonstrated to be compatible with multi-character control, such character interactions are typically handled by directly including information about the other characters' relative positions [161], or indirectly through information about objects both characters are interacting with and the action to be performed [93, 161, 162]. Wang et al. [183] also proposed to generate character interactions based on the history motion data of both characters, relying on a variant of the ERD architecture [41] to improve animation stability, where multiple LSTM layers constitute the recurrent network. Besides interactions with other characters, interactions with environment have been little investigated up to very recently. Lately, Alvarado et al. [11] proposed a character control framework to handle and anticipate interactions with a variety of obstacles. At the edge between physical and kinematic character control, this approach relies on a hybrid responsive character model following a given reference motion coupled with a lightweight physical upper-body model, making it able to interact with non-rigid objects such as vegetation through physically plausible dynamic responses.

Finally, despite the impressive advances made by the aforementioned methods, traditional animation approaches are still commonly used in animation pipelines to control human characters because of the quality of the motions produced. However, novel approaches, such as *Learned Motion Matching* [61], have recently begun to be explored with the goal of breaking down and replacing individual components of animation algorithms by individual specialised neural networks. They balance the advantages of more traditional approaches with the scalability of neural network based models.

2.4 MOTION EDITING

In the previous section, we looked at how motion data can be synthesised, either through offline generative approaches or interactive frameworks. Another important area of character animation is motion editing, which greatly diversifies the creative capabilities of artists. In this section, we focus on DL-based approaches divided into three topics: motion cleaning, motion retargeting and style transfer. The methods presented in this section are summarised in Table 2.4.

Table 2.4 – Summary of the methods presented in Section 2.4. Miscellaneous data includes hand-crafted, synthetic, proprietary, unspecified or other public datasets.

Reference	URL	Section			Dataset							Architecture								Repr.
		2.4.1 Cleaning	2.4.2 Retargeting	2.4.3 Style Transfer	CMU [170]	Human3.6M [66]	HDM05 [127]	Holden et al. [59]	Mixamo [5]	NTU RGB+D [149]	Miscellaneous	FC	CNN	GCN	RNN	Transformer	AE	RBM	GAN	
Holden et al. 2015 [62]		×			×								×				×			3D positions
Wang et al. 2015 [182]		×			×												×	×		Unknown
Bütepage et al. 2017 [23]		×			×	×						×								$\mathfrak{so}(3)$
Holden 2018 [57]		×			×						×	×								Euler Angles
Li et al. 2019 [101]		×			×										×		×			3D positions
Smith et al. 2019 [156]		×		×							×	×								3D positions
Wang et al. 2019 [185]		×			×						×				×				×	Unknown
Wang et al. 2019 [178]		×			×		×	×			×				×					3D positions
Jang et al. 2020 [69]	</>	×				×									×		×		×	$\mathfrak{so}(3)$
Li et al. 2020 [99]		×			×						×				×		×			3D positions
Shi et al. 2020 [150]	</>	×			×	×							×						×	3D pos. & quat.
Shimada et al. 2020 [152]		×				×					×		×							2D positions
Zou et al. 2020 [211]	</>	×				×					×		×							2D positions
Lohit et al. 2021 [109]		×					×			×			×				×		×	2D positions
Yang et al. 2021 [196]		×			×						×				×					Ad hoc
Villegas et al. 2018 [176]	</>		×			×			×						×				×	Quaternions
Lim et al. 2019 [104]	</>		×						×				×				×		×	Quaternions
Aberman et al. 2020 [3]	</>		×						×				×				×		×	3D pos. & quat.
Kim et al. 2020 [81]	</>		×			×			×				×						×	Quaternions
Li et al. 2022 [100]			×						×				×							3D pos. & quat.
Holden et al. 2016 [59]	</>			×				×					×				×			3D positions
Holden et al. 2017 [60]				×							×		×				×			3D positions
Wang et al. 2018 [181]				×							×				×		×		×	Unknown
Aberman et al. 2020 [4]	</>			×							×		×				×		×	3D pos. & quat.
Dong et al. 2020 [33]				×							×		×						×	Quaternions
Wang et al. 2020 [180]	</>			×							×				×		×		×	Euler Angles
Park et al. 2021 [133]	</>			×							×			×					×	3D pos. & quat.
Jang et al. 2022 [70]				×	×									×						3D pos. & rot.
Kuriyama et al. 2022 [89]				×							×					×	×			$\mathfrak{so}(3)$

2.4.1 *Cleaning*

As mentioned by several authors, projection to and inverse projection from learnt manifolds of human motion can be used to clean motion data, e.g. to perform operations such as denoising, fixing corrupted information or filling in missing motion sequences. Such problems have for instance been explored using RBMs [182], convolutional autoencoders [62], temporal autoencoders [23, 109], spatio-temporal RNNs [178], sequential RNNs [69], BiLSTMs [101], as well as RNN-based GANs [185]. While most approaches typically clean motion data through direct projection and inverse projection (e.g. [62, 182]), then fix residual errors as a post-process, it is also possible to include additional constraints during training. For instance, it is possible to enforce bone length constraints and smoothness by including specific loss functions [99, 101], or to include an additional perceptual loss measuring the difference in high-level features extracted by a pre-trained perceptual autoencoder [99], which improves overall visual quality at the cost of a slight increase in reproduction error. Lohit and Anirudh [109] also proposed to optimise the latent representation to minimise the error between the reconstructed sequence and the correct information of the input sequence, which they applied to filling missing joint trajectories. While most approaches focus on cleaning directly kinematics skeletal data, Holden [57] proposed an approach producing joint transforms directly from raw marker data, in a way which is robust to errors in the input data. This approach is based on learning a deep denoising feedforward neural network using marker locations synthetically reconstructed from motion capture data, where the marker data is corrupted in terms of occlusions and positional shifts.

Footskate

Apart from general-purpose motion cleaning addressed above, researchers have explored footskate cleaning in particular [86]. Artefacts referred to as *footskate* constitute a family of artifacts well-known in character animation and consist in any motion inconsistency between the ground and the feet, e.g. sliding on, passing through or floating above the ground. They are common and easily introduced when processing existing motion data and are known to be very detrimental to the perceived realism

and disturb human perception even at very low intensities [138]. Moreover the problem of footskate is closely related to foot contacts which are necessary to quantify, prevent or fix such artefacts. In particular, most recent approaches remove footskate artefacts from motion sequences by enforcing foot contacts via IK [3, 52, 58, 121, 161, 163]. As a consequence, foot contacts are needed to clean footskate artefacts and foot contact detection is therefore closely related to the problem of footskate.

The most widespread family of approaches in both animation research and industry to extract foot contact labels from motion data relies on simple heuristics with hand-crafted thresholds, applied to velocity and proximity relative to the ground as proposed by Bindiganavale and Badler [20] or Lee et al. [92]. However, such methods lack of temporal precision and are not reliable [91]. Therefore, they are either limited to algorithms that are insensitive to the accuracy of contact labels [132], or require tedious manual checking and corrections [58, 118, 185] to produce faithful contact detection results. Moreover, the accuracy of these approaches often dramatically drops when decreasing motion sequences quality, e.g. increasing noise or artefacts.

Other heuristic approaches have been sparsely investigated, including the work of Le Callennec and Boulic [91]. Stationary or rotating point constraints are computed by solving linear equation systems assuming rigid transformations and a uniform noise pattern, itself estimated from known user-defined constraint(s). While addressing the unreliability of common heuristics-based methods due to the fact that the zero velocity assumption does not hold in the presence of noise, this approach is not fully automatic. Moreover, the noise pattern is assumed to be uniform along time, which is not always true.

Researchers also investigated learned contact detection models. E.g. Ikemoto et al. [64] proposed a k-nearest neighbours (KNN) classifier to determine when the feet should be planted and claimed to be more accurate than heuristics-based approaches. However, a very low amount of manually annotated motion data was used (about 3 minutes, single subject) preventing the approach to generalise well. The proposed KNN classifier achieves 90.78% accuracy over about 33 seconds of hand-labelled motion data, compared to 57.45% and 57% for speed-based and height-based thresholding baselines,

respectively. However, these relatively low accuracies suggest that the corresponding thresholds are not optimal.

More recently, researchers leveraged neural networks to solve this problem. Smith et al. [156] detected foot contact labels using a dedicated MLP to remove footskate artefacts through IK in a motion style transfer pipeline. Zou et al. [211], Shi et al. [150], and Rempe et al. [142] concurrently proposed to leverage foot contact detection to refine human motion estimation. In these works, foot contact labels are detected from 2D key points, themselves estimated from images using *OpenPose* [24], a state-of-the-art 2D pose estimator. Both Zou et al. [211] and Rempe et al. [142] used a dedicated module for contact detection while Shi et al. [150] detected them together with 3D joint rotations and global root positions. Shimada et al. [152] extended the contact detection module from Zou et al. [211] to additionally detect at each frame whether the subject is stationary or not. However, in all these works the foot contact detection modules were trained with ground truth contacts obtained from simple heuristics, as previously described. Tedious manual screening has occasionally been used to correct or label motion sequences, but this approach limits the amount of labelled data which is crucial with neural networks.

Finally, researchers also embedded foot contact detection into their model with the goal of improving foot positioning consistency, but do not support detection from motion data. Yang et al. [196] estimated lower-body pose and foot contacts from upper-body tracking devices. Harvey et al. [52] proposed a motion in-betweening method where pose and contact labels are interpolated between their known values at user-specified keyframes. Likewise, Holden et al. [58] and Starke et al. [163] include foot contact detection at the next frame in character control frameworks, which enables foot contact continuation. Min and Chai [121] modelled foot contacts into a graph-based framework called *Motion Graphs++*, by annotating individual motion primitives with embedded contact information. Then, this framework is able to randomly synthesise motion along with corresponding contacts at runtime.

2.4.2 *Retargeting*

Retargeting [44] refers to the task of transferring the movements of a source character in a skeletal animation sequence to a target character with a different skeleton, i.e. different bone lengths (morphology) and possibly different joints or connections between joints (topology). As pointed out by Aberman et al. [3], there is no formal specification of the task. The purpose at large is to abstract out the dynamics of the source sequence and to reproduce it on a character whose body differ. Effectively, the sought goal is to create retargeted sequences with new morphologies and topologies whose motion mimics the source while remaining visually plausible and natural. Since it is difficult in practice to obtain ground-truth pairs of (source, target) sequences with exactly the same motion, most learning approaches to retargeting, and all the schemes surveyed in this section, rely on unpaired training data without motion correspondences across characters.

In the seminal work of Villegas et al. [176], the retargeting network is built around two RNNs. An encoder RNN captures the motion context of the source sequence in its hidden state and forwards it to a decoder RNN that outputs each frame of the retargeted sequence. A reference pose of the target skeleton is provided to the decoder. Besides regularisation loss terms, network training is driven by an adversarial loss and a cycle consistency loss. The adversarial loss attempts to minimise discrepancies between the joint velocities of ground truth (true) and retargeted (fake) sequences. The cycle consistency loss ensures that a source motion sequence that is retargeted to a target character and then retargeted back to its original character remains as close as possible to the original.

Lim et al. [104] and Kim et al. [81] reported that the above approach tends to generate unrealistic motion. To mitigate this issue, Lim et al. [104] retargeted the motion of the root joint separately from the poses at each time step, and combined the results to construct the output sequence. The retargeted poses are computed as joint angles, represented as quaternions, to be applied to a reference pose of the target skeleton. The cycle consistency loss of Villegas et al. [176] is replaced by a reconstruction loss associated to self-retargeting to the same character. Li et al. [100] then proposed a

similar approach without adversarial loss but using an iterative solution: at runtime, the retargeted motion is iteratively optimised such that source and retargeted motions have the same end-effectors velocities (normalised by kinematic chain lengths). Moreover, a smoothness term between adjacent frames is also used in this iterative optimisation scheme. Kim et al. [81] argued that CNNs are better suited than RNNs for retargeting because they can more accurately capture the short-term motion dependencies that mostly condition the performance of the task. Accordingly, their scheme relies on a purely convolutional network with temporally dilated convolutions that retargets whole motion sequences in one batch.

In the previous approaches, retargeting was done only across different morphologies, only supporting source and target characters with the same skeleton joints. Aberman et al. [3] extended the scope of retargeting to skeletons with different topologies, subject to the condition that source and target topologies are homeomorphic. This implies that they can all be reduced to a common *primal skeleton* by merging pairs of adjacent bones. Their representation of skeletal motion separates the temporally invariant bone offset vectors that define the character morphology from the time-varying bone rotations at each joint that capture the motion dynamics (see Figure 2.2). These two components are processed in distinct parallel branches of the retargeting network. Importantly, a motion sequence is modelled as a graph whose edges correspond to armatures. This provides a principled formalism for processing motion data sampled on the skeletal graph. The retargeting network includes three types of modules: space-time graph-convolutional operators acting on spatio-temporal joint neighbourhoods, graph pooling and graph unpooling operators. Graph pooling merges features of two adjacent edges (armatures) into a single feature. Each graph unpooling operator is designed as the inverse of a graph pooling operator, splitting one edge into two adjacent edges whose features are copied from the original feature. Even though homeomorphic skeletons are supported, a separate autoencoder has to be learnt for each skeletal structure to encode motion sequences into a common latent space expressed in the common primal skeleton topology, and to decode back latent code to motion sequences. Retargeting is achieved by composing the encoder for the source character with the decoder for the target character. The need of a separate autoencoder for each different homeomorphic

skeleton topology prevents this approach to easily generalise to any homeomorphic skeleton topology.

2.4.3 *Style Transfer*

Style Transfer refers to algorithms manipulating data such as images, videos or human animations with DNNs to make the stylistic components look like another data sample. Gatys et al. [43] first introduced a method to perform neural style transfer on images, using the Gram matrix of the deep features as the artistic style information of an image. In human animation, style transfer aims at transferring the style from one motion sequence to another whose content is retained, called hereafter style and content motion sequences, respectively. For example, we might want to edit a particular motion by affecting the state of mind of the character (e.g. enthusiastic, sad, angry) while preserving the performed action, e.g. locomotion from point A to point B. In the following, we split approaches to motion style transfer into two categories, supervised and unsupervised. Both categories are data-driven but the former uses motion sequences annotated with style labels while the latter does not. Human motion databases with style labels obviously facilitate the learning of style transfer frameworks; however they are rare and hence their usage limits the scope and capability to generalise of such frameworks.

Unsupervised

Holden et al. [59] pioneered human motion style transfer using their deterministic generation model (see Section 2.3.1). In this framework, different types of constraints can be applied on the generated motion, such as trajectories, bone lengths or joint positions, solving a constrained optimisation problem in a learnt human motion manifold. Gradients are back-propagated through an autoencoder representing the manifold to optimise latent representations. Style transfer constitutes a particular case, where both joint positions and style are constrained with respect to the content and style motion sequences, respectively. Moreover, Holden et al. [59] followed prior work in image style transfer [43] by using the Gram matrix in the latent representation as a style similarity

measure. They further improved this approach by training a feedforward network to perform style transfer thousands of times faster [60]. Gradients are back-propagated to train the feedforward network instead of optimising the latent representation. More recently Jang et al. [70] proposed to break down style information into local per-body-part style components to enable local style editing and increasing the range of stylised motions. Spatial-temporal GCNs are used into a framework that can be learnt from a motion dataset without labels or paired motion sequences. To do so, four objectives are used: a reconstruction loss on unchanged styles, a cycle consistency loss to preserve the style-invariant features, a third loss to preserve the linear and angular velocities of the root joint, and a smoothness term between adjacent frames.

Supervised

Alternatively, style annotations can also be used to guide the learning of a style transfer model. Smith et al. [156] divided the task of style transfer into spatial and temporal style variations networks, both taking as inputs joint positions as well as a one-hot style vector, where the networks are applied consecutively to predict corresponding style variations. However, this approach requires motion data registered with similar poses in different styles. Wang et al. [180, 181] leveraged an LSTM-based sequential adversarial autoencoder whose encoder learns to map motions to separate content and style encodings: two additional discriminators are trained to recover style labels from content and style encodings, respectively. The encoder tries to fool the former and helps the latter in order to free the content encoding from the style information while preserving it in the style encoding. At runtime, both the content and style motion sequences are encoded into separate content and style embeddings. The decoder combines the content embedding of the content motion sequence and the style embedding of the style motion sequence to produce the style transfer output. Following these works, Aberman et al. [4] also extracted content and style encodings but with two separate temporal convolutional encoders. Furthermore, the style encoder learns a common embedding from both 2D and 3D joint positions with a triplet loss, which enables style extraction from videos. The output motion is synthesised from the content encoding while the style is controlled by the style encoding through temporally

invariant adaptive instance normalisation (AdaIN). Moreover, a multi-style discriminator assesses the output motion style. Park et al. [133] pointed out two limitations of the latter framework: firstly, spatial relations between joints are not considered with 1D convolutions, which might be an issue when style and content motions are significantly different. Secondly, input style in previous works can only be given from an example motion, which is not always available in real-world applications. To tackle the former issue, Park et al. [133] presented a GAN-based framework relying on spatial-temporal GCNs to extract style features in both spatial and temporal domains. In addition, a *mapping* network is learnt to generate style codes sampled from a Gaussian distribution and to match style codes encoded from style motions. At runtime, the space of style codes can be explored through the mapping network and used to generate new style codes without requiring example motions from which to extract style codes, addressing the latter issue.

Beyond general-purpose style transfer, a few particular cases have been explored. Kuriyama et al. [89] investigated the topic of style transfer of expressive or exaggerated upper-body gestures. To this end, a transformer-based autoencoder is leveraged to first embed both content and style gestures into its latent space, where another transformer network called *style transformer* swaps content and style gesture tokens. Finally, Dong et al. [33] focused on translation from adult to child motions and vice versa. In this particular case, retargeting is not sufficient since it does not capture the natural stylistic differences between adults and children [33]. This work leverages the capacity of *CycleGAN* [210] to learn the mapping between adult and child motion distributions without paired training data, which is critical due to the very limited availability of child data.

Whereas most of the aforementioned approaches rely on style labels, Victor et al. [174] recently pointed out that style is still loosely defined and arbitrary categories are too subjective, arguing that style labels are to high-level. As a compromise, they introduced the idea of objective intermediate-level descriptors (e.g. openness of the shoulders), referred to as *pose metrics*, computed single poses and intended to be provided by the user. Then *metric networks* operating into the latent space of a pre-trained pose

autoencoder learn to edit pose metrics (one metric per network) given input and target training pairs.

2.5 CONCLUSION

As presented throughout this chapter, skeletal character animation is a wide topic consisting of a lot of different but closely related problems. As an example, retargeting and style transfer have clearly distinct purposes, while simultaneously being both interested in extracting and transferring high-level semantic information (decoupling the motion from the skeleton and the style from the motion, respectively).

A lot of promising methods have been proposed to create new motions and edit existing motions. Even though deep learning has brought a diversity of novel tools for character animation, current solutions often suffer from artefacts, such as footskate, which are typically mitigated afterward through inverse kinematics for example. In addition, current deep data-driven models lack capacity to generalise, in part due to the difficulty of capturing high-quality and in-the-wild motion data, requiring an expensive motion capture system that not all research teams can afford. Also, large general-purpose motion databases are missing and there is no proper methodology to learn models from multiple smaller, specialised databases, as one can do in computer vision or natural language processing with image and text data, respectively. Indeed, most human motion databases have different skeleton structures and there is currently no satisfactory way to unify them. For these reasons, deep models struggle to integrate workflows in animation studios which still mostly rely on traditional animation approaches.

The goal of this thesis is then to propose effective approaches to enhance workflows based on deep data-driven models and foster their use in animation studios. In particular, we attempt to ease and enhance acquisition, unification and processing of human motion data, three critical aspects to improve state-of-the-art methods as well as to enlarge interoperability of animation systems, pipelines and databases.

3

JUMPS: JOINTS UPSAMPLING METHOD FOR POSE SEQUENCES

Chapter abstract

Human pose estimation is a low-level task useful to easily collect pose and motion data with little equipment compared to motion capture systems, promising for animation of synthetic characters. It also offers interesting perspectives for human action recognition, scene understanding at large, and surveillance. For these applications, and especially the former, accurately estimating the position of many joints from images is desirable for larger high-quality in-the-wild human motion data volumes. In this chapter we present a novel method called JUMPS for increasing the number of joints in 2D pose estimates and recovering occluded or missing joints. We believe this is the first attempt to address the issue. We build on a deep generative model that combines a generative adversarial network to learn the distribution of high-resolution human pose sequences with an encoder that maps the input low-resolution sequences to its latent space. Joints completion and upsampling are then obtained by computing the latent representation whose decoding by the generator optimally matches known joints. Post-processing a 2D pose sequence using our method enhances and enriches representations of the character motion which reduces ambiguity and eases inverse mapping from 2D to 3D, commonly done in 3D pose estimation. We show experimentally that the localisation accuracy of the additional joints is on average on par with the original pose estimates.

Figure 3.1 – Illustration of our method applied to human pose estimation: the yellow 12-joint skeletons (top) depicts poses estimated by AlphaPose [192] at different timestamps. Our approach completes and upsamples joints in a spatially and temporally consistent way to enrich estimated pose sequences, and yields the corresponding green 28-joint skeletons (bottom). In both images, the background has been stitched from a video with a smaller field of view and five images of the jumper at different frames have been overlaid as foreground.

3.1 INTRODUCTION

Human pose estimation refers to the problem of estimating joint positions (either in 2D or 3D) of one or more people in an image or video. In addition, it is appealing in the context of skeletal character animation as an alternative to acquire in-the-wild motion data with little equipment in opposition to traditional motion capture systems which are cumbersome and expensive. It has been a topic of active research for several decades, and all state-of-the-art solutions rely on deep learning, e.g. [15, 24, 164, 206]. Even then, the best approaches extract skeletons with a limited number of joints, usually from 12 to 16, which is too rough for the movie industry or video games applications. This issue concerns both the 2D and 3D cases since 3D joint extraction often relies on 2D pose estimation. Moreover, in the presence of strong foreshortening such left-right

ambiguities, (self-)occlusions, or on previously unseen complex poses, both the 2D pose estimation and the mapping from 2D to 3D are severely underconstrained.

In this chapter we present a novel approach to enhance existing state-of-the-art human pose estimation solutions by upsampling human joints and completing occluded ones, thereby paving the way for downstream applications that require complete poses and higher joint resolutions. Starting with a temporal sequence of partially occluded poses, we recover missing joint locations and improve the resolution of the skeleton animation by estimating the position of additional joints. To the best of our knowledge, no work has been previously proposed to recover missing joints and increase joint resolutions of animated skeletons in 2D. We believe that enriching the representation provide a better understanding of the motion which helps in many cases, especially for extremities such as feet and hands. For instance, extracting the toe in addition to the ankle provides an finer sense of feet contacts.

To this purpose, we draw inspiration from past research on human pose estimation, human motion modelling and image inpainting: we leverage a deep generative model that provides an effective spatio-temporal prior on human motion. Our model builds on a GAN, which we complement by a front-end encoder to learn a mapping from the space of human motions to the GAN latent space. The encoder helps selecting better samples in latent space and stabilises the training of the GAN. At runtime, we leverage our deep generative model to complete and upsample joints through an optimisation in the latent space of our model to best match a 2D pose sequence to be enhanced, typically estimated from video beforehand.

3.2 RELATED WORK

In this section, we overview works related to our approach, i.e. recent approaches for image inpainting in computer vision from which we draw inspiration and then methods to interpolate and extrapolate motion sequences in the context of character animation which are related to the joints completion component of the problem we address.

3.2.1 *Image Inpainting*

Deep generative models have demonstrated impressive performance on image inpainting [18, 63, 198, 199]. For this task, the need to faithfully reproduce the visible surroundings of missing image regions adds an additional constraint to the generative synthesis process. Yeh et al. [198] rely on a GAN [46] whose generator is fed with noise samples from a known prior distribution. In the inference stage, the closest noise sample to the corrupted image is obtained by backpropagating the gradients of the generator network, and fed to the generator to obtain the inpainted image. In their seminal paper, Pathak et al. [134] take a different approach that combines a GAN and a context encoder. The latter network is trained to reconstruct a full image from an input with missing parts. It acts as the GAN generator, while the discriminator enforces the plausibility of the inpainting result. Iizuka et al. [63] enrich this architecture with two discriminators that separately process small-scale and large-scale image textures. Yu et al. [199] further add a self-attention module to better take advantage of distant image patches to fill the missing regions. Bao et al. [18] replace the context encoder with a VAE [83] and incorporate an image classifier to specialise the generative process to sub-categories.

3.2.2 *Motion Prediction and In-Betweening*

As seen in Chapter 2, motion in-betweening refers to the process of temporally interpolating motion between input keyframes that can be manually edited by an artist. By definition, in-betweening completes pose sequences in the temporal domain and usually operates on 3D motion data [51, 52, 139]. More closely related to our work is the approach of Hernandez Ruiz et al. [56], addressing motion prediction through spatio-temporal inpainting. They combine an ERD [41] with a GAN whose generator operates in the latent space of the encoder. Their scheme can both temporally interpolate motion frames and recover missing joints within a pose. Unlike these approaches that are fed with 3D motion data, our scheme takes as input 2D joint locations that could typically be first estimated from video using a 2D human pose estimation framework. We believe

2D pose inpainting is more challenging: although depth is missing, it needs to be accounted for by the inpainting process in order to obtain 2D joint location estimates that are consistent with the motion of the character in 3D space.

3.3 METHOD

In this section we describe the method we propose to upsample and complete a 2D pose sequence, typically previously estimated from video, to infer the location of missing or unseen joints and provide a higher-resolution representation of the body. To this purpose we leverage a deep generative model that we train on pose sequences, rather than static poses, in order to lower ambiguities with spatio-temporal cues.

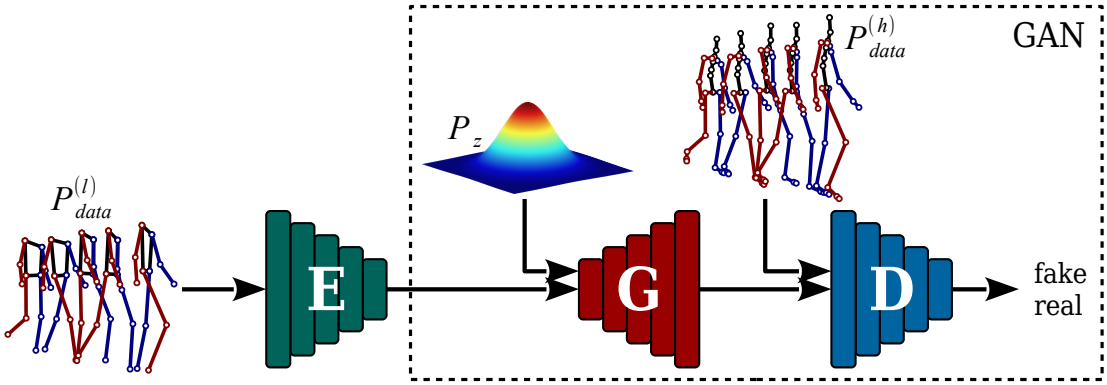


Figure 3.2 – Coarse representation of our deep generative model. The upper right part is the basis of our model: a GAN with generator G and discriminator D operating on motion sequences with higher number of joints. Moreover, we complement our model with encoder E to map motion sequences with lower number of joints to the latent space of the GAN. P_z denotes the prior distribution while $P^{(h)}_{data}$ and $P^{(l)}_{data}$ denote motion data distributions at with higher and lower number of joints, respectively. The latter is sampled by discarding predefined joints from motion sequences sampled from the former.

3.3.1 *Network Architecture*

As illustrated in Figure 3.2, our model consists of a GAN coupled with an additional encoder intended to map pose sequences with lower number of joints in the latent

space of the GAN which represents pose sequences with higher number of joints. Hence, this model benefits from the generative power of GANs and mitigates instability during training by introducing supervision from the encoder.

Our network conforms to the architecture of deep convolutional GANs (DCGANs) [140], using fractionally-strided transposed convolutions in the generator and strided convolutions in the discriminator (see detailed architecture in Section 3.4.1). Moreover, DCGANs use rectified linear units (ReLUs) in the generator and leaky rectified linear units (LReLUs) in the discriminator, and batch normalisations (BNs) are applied after almost each convolutional layer. Except for the size its first and last layers, our encoder follows the architecture of the discriminator.

3.3.2 *Data Representation and Notation*

2D pose sequences represented by joint positions are usually stored in three-dimensional tensors containing the 2D coordinates of each joint at each frame. Instead of temporal 1D convolutional layers, we use 2D convolutional layers operating in the spatio-temporal domain to better model motion dynamics. To obtain meaningful convolutions in the spatial domain, we rearrange the joints as shown in Figure 3.3. In this arrangement,

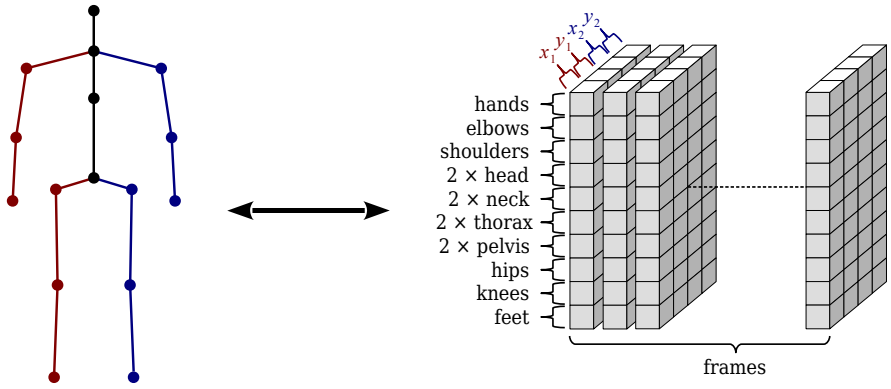


Figure 3.3 – Illustration of how our model internally represents a pose sequence whose topology is depicted on the left part. The right part shows how joint coordinates are arranged, with body parts ordered following the human skeleton from hands to feet. x_1, y_1, x_2 and y_2 denote the 2D coordinates of left and right joints, respectively, for symmetrical joints (e.g. hips) and duplicated 2D coordinates of asymmetrical joints (e.g. pelvis).

each entry holds four coordinates, i.e. the 2D coordinates for two joints. The symmetric joints (e.g. feet, knees, etc.) are paired to form a single entry, while the other joints (e.g. pelvis, thorax, etc.) are duplicated [1] to obtain consistent four-channel entries.

In the following, we note E, G and D the encoder, generator and discriminator networks, respectively. $P_{data}^{(h)}$ is the GAN data distribution, whose samples are pose sequences picked from the training dataset. Then, $P_{data}^{(l)}$ denotes the data distribution used to feed our encoder, which will encode pose sequences with lower number of joints into the latent space of our GAN which represents pose sequences with higher number of joints. It corresponds to $P_{data}^{(h)}$ mapped by discarding predefined joints to obtain pose sequences with lower number of joints. Finally P_z denotes the prior distribution and $P_{\tilde{X}}$ stands for the distribution of uniformly sampled points along straight lines between pairs of points sampled from the GAN data distribution $P_{data}^{(h)}$ and the generator distribution, i.e. P_z mapped through G, used to improve the training of Wasserstein GANs [48].

3.3.3 *Training*

Adversarial Loss

Traditionally, a GAN consists of a generator and a discriminator. The former is trained to produce realistic samples while the latter aims at distinguishing those from real samples, both competing against each other. The ability to generate realistic samples can be expressed more formally as the similarity between two probability distributions that are the data distribution and the distribution of samples produced by the generator. The original formulation of GANs [46] measures the similarity with the Jensen–Shannon divergence. However, this divergence fails to provide meaningful values when the overlap between the two distributions is not significant which often makes GANs quickly diverge during training. Arjovsky et al. [14] introduced Wasserstein generative adversarial networks (WGANs), showing that, under the hypothesis that the discriminator is 1-Lipschitz, the Jensen–Shannon divergence can be replaced by the

1. Both discriminator and encoder duplicate those joints while the generator produces duplicated joints in a first step and then outputs the average of the two versions.

Wasserstein distance that have better properties for convergence. Then, Gulrajani et al. [48] proposed a gradient penalty term in the WGAN loss function to enforce the 1-Lipschitz hypothesis on the discriminator.

Therefore, we opt for the gradient-penalised WGAN and have the following loss functions for the generator and the discriminator, respectively:

$$\mathcal{L}_G = - D\big(G(z)\big) \tag{3.1}$$

$$\mathcal{L}_D = D\big(G(z)\big) - D(X) + \lambda_{gp}\Big(\|\nabla_{\tilde{X}} D(\tilde{X})\|_2 - 1\Big)^2 \tag{3.2}$$

where λ_{gp} is the gradient penalty coefficient, z is a latent code sampled from the pior distribution P_z, and both X and $\tilde{X}$ are pose sequences sampled from $P_{\tilde{X}}$ and $P^{(h)}_{data}$, respectively.

Reconstruction Losses

As previously mentioned, we complement the GAN of our model with an encoder. Then we encourage our model to reconstruct inputs that are encoded and then decoded through the generator with a (forward) reconstruction loss $\mathcal{L}^{forward}_{rec}$ minimising differences between input and output like in autoencoders. We also incite our model to be consistent when generating samples from the prior distribution and then encoding them back into the latent space with a backward reconstruction loss $\mathcal{L}^{backward}_{rec}$, as in cycle-consistent VAEs [71]. Such backward reconstruction loss facilitates the convergence but more importantly it enforces encoder outputs to follow the prior distribution P_z imposed on our GAN. As illustrated in Figure 3.4b, our reconstruction loss is then

$$\mathcal{L}_{rec} = \mathcal{L}^{forward}_{rec} + \mathcal{L}^{backward}_{rec} \tag{3.3}$$

where $\mathcal{L}^{forward}_{rec}$ is itself made up of two terms, respectively minimising the divergences of position and velocity of the reconstructed pose sequence $\hat{X} = G(E(X))$. More

formally, we use the MPJPE [66] to quantify the positional divergence between two pose sequences X and Y:

$$MPJPE(X,Y) = \frac{1}{F}\frac{1}{J}\sum_{f=1}^{F}\sum_{j=1}^{J}\|X_{f,j} - Y_{f,j}\|_2 \tag{3.4}$$

where F and J denotes the number of frames and joints, respectively. In analogy with the MPJPE, we define the mean per joint velocity error (MPJVE) to quantify the velocity divergence between X and Y as:

$$MPJVE(X,Y) = \frac{1}{F}\frac{1}{J}\sum_{f=1}^{F}\sum_{j=1}^{J}\|v(X_{f,j}) - v(Y_{f,j})\|_2 \tag{3.5}$$

where $v(\cdot)$ denotes the velocity of a joint, which is in practice computed via numerical differentiation. This secondary term acts as a powerful regulariser that speeds up the convergence in early iterations and mitigates temporal jitter in generated pose sequences. Finally, $\mathcal{L}_{rec}^{forward}$ is the weighted sum of MPJPE and MPJVE:

$$\mathcal{L}_{rec}^{forward} = \lambda_p \cdot MPJPE\Big(X, G\big(E(X)\big)\Big) + \lambda_v \cdot MPJVE\Big(X, G\big(E(X)\big)\Big) \tag{3.6}$$

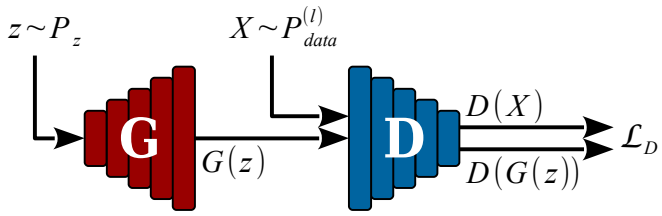
where λ_p and λ_v are hyperparameters and X a pose sequence sampled from $P_{data}^{(l)}$.

The second component $\mathcal{L}_{rec}^{backward}$ focuses on the reconstruction of latent codes sampled from the prior distribution P_z. It minimises the mean squared error (MSE) between $z \sim P_z$ and its reconstructed version $E(G(z))$:

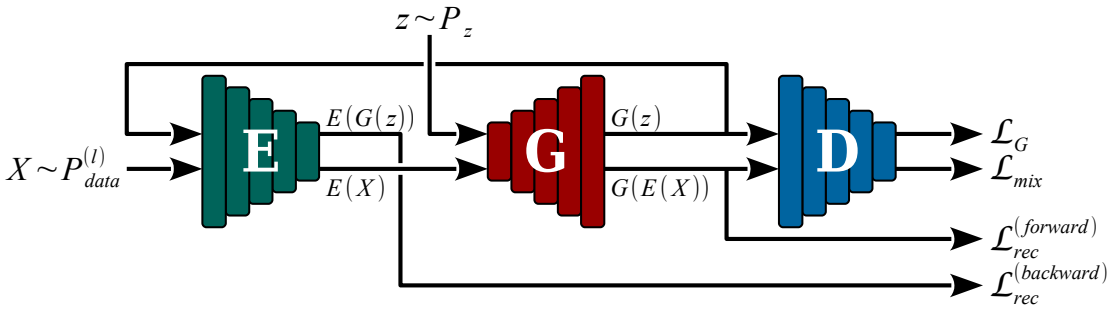
$$\mathcal{L}_{rec}^{backward} = MSE\Big(z, E\big(G(z)\big)\Big) \tag{3.7}$$

Mixed Loss

We further encourage the generation of realistic sequences with an additional loss term to penalise unrealistic reconstructed pose sequences. Here we make use of the discriminator to tell both the generator and the encoder whether the reconstructed



(a) Flow during discriminator's training step.



(b) Flow during generator/encoder's training step.

Figure 3.4 – Computational flows through our deep generative model during training.

pose sequence $\hat{X} = G(E(X))$ is realistic or not. We use the same formulation as for the generator adversarial loss (see Section 3.3.3) but applied to $\hat{X}$ instead of $G(z)$:

$$\mathcal{L}_{mix} = -\lambda_m \cdot D\Big(G\big(E(X)\big)\Big) \tag{3.8}$$

where λ_m is an hyperparameter to balance $\mathcal{L}_{mix}$ with respect to $\mathcal{L}_G$ and $\mathcal{L}_D$, and $X \sim P^{(l)}_{data}$.

Optimisation

In summary, the encoder, the generator and the discriminator are trained by minimising the loss functions $\mathcal{L}_{rec} + \mathcal{L}_{mix}$, $\mathcal{L}_G + \mathcal{L}_{rec} + \mathcal{L}_{mix}$ and $\mathcal{L}_D$, respectively. Similarly to a GAN, we optimise at each iteration the discriminator in a first step and then both the generator and the encoder. Figure 3.4 illustrates the computational flows through the network during both training steps.

Temporal Variance Regularisation

GANs are known to produce sharp samples, but for the considered task this can lead to perceptually disturbing temporal jitter in the output pose sequences. To optimise the trade-off between sharpness and temporal consistency, we feed the discriminator with stacked joint positions and velocities (computed through numerical differentiation). The velocities favour the rejection of generated samples that are either temporally too smooth or too sharp. This idea is inspired from Karras et al. [79], where the variation of generated samples is regulated by concatenating mini-batch standard deviations at some point of the discriminator.

3.3.4 *Inference*

At runtime, we leverage the generative model learnt to recover missing joints in an incomplete pose sequence X and upsample joints to increase the level of details. Given an input pose sequence X, we iteratively optimise the latent code z initialised to $E(X)$ by backpropagating gradients through our generator to minimise the divergence of $G(z)$ from X over available joints. In addition to this *contextual loss term*, we add a *prior term* which maximises the discriminator score on the generated pose sequence. This process is closely related to the semantic image inpainting approach proposed by Yeh et al. [198]. Formally, we solve

$$z^* = \arg\min_z \mathcal{L}_{Inp} \tag{3.9}$$

where $\mathcal{L}_{Inp}$ is our inpainting objective function composed of a contextual and a prior term. The contextual term minimises the weighted sum of MPJPE and MPJVE between input X and output $\hat{X} = G(z)$, while the prior term maximises the plausibility of the output by maximising the discriminator score $D(\hat{X})$:

$$\mathcal{L}_{Inp} = \underbrace{\gamma_p \cdot MPJPE(X, \hat{X}) + \gamma_v \cdot MPJVE(X, \hat{X})}_{\text{contextual loss}} \underbrace{-\gamma_d \cdot D(\hat{X})}_{\text{prior loss}} \tag{3.10}$$

where γ_p, γ_v and γ_d are additional hyperparameters. Then, we finally generate $X^* = G(z^*)$ – which best reconstructs X with respect to $\mathcal{L}_{Inp}$ – to get an enhanced version of input X where missing joints have been recovered and additional joints have been upsampled.

Moreover, we translate, scale and rotate $\hat{X}$ to best match X at each iteration of the gradient descent. Ordinary Procrustes analysis (OPA) is used to analytically find optimal translation, scaling and rotation, and hence has a low overhead while making the gradient descent convergence several times faster and improving results.

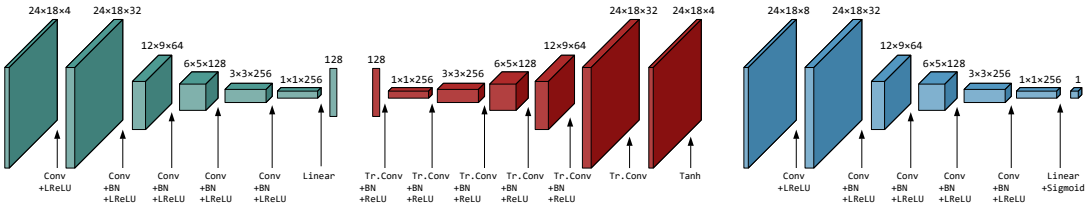
Finally, our network only supports fixed-length pose sequences. Here we describe a simple yet effective mechanism to handle longer variable-length pose sequences. Given a pose sequence X longer than F frames, one can independently inpaint fixed-length sub-sequences of X and then concatenate the results into a single inpainted pose sequence having the same length as X. However, this approach does not guarantee that two consecutive sub-sequences will be concatenated smoothly. To prevent such discontinuities we use half overlapping sub-sequences and for each frame we select between the two overlapping inpainted frames the closest to the input, in the sense of the minimal contextual loss term in $\mathcal{L}_{Inp}$.

3.4 EVALUATION

In this section, we assess the proposed method. First, we provide necessary implementation details in Section 3.4.1. Then, we evaluate our method on pure joint upsampling in Section 3.4.2 and on 2D pose estimation enhancement in Section 3.4.3, i.e. joints completion and upsampling from pose sequences estimated from video.

3.4.1 *Implementation Details*

DATA. To train and evaluate our method, we rely on *MPI-INF-3DHP* [119]. This dataset contains image sequences in which 8 actors perform various activities with different sets of clothing. It is well suited for joint upsampling since it is one of the



(a) Encoder (b) Generator (c) Discriminator

Figure 3.5 – Detailed description of our network architecture. Notations: Convolution (Conv), Transposed Convolution (Tr.Conv), Batch Normalisation (BN), Rectified Linear Unit (ReLU) and Leaky Rectified Linear Unit (LReLU).

public databases having the highest number of joints, i.e. 28 joints. Since our method focuses on fixed-length 2D pose sequences, we generated a set of around 835'000 2D pose sequences of $F = 24$ frames by randomly projecting original 3D pose sequences. We also selected around 166'000 of images annotated with 2D poses directly from MPI-INF-3DHP with no preprocessing for testing.

ARCHITECTURE. The architecture of our model is detailed in Figure 3.5. It has about 3 millions learnable parameters that are almost equally distributed over the encoder, the generator and the discriminator.

TRAINING. Our implementation is written in Python and deeply relies on PyTorch. Training and experiments have been executed on an NVidia Tesla P100 GPU. We trained our model for 60 epochs (about 11 hours) with a mini-batch size of 256 using Adam optimisation algorithm [82] with learning rate $\alpha = 0.0001$ and hyperparameters $\beta_1 = 0.5$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. We followed the suggestions for DCGAN from Radford et al. [140] to reduce α and β_1, with respect to Kingma and Ba [82]. As Radford et al. [140], we observed that $\beta_1 = 0.5$ helped to stabilise the training. We set the Wasserstein gradient penalty weight to $\lambda_{gp} = 10$ as proposed by Gulrajani et al. [48], and our loss weights as follows: $\lambda_p = 200$, $\lambda_v = 100$, $\lambda_z = 2$ and $\lambda_m = 1$. We empirically found these values to work well.

INFERENCE. We optimise the latent code again using Adam algorithm with $\alpha = 1$, $\beta_1 = 0.8$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The weights of the inpainting loss terms are set to $\gamma_p = 10$, $\gamma_v = 5$ and $\gamma_d = 15$. We run the optimisation for 200 iterations. These hyperparameter values have been chosen to enforce the convergence of the optimisation in a limited number of iterations and avoid matching noise or imperfections in inputs. These hyperparameter values have been chosen to enforce the convergence of the optimisation in a limited number of iterations and hence avoid to reproduce input noise and imperfections. Finally, to improve inference results we perform several optimisations of the latent code in parallel for a single input, starting from different initialisations. One of these starting points is computed as the output of the encoder fed by the input pose sequence, the others are randomly sampled from the prior distribution. We keep the one closest to the input, in the sense of the inpainting loss $\mathcal{L}_{Inp}$.

ABLATIVE STUDY. We perform our evaluation on different variants of our method in order to show the relevance of the different components of our method. In the first variant considered, *Ours w/o OPA*, we disable the OPA alignment mechanism during inference (see Section 3.3.4) to measure its impact. Then, we trained from scratch our model without its encoder (*Ours w/o ENC.*), falling back to a pure GAN. Finally, we evaluate our method without using our overlapping sub-sequences mechanism described in Section 3.3.4 (*Ours w/o overlap*).

HUMAN POSE ESTIMATION. In the second part of our evaluation (see Section 3.4.3), we apply our method on real-world pose sequence estimates. We rely on *AlphaPose* [2] to perform 2D pose estimation from videos picked from our test set. AlphaPose provides 12-joint estimates that are then enhanced using our method, recovering missing (e.g. occluded) joints and upsampling to 28 joints.

METRICS. Finally, we report our results in the following with the percentage of correct key points normalised with head size (PCKh) [13] and the area under the

2. Implementation based on [40, 97, 192] available at https://github.com/MVIG-SJTU/AlphaPose

curve (AUC) [65] metrics. The PCKh consider a joint as correct if its distance to the ground truth normalised by head size is less than a fixed threshold and the AUC aggregates the PCKh over an entire range of thresholds. We refer to the PCKh with threshold α with PCKh@α and we compute the AUC over the range $[0,1]$ of thresholds.

3.4.2 *Joints Upsampling*

Our first experiment focuses on pure joint upsampling. To do so, we purposely downsample ground truth pose sequences from 28 joints to the 12 joints that are common to our test set and AlphaPose outputs (see Figure 3.7 left), upsample them back to 28 joints using our method, and compare the result to the ground truth. Table 3.1 provides PCKh and AUC obtained by our method and its variants.

Assuming a typical human head size, the main variant of our method achieves a positioning error lower than 2.25 cm for half of the upsampled joints (i.e. PCKh threshold of 0.1) and lower than 11.25 cm for 95% of them (i.e. PCKh threshold of 0.5). Removing the encoder in front of the GAN in our architecture (*Ours w/o ENC.*) substantially degrades performance. Indeed, the encoder regularises the generative process and improves the initialisation of the latent code at inference time, yielding sequences that better match available input joints. Then, Table 3.1 show that our OPA alignment mechanism is critical to the performance of our approach.

Table 3.1 – Quantitative evaluation of our method and its variants on pure joint upsampling. We purposely remove 16 of the 28 joints from ground truth pose sequences, apply our method to recover them and compare the result to the ground truth. We report the percentage of correct key points normalised with head size (PCKh) at different thresholds and corresponding area under the curve (AUC) over $[0,1]$ (see Section 3.4.1). Disabling ordinary Procrustes analysis alignment (*Ours w/o OPA*) and removing encoder (*Ours w/o ENC.*) substantially degrade performance. Higher PCKh and AUC mean better performance.

Model	PCKh@0.1	PCKh@0.5	PCKh@1.0	AUC
Ours w/o OPA	0.0368	0.4384	0.6814	0.3912
Ours w/o ENC.	0.1701	0.8259	0.9678	0.7005
Ours w/o overlap	0.5821	0.9648	0.9962	0.8727
Ours	**0.6096**	**0.9674**	**0.9965**	**0.8803**

Finally, in absence of the overlapping sub-sequences mechanism (*Ours w/o overlap*), the error measured is only slightly increased, but grows at high accuracy levels. However, as illustrated on Figure 3.6, we found that processing overlapping chunks of frames noticeably improves the temporal consistency of the output pose sequence. We

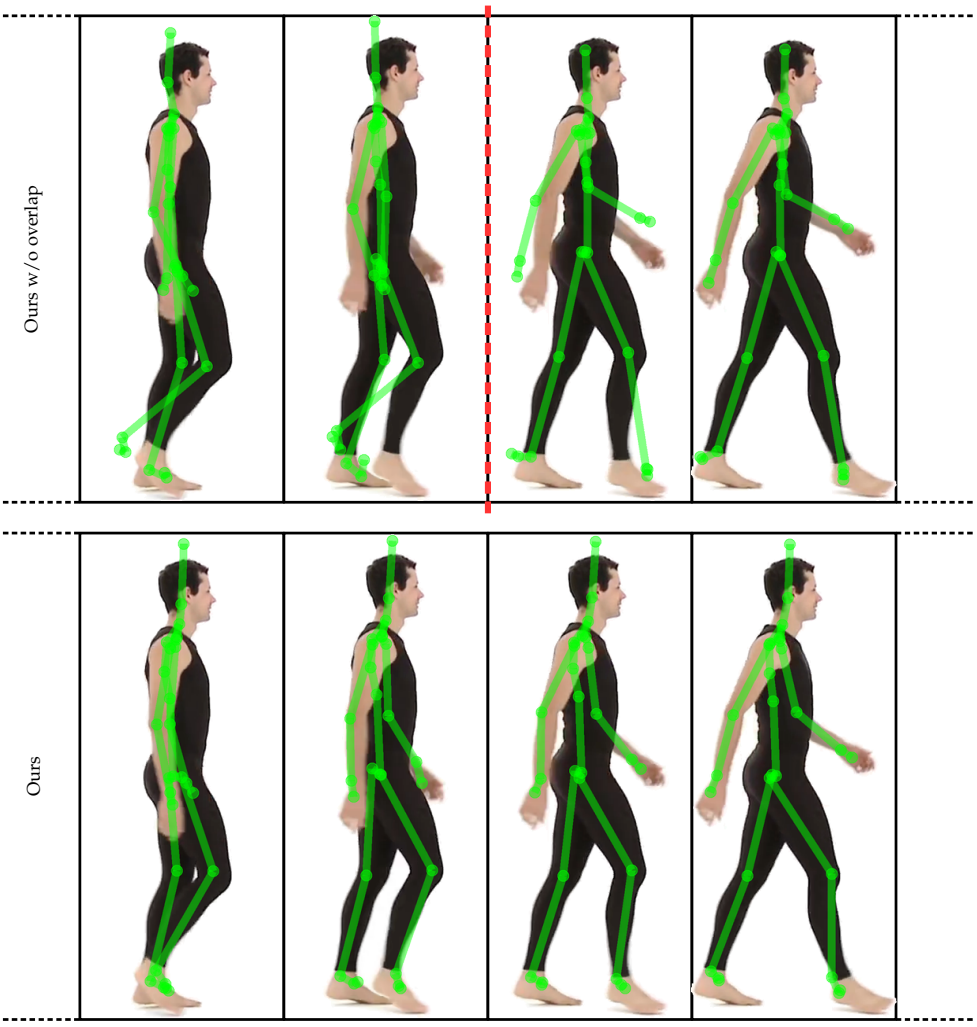


Figure 3.6 – Qualitative example of the influence of overlapping temporal chunks. This subset of four consecutive frames in a longer sequence is inpainted with no overlap (top) and half overlap (bottom). The frames are located at the end (first two) and the beginning (last two) of two consecutive chunks. Note the temporal discontinuity of joint locations (e.g. head top, elbows, ankles) in the sequence inpainted with no overlap at the chunk boundary (dashed red line, top). The temporal consistency is much better with half overlap (bottom).

observed that the per-frame joint positioning accuracy drops at the extremities of the processed chunks, probably because of the reduced temporal context information at the extremities. Without our overlapping sub-sequences mechanism, an increased temporal jitter is introduced at the chunk boundaries of the generated pose sequence, which is likely to incur perceptually disturbing artifacts e.g. when applying our method to character animation.

3.4.3 *Human Pose Estimation*

Our second experiment evaluates the performance of our method and its variants when applied on 2D human pose sequences estimated from videos using AlphaPose. Table 3.2 summarises the results for this experiment. The first row, entitled AlphaPose, gives the amount of error in AlphaPose estimates. By comparing it to our main variant (*Ours*) we see that our method is able to upsample input pose sequences from 12 to 28 joints without significantly increasing joint position errors. Moreover, estimated joint positions with low confidence are sometimes corrected using our method, as illustrated in Figure 3.7. The results obtained by the other variants of our method tend to confirm the conclusions drawn in the previous section.

Table 3.2 – Quantitative evaluation of our method and its variants on joint completion and upsampling applied on 2D human pose estimated from videos using AlphaPose [192]. We report the percentage of correct key points normalised with head size (PCKh) at different thresholds and corresponding area under the curve (AUC) over $[0,1]$ (see Section 3.4.1). The first row reports errors observed in AlphaPose outputs before applying our method. The ablation studies confirm the conclusions drawn from our pure joint upsampling evaluation (see Table 3.1). Higher PCKh and AUC mean better performance.

Model	PCKh@0.1	PCKh@0.5	PCKh@1.0	AUC
AlphaPose [192]	**0.0941**	0.7659	0.9157	0.6310
Ours w/o OPA	0.0207	0.3423	0.6304	0.3249
Ours w/o ENC.	0.0537	0.6801	0.9059	0.5692
Ours w/o overlap	0.0831	0.7701	**0.9277**	0.6326
Ours	0.0842	**0.7723**	0.9276	**0.6341**

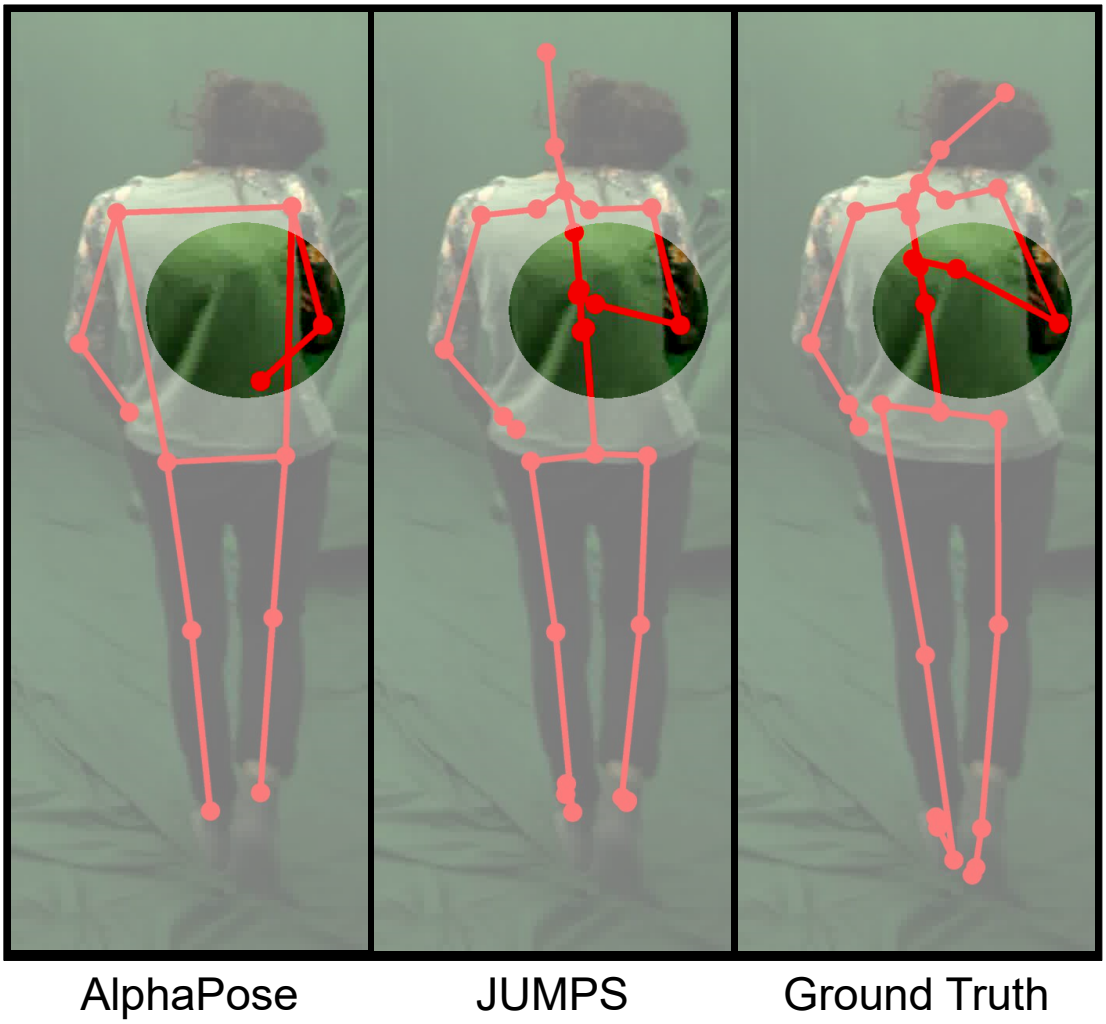


Figure 3.7 – Example of a limb (right forearm) occluded by the subject's body inaccurately estimated by AlphaPose [192] but recovered by our method based on human motion priors. Note that these images have been intentionally whitened except for the area around the occlusion for clarity purposes.

3.5 CONCLUSION

In this chapter we presented a novel method relying on a hybrid adversarial generative model which is able to upsample joints and complete missing joints in 2D human pose sequences. Intended to enhance 2D pose estimates, we achieve to increase joint resolution without of loss accuracy. To the best of our knowledge, this is the first attempt to solve this problem in 2D with deep learning. Our framework considers a 12-joint 2D pose sequence as input and produces a valuable 28-joint 2D pose sequence by inpainting the input. The proposed model consists of a DCGAN coupled to a front-end encoder. Ablation studies have shown the strong benefit of the encoder, since it provides some supervision that greatly helps the convergence and accuracy of the combined model. Given an input sequence, inpainting is performed by optimising the latent representation that best reconstructs the low-resolution input. The encoder provides the initialisation and a prior loss based on the discriminator is used to improve the plausibility of the generated output.

The obtained results are encouraging and open up future research opportunities toward high-quality human pose estimation and increased availability of high-quality in-the-wild human motion data. Improvements could come from an improved consistency with respect to actual human motion. Explicitly modelling or enforcing biomechanical constraints could help. Adapting our method to 3D pose sequences might also benefit from the richer information of 3D joints. Additionally, a potentially fruitful line of research could be to tackle as a whole 3D human pose estimation and joint upsampling. Finally, temporal consistency can be improved. Even though the overlapping sub-sequences mechanism introduced helps to provide smooth output pose sequences, the underlying architecture is still limited to fixed-length sequences. Moreover, CNNs are known to struggle to model long-term correlations. Transformers could be a promising candidate to better model long-term dependencies and handle variable-length sequences. Indeed, as explained in Section 2.2.4, transformers have recently been successful in natural language processing, in particular to efficiently learn distant correlations, and lately became very popular. In the following chapter, we explore such a transformer model to learn an abstract deep motion representation of variable-length

sequences with variable number of joints, which natively supports joint upsampling as well as other applications.

4

TRANSFORMER-BASED UNIFIED DEEP MOTION REPRESENTATION

Chapter abstract

Motion retargeting is a long-standing problem in character animation, which consists in transferring and adapting the motion of a source character to another target character. A typical application is the creation of motion sequences from off-the-shelf motions by transferring them onto new characters. Motion retargeting is also promising to increase interoperability of existing animation systems and motion databases, as they often differ in the structure of the skeleton(s) considered. Moreover, since the goal of motion retargeting is to abstract and transfer motion dynamics, effective solutions might coincide with expressive and powerful human motion models in which operations such as cleaning or editing are easier. In this chapter, we present a novel abstract representation of human motion agnostic to skeleton topology and morphology. Based on transformers, our model is able to encode and decode motion sequences with variable morphology and topology – extending the scope of retargeting – while supporting skeleton topologies never seen during the training phase. More specifically, our model is structured as an autoencoder and encoding and decoding are separately conditioned on skeleton templates *to extract and control morphology and topology. Beyond motion retargeting, our model has many applications since our representation is convenient to embed motion data from different sources. It might be benefical to a number of data-driven methods, allowing them to combine scarce specialised motion datasets (e.g. with style or contact annotations) and larger general motion datasets for improved performance and generalisation ability. Other applications include motion denoising and joint upsampling.*

4.1 INTRODUCTION

In this chapter, we propose a general approach to learn an abstract representation of motion that is agnostic to the skeleton topology and morphology. Possible applications are diverse and numerous since our representation is convenient to embed motion data from different sources. For example it could be beneficial to apply a number of data-driven methods on both scarce specialised motion datasets (e.g. with style or contact annotations) and larger general motion datasets for improved performance and generalisation ability. Specific applications include motion denoising and *motion retargeting*, by mapping in two steps given motion sequences into our representation, and then to the desired representation.

Motion retargeting is a long-standing problem in character animation. Specifically addressed by Gleicher [44] in 1998, it refers to the problem of transferring and adapting the motion of a source character to another target character. However, it has no formal specification since the goal is to abstract out motion dynamics across different characters, which makes motion retargeting difficult to address. Typical applications in animation pipelines include the creation of novel motion sequences by transferring onto new characters off-the-shelf motions. Motion retargeting could also be very interesting to make existing animation systems and motion databases interoperable. Indeed, they often differ in the structure of the skeleton(s) handled, which are difficult to swap. Finally, finding effective models for motion retargeting might coincide with building expressive and powerful motion representations in which operations such as cleaning or editing are easier.

Motion retargeting has not been much explored compared to other topics in character animation such as generative synthesis of human motion. Nonetheless, a few approaches have been proposed in the last years, exclusively relying on deep learning. In the general case, source and target characters considered in retargeting might differ in the length of their bones (morphological variations) or in which joints and bones they are composed of (topological variations). However, most existing methods only address morphological variations. As an exception, Aberman et al. [3] recently extended this scope to homeomorphic skeletons only and is limited to known skeleton topologies.

Moreover, there is no significant amount of pairs of source and retargeted motion sequences available. In consequence, researchers rely on unsupervised approaches and often additionally introduce various heuristics to disambiguate the learning phase of their model.

To this end, we present in this chapter a novel approach to learn an abstract deep motion representation unified across human morphologies and topologies. Based on transformers, our model is able to encode and decode motion sequences with variable skeleton topology and morphology, extending the scope of retargeting beyond homeomorphic skeletons while supporting skeleton topologies never seen during the training phase. More specifically, our model consists in an autoencoder in which both encoder and decoder follow a transformer-based architecture, support motion sequences with variable number of joints and are separately conditioned on *skeleton templates* to extract and control morphology and topology. Trained on a large amount of data gathered from multiple existing databases with different skeleton topologies, our model successfully abstracts out motion features from morphological and topological features.

Motion retargeting can then be performed through encoding conditioned on the source character followed by decoding conditioned on the target character. Moreover, as our decoder has learnt to produce clean motion sequences, motion denoising can be performed through *self-retargeting* (retargeting with identical source and target characters) of noised motion sequences.

4.2 RELATED WORK

In the following, we give an overview of existing works related to our transformer-based approach for motion retargeting.

4.2.1 *Motion Retargeting*

Motion retargeting at large might be addressed within different representations of human dynamics. For instance, Aberman et al. [2] proposed a 2D approach based on visual features where inputs and outputs are image sequences. Other variants include the retargeting of body shapes which considers the dynamics of body surfaces, and is notably addressed in parallel to [175] or on top of [115] skeletal retargeting. Our work is specifically focused on the latter, i.e. 3D skeletal motion retargeting, which considers motion dynamics of character skeletons as commonly done in character animation.

Before Gleicher [44], techniques specifically addressing skeletal motion retargeting have been little explored. In his work, Gleicher [44] uses different constraints, e.g. on joint locations or body segment orientations, to define specific motion dynamics features that must be preserved or transferred. Retargeting is then performed by numerically solving the corresponding constrained-optimisation problem. About two decades later, researchers in character animation tried again to tackle this long-standing problem, relying on deep learning.

As detailed in Section 2.4.2, recent approaches address only morphological retargeting [81, 100, 104, 176], except for Aberman et al. [3] who address homeomorphic skeleton topologies. Besides the scope addressed, these recent methods mostly disagree on what is the most effective architecture for this task: recurrent, 1D temporal convolutional and spatio-temporal graph-convolutional architectures have all been leveraged.

Indeed, both convolutional and recurrent architecture have advantages and disadvantages, as seen in Section 2.2.4. Theoretically, RNNs can model correlations across any temporal horizons. However, as they process each timestep of time series one after another, training gradients exponentially vanish, limiting the effective maximal

temporal horizon across which correlations are learnt. Moreover, the sequential processing of RNNs is inefficient since parallelism is little exploited compared to CNNs, which are more efficient, allowing for larger data volumes and longer training times. The temporal horizon of CNNs is strictly limited by the size of their receptive field, itself determined by the number of convolutional layers and the size of their kernel. GCNs suffer from the same limitations as CNNs, but might better model spatio-temporal dependencies. To the best of our knowledge, no transformer-based architecture has yet been explored in the context of motion retargeting, even though it is very promising especially for modelling long-term temporal dependencies (see Section 2.2.4).

Beyond network architectures, a wide diversity of training losses have been explored without any consensus emerging. They can be sorted into three categories:

— unsupervised losses, e.g. adversarial, reconstruction and cycle consistency losses;

— retargeting heuristics, e.g. to reproduce end-effector velocities or to bound joint twists;

— regularisation losses, e.g. to foster smoothness or quaternion validity.

Losses in the first category shape the model learnt while the ones in the second increase supervision, translating the difficulty to learn such models from unpaired motion sequences. Losses in the third category are typically used to ease model training and guarantee some properties. For a detailed review of methods cited beforehand, please refer to Section 2.4.2.

In contrast with existing methods, ours gets rid of retargeting heuristics, as we only rely on a reconstruction loss to learn a deep motion representation and a temporal bone lengths consistency loss to enforce constant bone lengths over time, since we use a positional pose representation. Moreover, we leverage transformers which are a promising alternative to better model distant spatio-temporal correlations. Last but not least, our learnt deep motion representation is unified across skeleton morphologies and topologies, pushing the scope of retargeting beyond homeomorphic topologies, and generalises well to novel topologies unseen during training.

4.2.2 *Transformers*

A transformer is a scheme of neural network architecture adopting the mechanism of self-attention, in which the network itself finds out which part of input data it should pay more attention to (see Section 2.2.4 for an introduction to transformers). First introduced in natural language processing, transformers are getting more and more attention in all topics leveraging deep learning. For instance in computer vision, transformers have been leveraged in specialised architectures for images, such as vision transformers (ViTs) [34], and tend to replace CNNs as self-attention layers with sufficient number of heads are at least as expressive as any convolutional layer [29]. In animation, they have been explored for 3D facial animation [26] for instance. In skeletal character animation in particular, solutions to style transfer [89] and motion in-betweening [36, 131, 139] have also been built around transformer architectures.

In our work, we draw inspiration from the architecture proposed by Chandran et al. [27] for 3D shape modelling, who relied on a transformer-based autoencoder to model topology-independent static 3D shapes such as human faces, hand shapes or full human body shapes. Later on, Chandran et al. [26] extended their method to include the time dimension for facial animation, where static facial identity and dynamics are separately encoded such that the model learns a disentangled representation. On our side, we instead adapt our transformer-based architecture to process spatio-temporal sequences of joint positions. To this end, we introduce point-wise 1D temporal convolutions at different places to encode, process and extract human joint trajectories (see next section).

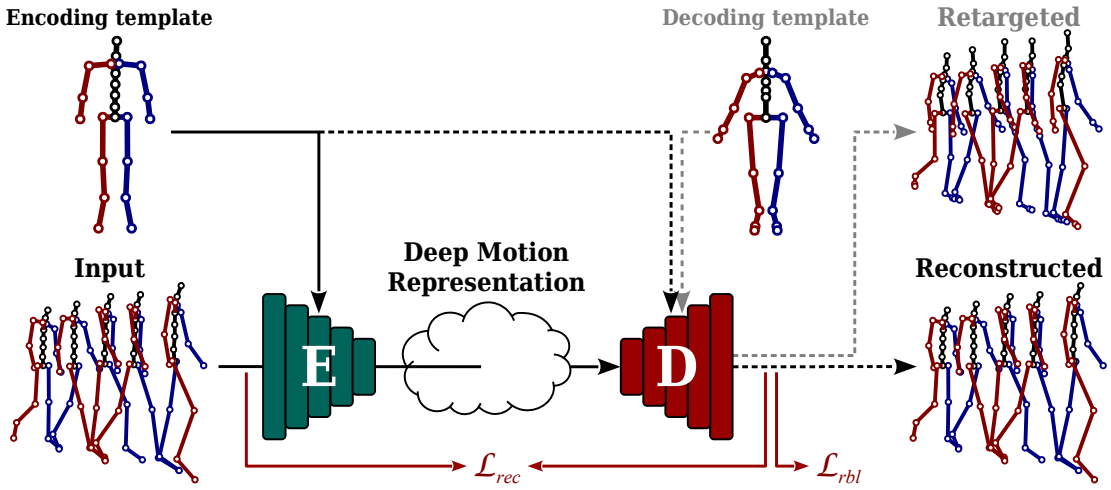


Figure 4.1 – Overview of our model which is a transformer-based autoencoder whose encoder E and decoder D are conditioned on skeleton templates to control morphological and topological features. Skeleton templates consist in neutral poses (see Section 4.3.2). During training, our model is guided by the reconstruction loss $\mathcal{L}_{rec}$ as well as the temporal bone lengths consistency loss $\mathcal{L}_{rbl}$ (see Section 4.3.4). At inference, applications of our model include motion retargeting that can be performed by setting the decoding template to the target character skeleton, as well as motion cleaning by encoding and decoding with the same template as our decoder has learnt to produce clean motion sequences.

4.3 METHOD

In this section, we describe our methodology to build a deep motion representation unified across skeleton topologies and morphologies. Our goal is twofold: first, we need an architecture flexible enough to encode variable size inputs to latent variables lying in our deep representation and then to decode them to variable size outputs. In particular, inputs and outputs are motion sequences with variable number of frames and joints. Second, we not only want our deep representation to be able to encode motion sequences with variable numbers of joints, but also to be *shared* across skeleton topologies and morphologies. In other words, we want similar motions to be represented by close latent codes in our deep representation, even when skeleton topologies and morphologies differ. To this end, we propose a model relying on an autoencoder (see Figure 4.1) in which both encoder and decoder have transformer-based architectures and are conditioned on *skeleton templates* (see below), representing morphological

and topological variations. Our deep motion representation then corresponds to the latent space of our autoencoder and is supposed to encode motion free from skeleton topology and morphology. Hence, we make the hypothesis that "pure" motion features can be reasonably separated from morphological and topological features. After a brief overview of data representations and notations, we present in detail the proposed model in the following: we begin with the description of our conditioning mechanism used to control and abstract skeleton topology and morphology, followed by the proposed associated architecture. Finally, we present the procedure used to train our model and how to exploit it at inference.

4.3.1 *Notation and Representation*

Our deep representation operates on variable-length motion sequences. We represent them by the position of the joints at each frame, expressed in the global Euclidean space. We note $X^{(a)} \in \mathbb{R}^{\mathcal{J}(a)\times 3\times F}$ a motion sequence of F frames "performed" by the skeleton a having $\mathcal{J}(a) \in \mathbb{N}^+$ joints, where the function $\mathcal{J}(\cdot)$ gives the number of joint of a given skeleton. Then $X^{(b)} \in \mathbb{R}^{\mathcal{J}(b)\times 3\times F}$ denote a motion sequence consisting of the same motion features as $X^{(a)}$ but performed by a another skeleton $b \neq a$. Finally, we note E and D our encoder and decoder modules, $E(X^{(a)}|\mathcal{T}(a))$ the encoding of a motion sequence $X^{(a)}$ conditioned on the skeleton template $\mathcal{T}(a)$ associated to skeleton a, and $D(z|\mathcal{T}(b))$ the decoding of latent code z conditioned on the template $\mathcal{T}(b)$ associated to skeleton b, where the function $\mathcal{T}(\cdot)$ gives the template of given skeleton. In the next section we will see what skeleton templates are.

4.3.2 *Template Conditioning*

Our neural network is an autoencoder made up of an encoder E and a decoder D, with our deep motion representation corresponding to the latent space lying in between. A first consequence of our goal to abstract out human motion in the latent space independently of skeleton topology and morphology is the following: no information

about topology and morphology should be encoded into our latent space. Then, the decoding of such skeleton-agnostic latent codes must be conditioned on the desired topology and morphology of the output motion sequence. To this end, we introduce *skeleton templates* as conditioning representatives of topological and morphological features. In the following, the short form "template" is equivalent to skeleton template if not stated otherwise.

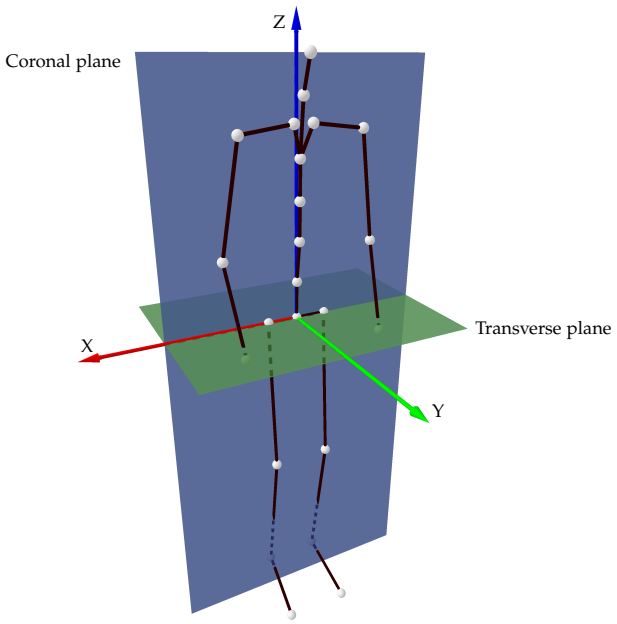


Figure 4.2 – Illustration of a typical *skeleton template* used to condition our model, consisting in a neutral standing pose with arms along the body (i.e. N-pose) with the pelvis at the origin and transverse (green) and coronal (blue) planes aligned with XY and XZ planes, respectively.

In our approach, a skeleton template is a static *neutral* pose, i.e. standing with arms along the body (sometimes referred to as N-pose in animation, in opposition to T-pose and A-pose), which we represent by the position of its joints expressed in the global Euclidean space. Furthermore, we normalise them such that the position of the pelvis is at the origin, the transverse [1] plane is parallel to the XY-plane and the coronal [2] plane is parallel to the XZ-plane. See Figure 4.2 for a typical skeleton template aligned with

1. Anatomical plane that divides the body into superior and inferior sections; see Figure 4.2.
2. Anatomical plane that divides the body into dorsal and ventral sections; see Figure 4.2.

transverse and coronal planes. These skeleton templates explicitly encode topological features through the order and position of their joints while morphological features are implicitly encoded through the relations between joint positions which correspond to the bone lengths. These poses are required to be neutral for our network to be able to compare templates and to register motion sequences with respect to their respective skeleton templates.

Such static neutral poses are straightforward to obtain. Indeed, topological variations generally appear across animation systems or motion databases, while morphological variations are more common and appear between any different characters or persons. Therefore, after manually predefining a single neutral pose per database or animation system considered, the skeleton template corresponding to a given motion sequence is obtained by scaling the bone lengths of the predefined neutral pose according to the bone lengths observed in the motion sequence. Moreover, the simplicity of neutral poses makes their manual editing straightforward.

Finally, in addition to our decoder, we condition our encoder on the very same skeleton templates which are expected to be consistent with input motion sequences to be encoded. Still, both encoder and decoder conditionings are independent and different templates might be used to encode and then decode a given motion sequence, e.g. to transfer a motion from one character to another, i.e. to perform retargeting. As we will see in the following, these templates additionally serve as spatial references when encoding or decoding motion sequences, in a similar way to positional encoding which is critical to transformers performance (see Section 2.2.4).

4.3.3 *Network Architecture*

Our model relies on an autoencoder network based on the recent transformer models and whose latent space constitutes the proposed unified deep motion representation. In this section we describe its architecture in detail. For a general overview of what a transformer is and how it works, please refer to Section 2.2.4. Moreover, to reproduce our approach, please also see implementation details provided in Section 4.4.1.

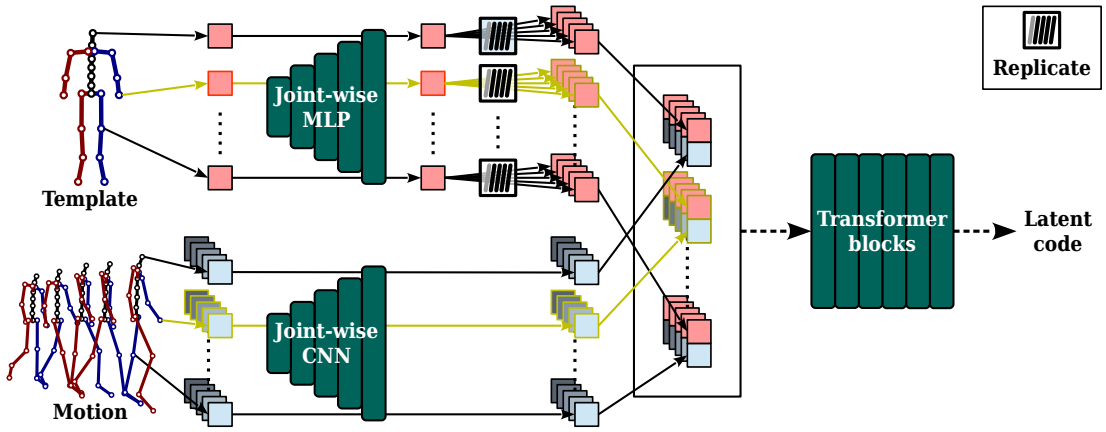


Figure 4.3 – Illustration of the spatial positional encoding used in our encoder, performed via joint-wise concatenation of intermediate features of skeleton template and motion. Hence, for each joint, motion features are registered to template features of the same joint (e.g. left wrist highlight in yellow), telling the network which piece of information is associated to which joint. Thereby, the skeleton template acts as a reference frame for motion features.

Positional Encoding

Inputs and outputs of our model are sequences of joint positions given in an arbitrary order in the spatial domain but in a consistent order across frames. As commonly done in transformer networks, we resort to positional encoding with sine and cosine functions of varying frequencies to give both encoder and decoder knowledge about each frame position in the sequence and hence exploit the temporal structure of motion sequences. To further give information about the spatial structure of input sequences, we rely on another type of positional encoding which consists in concatenating reference features to input features. In our case, this corresponds to concatenating skeleton template features and joint position features, joint by joint, as illustrated in Figure 4.3. In other words, each joint position of an input motion sequence is registered with the same joint in input skeleton template, and hence this mechanism is closely related to our conditioning scheme and to the representation of skeleton templates. Previous work in 3D shape modelling has found this kind of positional encoding to be beneficial [27, 105].

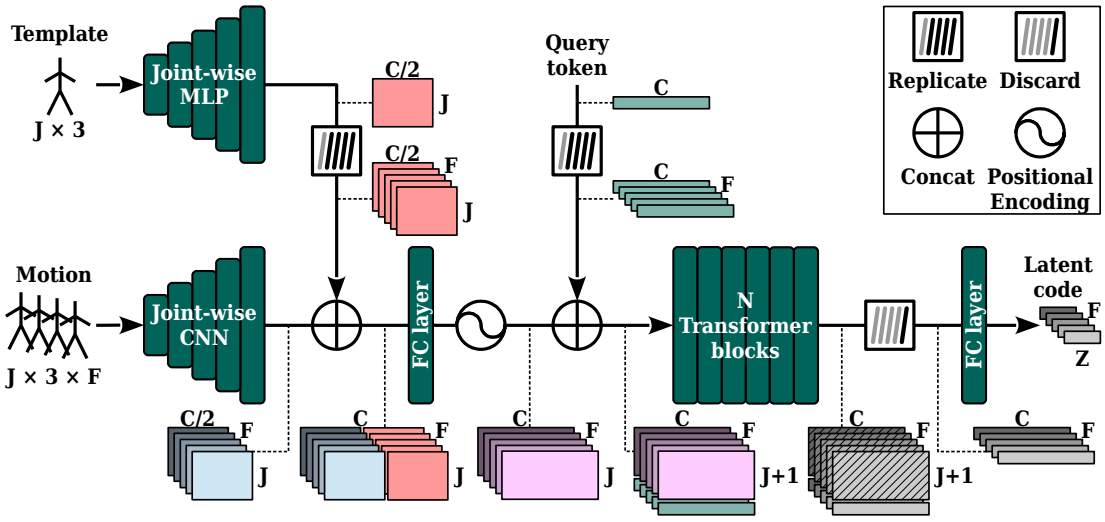


Figure 4.4 – Illustration of our encoder architecture: output latent code (right) is computed from input template (top-left) and motion (bottom-left) through the different network layers (dark cyan rounded rectangles) and other functions (see top-right caption), following black arrows. Coloured rectangular slices depict tensor shapes of template (red), motion (blue), query token (cyan) and latent (grey) features at different points in the network. Colour gradients across slices illustrate varying features over time dimension.

Encoder

As illustrated in Figure 4.4, our encoder is structured as follows: it takes a motion sequence and the corresponding skeleton template as inputs and embeds both of them in respective intermediate representations. The results are merged and fed to a stack of transformer blocks. Both motion sequences and skeleton templates embeddings are performed at each joint independently, i.e. joint-wise, by a 1D temporal CNN and an MLP, respectively. Template embedding is temporally replicated to be concatenated to motion embedding, joint by joint. The result is further processed by a linear layer and modulated by temporal positional encoding to constitute input tokens of the subsequent transformer blocks.

As previously stated, our model is expected to encode a given motion sequence $X^{(a)}$ conditioned on skeleton template $\mathcal{T}(a)$ to $z = E(X^{(a)}|\mathcal{T}(a))$ and might be asked to decode it conditioned on another template $\mathcal{T}(b)$ toward $\hat{X}^{(b)} = D(z|\mathcal{T}(b))$. In such a scenario, both $X^{(a)}$ and $\hat{X}^{(b)}$ will have the same number of frames, but might have

a different number of joints. As a consequence, the whole process conforms to the sequence-to-sequence paradigm, as in original transformers, in the temporal dimension. However, it is not true in the spatial domain as the number of joints is variable across inputs and outputs. This observation coupled with our goal to have latent codes free from skeletal features imposes spatially fixed-size latent codes, and that our encoder and decoder act as sequence-to-item and item-to-sequence functions, respectively, in the spatial domain. To address this issue, we follow Chandran et al. [27] who use an additional learned token, called *query token*, which is appended to input sequences of tokens. First introduced in BERT [31], such additional sequence-level token, sometimes referred to as `[CLS]`, is a common design pattern in transformer architectures. In our particular case, it is used as a fixed-size container for output features while other output tokens of the final transformer block are discarded. During training, the network is thus forced to progressively migrate necessary information toward the query token along transformer blocks.

We use a hybrid approach following Chandran et al. [27] with the query token. We further temporally replicate it to match the number of frames of the input motion sequence to obtain latent codes with fixed spatial size but variable temporal size. More formally, the query token is a tensor of shape $1 \times C$, where C is an intermediate number of features. Given an input motion sequence $X^{(a)} \in \mathbb{R}^{\mathcal{J}(a) \times 3 \times F}$, the sequence of tokens obtained after positional encoding will be a tensor of shape $\mathcal{J}(a) \times C \times F$. Then, the query token is replicated F times, producing a tensor of shape $1 \times C \times F$ and appended to the sequence of tokens, resulting in a tensor of shape $(\mathcal{J}(a) + 1) \times C \times F$, which will be the actual input sequence to the transformer blocks. Finally, at the end of the final transformer block, the first $\mathcal{J}(a)$ slices are discarded to get a tensor of shape $1 \times C \times F$ which is further processed by a linear layer toward the latent code $z \in \mathbb{R}^{Z \times F}$ where Z is the spatial size of the latent space. See Figure 4.4 for a better understanding of how these different tensors are manipulated into our encoder.

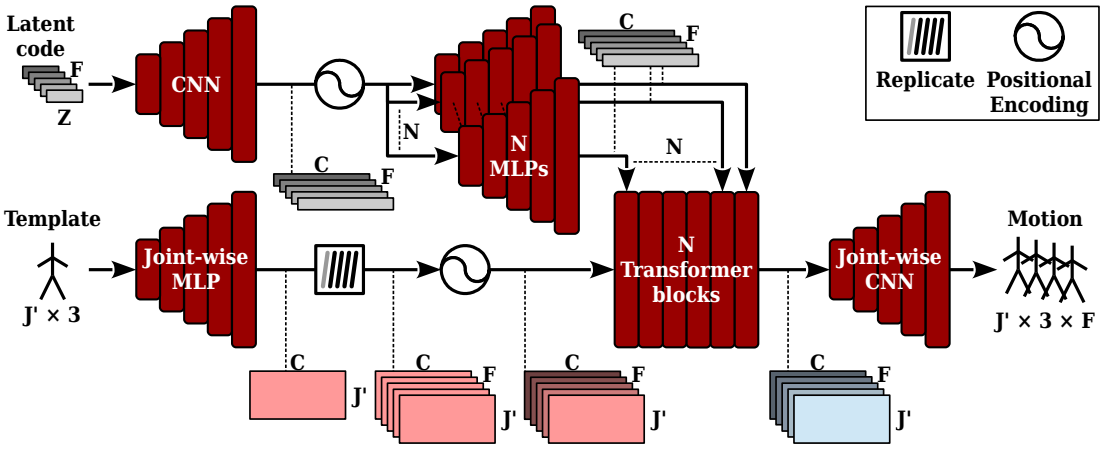


Figure 4.5 – Illustration of our decoder architecture: output motion sequence (right) is computed from input template (bottom-left) and latent code (top-left) through the different network layers (dark red rounded rectangles) and other functions (see top-right caption), following black arrows. Coloured rectangular slices depict tensor shapes of template (red), motion (blue) and latent (grey) features at different points in the network. Colour gradients across slices illustrate varying features over time dimension.

Decoder

As depicted in Figure 4.5, our decoder is structured as follows: it takes a latent code $z \in \mathbb{R}^{Z \times F}$ and a skeleton template $\mathcal{T}(b) \in \mathbb{R}^{\mathcal{J}(b) \times 3}$ as inputs and decodes them toward a motion sequence $\hat{X}^{(b)} = D(z|\mathcal{T}(b)) \in \mathbb{R}^{\mathcal{J}(b) \times 3 \times F}$ of the same temporal length as the latent code z but the same number of joints as the skeleton template $\mathcal{T}(b)$. To do so, the skeleton template $\mathcal{T}(b)$ is first processed at the entrance of the decoder: it is first embedded into an intermediate representation using a joint-wise MLP, then replicated F times to match the length of latent code and finally temporal positional encoding is applied.

The result is used as input tokens to a stack of transformer blocks, as in our encoder. However, these blocks differ from the ones used in our encoder to inject information from the latent code. In particular, we inject information derived from the latent code into each block rather than only as input to the first one. This scheme consisting in disseminating latent information at different points of the network, sometimes called *style modulation*, has proven to be effective when it comes to synthesising new samples, e.g. in the *StyleGAN* series [76–78]. In our decoder, each style-modulated transformer

block corresponds to the same transformer block as in our encoder with an additional *style* module which maps the latent code into a style code. The style code is then used to modulate input tokens of the transformer block through multiplication. Style modules are 1D temporal CNNs, well suited to process latent codes that are temporally structured (see previous section). Most of the first layers of style modules are shared across blocks (see Section 4.4.1).

Finally, the output of the last transformer block is further processed by a joint-wise 1D temporal CNN to get the final motion sequence.

Cross-Covariance Attention

Transformers typically transform a sequence of input tokens into a desired output sequence by modelling token-to-token interactions using self-attention [172]. In this work, we consider motion sequences and model interactions between joint positions at any frames. However, one of the limitations of transformers is the time and memory complexity of self-attention layers which increase quadratically with the number of input tokens [37]. In our model, we instead leverage cross-covariance attention (XCA), recently proposed by El-Nouby et al. [37], which is a transposed version of self-attention with linear complexity with respect to the number of tokens. It operates across feature channels rather than tokens where interactions are based on the cross-covariance matrix between keys and queries (see Section 2.2.4). El-Nouby et al. [37] further proposed a cross-covariance image transformer (XCiT) block adapted from original transformer block with three main changes: first, self-attention layer is replaced with XCA layer; second, a local interaction layer is added in-between attention and feed-forward layers; third, layer normalisation is applied before rather than after each layer of the block. The additional local interaction layer enables explicit communication across tokens (only implicit in XCA layer) and consists in two 2D convolutional layers. Even though our model operates on motion sequences rather than images, we build it on top of the same cross-covariance transformer blocks. Indeed, the only part that might be sensitive to the type of data processed is the local interaction layer made of 2D convolutional layers. Since its goal is to make tokens interact locally, 2D convolutions are as much relevant to image patches as to spatio-temporal motion sequence patches.

4.3.4 *Training*

After describing the architecture of our network in the previous section, we present here the training procedure used to successfully learn the sought unified deep motion representation.

Data Preparation

Basically, the goal of our approach is to learn the structure of the human body from data instead of relying on a single predefined skeleton topology. To this end, we collected a large amount of motion data gathered from UNDERPRESSURE (see Chapter 5), existing public databases (AMASS [111], PSU-TMM100 [148], Human3.6M [66] and MPI-INF-3DHP [119], see Table 4.1) and internal motion data. In total we have about 65 hours of motion data, spanning across 5 different skeleton topologies (see Figure 4.6) and 494 different morphologies.

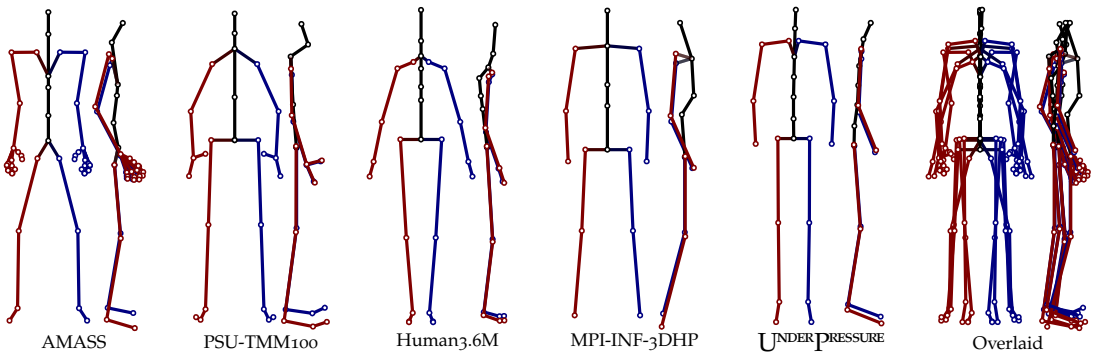


Figure 4.6 – Illustration of the different skeleton topologies present in the dataset we gathered, and their structural variations, represented by the corresponding predefined neutral poses. The left-most five columns show front and side views of the different topologies. Each one is associated to a subset of the data (from left to right: AMASS [111], PSU-TMM100 [148], Human3.6M [66], MPI-INF-3DHP [119] and UNDERPRESSURE [125]) The right-most column gives an insight about topological variations by overlaying the different topologies.

Table 4.1 – Overview of the different datasets gathered to train our model.

Dataset	Frames	Framerate	Size	Joints	Repr.	Availability
AMASS [111]	1.8×10^7	60–250 Hz	41.5 h	52	angular	Public
PSU-TMM100 [148]	1.3×10^6	50 Hz	7.4 h	17	positional	On request
Human3.6M [66]	3.6×10^6	50 Hz	20.0 h	25	angular	On request
MPI-INF-3DHP [119]	1.3×10^5	25 Hz	1.5 h	27	positional	Public
UNDERPRESSURE [125]	4.9×10^6	240 Hz	5.6 h	23	angular	Public

Then, our data preprocessing consists in three steps:

1. We resample all motion sequences to a common framerate (i.e. 30 Hz) using spherical linear interpolation on motion sequences represented with angular pose representations and linear interpolation on motion sequences represented with positional pose representations (see Section 2.2.1).
2. We convert all motion sequences to joint positions expressed in the same XYZ Euclidean space, by switching their axes, converting their coordinates to meters and/or applying forward kinematics (FK) on motion sequences represented with angular pose representations to get joint positions when needed.
3. We extract 1-second overlapping chunks (i.e. 30-frame chunks overlapping over 24 frames, or 0.8 s) and normalise them separately by removing the mean position computed over a set of major joints (i.e. ankles, hips, knees, shoulders, elbows, wrists).

The resulting unified, augmented and normalised motion sequences constitute our dataset, for a total of about 300 hours of motion data.

Training Objective

In an typical autoencoder, inputs are fed to the encoder to produce latent codes which are then decoded, and the resulting differences between inputs and outputs (known as reconstruction loss) is minimised to train the network. We slightly change this scheme to further prevent our model from being tied to one or more particular skeleton topology, and rely on a *stochastic joint subsampling* of input and target motion sequences. Moreover, we use temporal consistency loss over bone lengths. Both objectives are described in the following paragraphs.

RECONSTRUCTION LOSS. The idea of stochastic joint subsampling comes by thinking of motion sequences tied to their skeleton topologies as proxy representations of their underlying motion features. Within this context, we can discard a few joints from motion sequences and still consider their underlying motion features to be the same. Then, given an input motion sequence $X^{(a)}$ sampled from the training set, we first derive two sub-versions $X^{(a_1)} = \mathcal{S}(X^{(a)})$ and $X^{(a_2)} = \mathcal{S}(X^{(a)})$, where $\mathcal{S}(\cdot)$ stands for stochastic joint subsampling and discards a random set of joints. Skeleton template $\mathcal{T}(a)$ is also subsampled consistently with $X^{(a_1)}$ and $X^{(a_2)}$ subsamplings to get templates $\mathcal{T}(a_1) = \mathcal{S}(\mathcal{T}(a))$ and $\mathcal{T}(a_2) = \mathcal{S}(T(a))$, respectively. Hence, we consider these two sub-versions as different proxy representations of the same underlying motion features X. We encode $X^{(a_1)}$ and then decode the resulting latent code $z = E(X^{(a_1)}|\mathcal{T}(a_1))$ conditioned on a_2 to get $\hat{X}^{(a_2)} = D(z|\mathcal{T}(a_2))$. Finally, our reconstruction loss is implemented to minimise the deviation of $\hat{X}^{(a_2)}$ with respect to $X^{(a_2)}$. This mechanism is critical to the training of our model as it forces both encoder and decoder to strictly follow the conditioning template skeletons they are given to encode and decode motion sequences. Indeed, the variations in skeleton topologies seen by our network is much larger. Formally, we compute the deviation of $\hat{X}^{(a_2)}$ with respect to $X^{(a_2)}$ as the mean squared position error:

$$\mathcal{L}_{rec} = \frac{1}{F}\frac{1}{\mathcal{J}(a_2)}\sum_{f=1}^{F}\sum_{j=1}^{\mathcal{J}(a_2)} \|X_{f,j}^{(a_2)} - \hat{X}_{f,j}^{(a_2)}\|_2^2 \tag{4.1}$$

As described in Section 4.3.1 and consistently with the formulation of our reconstruction loss above, input and output motion sequences are represented by their joint positions. However, as explained in Section 2.2.1, positional pose representations do not constrain bone lengths to remain constant over time which is be problematic when synthesising motion sequences. A common approach to solve this issue is to add constraints to ensure that distance between pairs of adjacent joints, referred to as bones, are consistent [99, 101].

TEMPORAL BONE LENGTHS CONSISTENCY. Therefore, we rely on an additional temporal bone lengths consistency loss to train our model such that the motion sequences produced by the decoder have consistent bone lengths over time. It consists in minimising the temporal variance of bone lengths in the decoder output. However, bone lengths have large variations across the different bones and thus minimising absolute temporal variance of bone lengths might mostly fail to constrain short bones. To mitigate this issue, we instead use the absolute temporal variance of bone lengths obtained by normalising, per bone, output lengths with respect to the ground truth lengths before computing the temporal variance. In other words, we compute the temporal variance of output bone length ratios with respect to the true bone lengths. Our temporal bone lengths consistency loss is then:

$$\mathcal{L}_{rbl} = \frac{1}{\mathcal{B}(a_2)} \sum_{b=1}^{\mathcal{B}(a_2)} Var\left(\frac{\hat{X}_b^{(a_2)}}{X_b^{(a_2)}}\right) \tag{4.2}$$

where function $\mathcal{B}(\cdot)$ gives the number of bones of a given skeleton and $X_b^{(a)} \in \mathbb{R}^F$ denotes the vector of lengths of the bth bone at each of the F frames of motion sequence $X^{(a)}$.

In summary, we train our model with two objectives, a reconstruction loss and a temporal bone lengths consistency loss. Moreover, we rely on a stochastic joint subsampling mechanism within our reconstruction loss in order to augment topological and morphological variations seen by our model during training, and hence improve generalisation capability. This reduces the dependency of the network on specific joints of the input topology, in the same way as dropout reduces the dependency on specific nodes of a network. The total loss used to train our model is $\mathcal{L} = \mathcal{L}_{rec} + \lambda_{rbl} \cdot \mathcal{L}_{rbl}$, where λ_{rbl} is a hyperparameter to balance both losses.

4.4 EVALUATION

In this section, we evaluate our approach. First, implementation details necessary for reproducibility are provided in the next sub-section. Second, we quantify the representation accuracy of our deep motion representation. Third, we further show that our deep motion representation is well structured through the evaluation of a side task, i.e. motion denoising. Finally, we evaluate our model on motion retargeting and demonstrate that it achieves state-of-the-art performance.

4.4.1 *Implementation Details*

DATA. To train and evaluate our model, we split our dataset (see Section 4.3.4) into a training set and validation set. The latter is constituted of all chunks from MPI-INF-3DHP [119], to allow validation on unseen skeleton topology, as well as about 10% of chunks from other sub-datasets for validation on known skeleton topologies. This results in about 114'275 seen and 26'249 unseen 30-frames chunks reserved for evaluation.

SKELETON TEMPLATES. As explained in Section 4.3.2, we obtain the skeleton template for a given motion sequence by scaling the bone lengths of the corresponding generic neutral pose. We manually predefined a generic neutral pose for each data source as illustrated in Figure 4.6. Then, given a motion sequence, we scale bone lengths of the corresponding predefined neutral pose to match the bone lengths observed in the given motion sequence. Since some of our sources of motion data use a positional pose representation, the bone lengths are not always constrained and sometimes vary a little over time. In consequence, we use the temporal median of bone lengths of a given motion sequence as observed bone lengths.

ARCHITECTURE. As illustrated in Figure 4.4, our encoder begins with two joint-wise embedding modules: a 3-layer MLP is used to embed skeleton templates (16, 32 and 64 output features) while motion sequences are embedded by a CNN made of 4 1D

convolutional layers (3, 16, 32 and 64 output features) followed by 3 linear layers (64 output features each). Each layer of both modules is followed by exponential linear unit (ELU). Then, the core of our encoder is made of 4 transformer blocks with 8 heads and 128 features each. Finally, intermediate and final linear layers (see Figure 4.4) have both 128 output features. Similarly to our encoder, templates and latent codes are also embedded at the beginning of our decoder (see Figure 4.5). Skeleton templates are embedded by a 4-layer MLP (16, 32, 64 and 128 output features) with ReLU after each layer except the last. Latent codes are embedded by a CNN made of 3 1D convolutional layers followed by a linear layer (128 output features and ELU activation each). Then, the core of our decoder is also made of 4 transformer blocks with 8 heads and 128 features each. Moreover, a 2-layer MLP with 128 output features and ReLU activation is associated to each block to further independently process latent features before modulating input tokens (see Section 4.3.3). After transformer blocks, a final joint-wise CNN is used to reconstruct motion sequences. It is made of 3 1D convolutional layers (128 output features) followed by 3 linear layers (128, 128 and 3 output features), with ELU activation after each layer except the last. As explained in Section 4.3.3, transformer blocks in both our encoder and decoder leverage cross-covariance attention. In total, our model has 2'475'511 learnable parameters.

STOCHASTIC JOINT SUBSAMPLING. As seen in Section 4.3.4, we apply our reconstruction loss on two sub-versions of a given sample motion sequence, in which we discard random sets of joints. In practice, we discard each joint independently with probability $p = 0.1$ for major joints (i.e. ankles, hips, knees, shoulders, elbows, wrists) that are common to all skeleton topologies of our dataset, and with probability $p = 0.5$ for other joints. Moreover, we compute our temporal bone lengths consistency loss on reconstructed subsampled motion sequences. Since some joints have been discarded, we compute this loss over bones whose both joints have not been discarded for increased computational efficiency. Otherwise, it is required to encode and decode the original sample motion sequence in addition to its subsampled version, which roughly doubles the computation time.

TRAINING. Our implementation is written in Python and relies on PyTorch. Training was performed on an NVidia Ampere A100 GPU while other results were obtained either on an NVidia GeForce RTX 2060 GPU or on CPU. We trained our model for 10^7 iterations (about 385 epochs and 26.5 days) using Adam optimisation algorithm [82] with mini-batch size of 32 and hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$. Moreover, we used a triangular cyclic learning rate [157] with a period of 10 epochs and lower and upper bounds set at 10^{-4} and 10^{-5}, respectively. Finally we set the weight of our temporal bone lengths consistency loss to $\lambda_{rbl} = 0.5$ (see Section 4.3.4).

MOTION RETARGETING. In the following, we notably perform a quantitative evaluation of the performance of our model on motion retargeting. This evaluation is performed on motion sequences drawn from Mixamo dataset (see Section 2.2.2), as commonly done for motion retargeting [3, 81, 100, 104, 176]. However, models from other approaches are usually trained on a subset of Mixamo while ours is not. To make the comparison more fair, we fine-tune our model on a few motion sequences from Mixamo before performing our motion retargeting evaluation. The procedure is the same as the training of the model, except that the amount of motion data is far smaller (about 1 hour vs 65 hours for training) and its duration is much shorter (less than 2 hours vs 26.5 days). We also used a constant learning rate set at $5 \cdot 10^{-5}$ instead of cyclic learning rate. Finally, since the pelvises of skeleton templates are aligned (see Section 4.3.2), our model aligns the global trajectory of output motion sequences with respect to input motion sequences around the same region of the body. However, when performing motion retargeting on source and target characters of different scales, the resulting global trajectory might not be consistent with target character scale. To fix this issue, we further scale the global trajectory of output motion sequences when performing motion retargeting based on the leg length ratio of source and target characters, as typically performed in the literature [54, 88]

ABLATIVE STUDY. Throughout our evaluation, we notably show that our joint subsampling mechanism (see Section 4.3.4) is critical to the performance of our method. To this end, we perform a simple ablative study which considers a variant of our approach, called *Ours - no SJS*, in which the same neural network was trained without this mechanism.

4.4.2 *Representation Accuracy*

A key aspect of deep representations is their ability to accurately encode information, sometimes called the representation accuracy. In this section, we evaluate this aspect by quantifying the amount of distortion introduced by our deep motion representation when encoding and then decoding motion sequences drawn from our validation set (see previous section). We measure this amount of distortion using the mean per joint position error (MPJPE) of reconstructed motion sequences with respect to ground truth motion sequences given in input.

In Table 4.2, we provide the corresponding results for both our deep motion representation and its variant without stochastic joint subsampling and for each skeleton topology. Moreover, we separate the evaluation between skeleton topologies that have been seen during training from unseen skeleton topology, i.e. the one associated to MPI-INF-3DHP.

Table 4.2 – Quantitative evaluation of the representation accuracy of our deep motion representation. We measure the distortion introduced when encoding and then decoding ground truth motion sequences drawn from our validation set with the mean per joint position error (MPJPE) in centimetres. We distinguish skeleton topologies that have been seen during training from unseen topologies. Columns are associated to the different skeleton topologies in our validation set, named after the data sources used to constitute our dataset (see Section 4.3.4). Lower MPJPE means higher representation accuracy.

Model	Seen					Unseen
	AMASS	PSU-TMM100	Human3.6M	UNDERPRESSURE	Average	MPI-INF-3DHP
Ours - no SJS	2.54 cm	3.05 cm	2.91 cm	2.93 cm	2.64 cm	4.40 cm
Ours	**1.00 cm**	**1.74 cm**	**1.81 cm**	**1.22 cm**	**1.13 cm**	**3.09 cm**

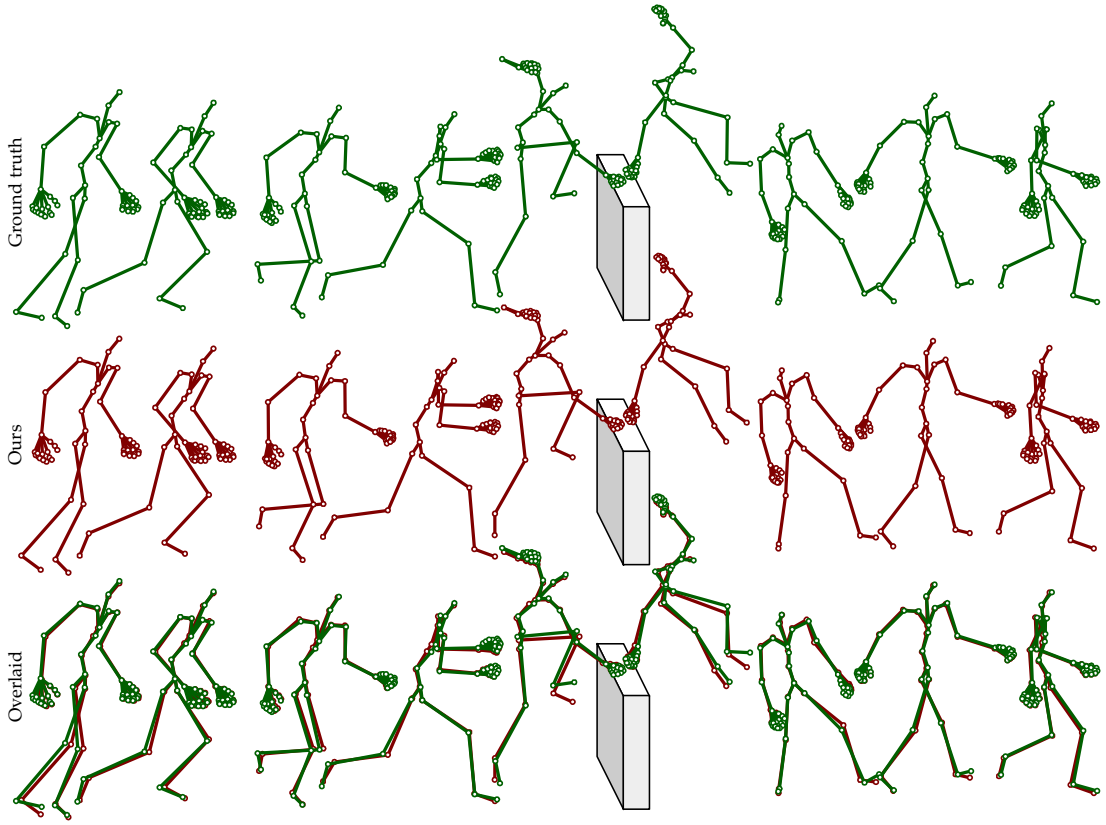


Figure 4.7 – Illustration of the representation accuracy of our deep motion representation on a challenging motion sequence, consisting in a speed vault to cross some fence or wall, depicted in white. The ground truth test motion sequence (top row, green) given as input to our model which encodes and then decodes it to get the corresponding reconstructed motion sequence (middle row, red). The bottom row shows both ground truth and reconstructed motion sequences overlaid to emphasise reconstruction errors.

Figure 4.7 illustrates the representation accuracy of our model on a challenging example motion sequence. In this example, we can see that the dynamics is relatively well encoded with fine details, such as hand orientations which are kept consistent throughout the sequence. Positional errors mostly occur during fast and ample movements such as leg swings when the character is running or jumping over the fence. Still, the MPJPE is of 1.33 cm on this example, which is slightly above global and topology averages (1.13 cm and 1 cm, respectively; see Table 4.2).

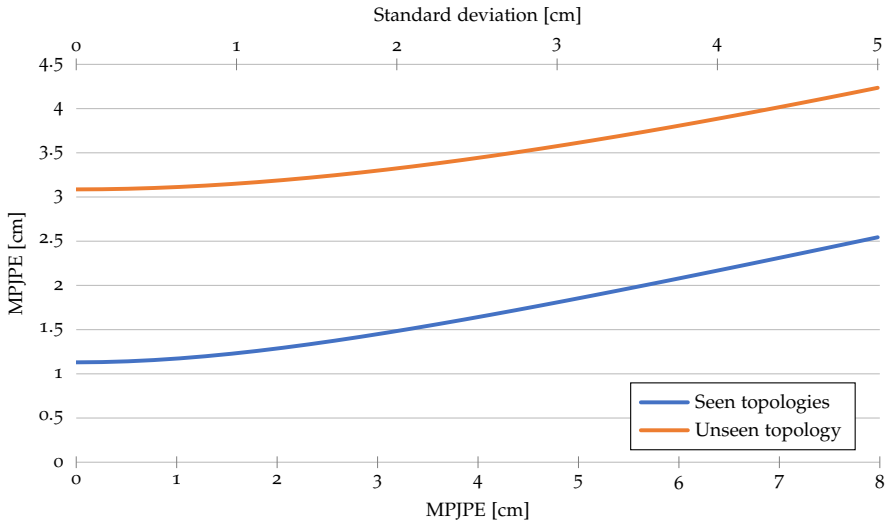


Figure 4.8 – Quantitative evaluation of our model on motion denoising. Gaussian noise is purposely added to clean motion sequences from our validation set. The resulting noised motion sequences are then encoded and decoded using our model to perform denoising. For an entire range of levels of noise, positional error on denoised sequences is plotted vertically against the positional error observed in noised sequences, horizontally. The top axis indicates the standard deviation of the different levels of noise, while the bottom axis gives the MPJPE observed in corresponding noised sequences, which is proportional to the standard deviation. Results are averaged over seen and unseen skeleton topologies. Flatter curves correspond to higher denoising ability.

4.4.3 *Motion Denoising*

In this section, we demonstrate that our model has also the ability to substantially clean noised motion sequences, which suggests that the underlying deep motion representation is well structured and prevents from encoding artefacts like noise. To this end, we pick clean motion sequences from our validation set and purposely add Gaussian noise on joint positions to obtain corresponding noised motion sequences. Then we simply encode and then decode these noised motion sequences and compare them to original clean motion sequences. We follow this procedure for an entire range of noise levels, controlled by the standard deviation of offset coordinates (sampled from a centred Gaussian) that are added to coordinates of joint positions.

In Figure 4.8, we vertically plot the MPJPE of denoised motion sequences with respect to clean motion sequences against the level of noise introduced horizontally. MPJPE is

computed for all motion sequences from our validation set and averaged over seen and unseen skeleton topologies. Without noise (at $x = 0$ in Figure 4.8), we fall back on the representation errors found when evaluating the representation accuracy of our model (see Table 4.2). Then, positional errors observed in denoised motion sequences increase with noise but slowly compared to noise increase, indicating that our model reduces noise. The right-hand side of Figure 4.8 demonstrates the ability of our model to reduce noise as the output positional error is lower than input positional error observed in noised motion sequences.

4.4.4 *Motion Retargeting*

As pointed by Aberman et al. [3], the task of motion retargeting has no formal specification as its goal is to abstract out motion dynamics. Nevertheless, research in motion retargeting relied on the so-called Mixamo dataset (see Section 2.2.2) as a source of motion sequences performed by different characters, considered as ground truth for evaluation purposes [3, 81, 100, 104, 176].

In this section, we provide quantitative retargeting results following the evaluation procedure from Aberman et al. [3], as well as qualitative visual results. Relying on Mixamo, 106 test motion sequences are considered. The evaluation is divided in two modes, called intra-structural and cross-structural. The former corresponds to motion retargeting with the same skeleton topology but different body proportions, while the latter additionally considers source characters with topology different from the target. Five characters are considered: *BigVegas*, *Goblin*, *Mousey*, *Mremireh* and *Vampire*. The first one has a different topology from others and is used as the source character to evaluate cross-structural retargeting. For intra-structural retargeting, Aberman et al. [3] considered all combinations of the four last characters as source and target characters. However, according to their publicly released implementation [3], these combinations are drawn with replacement, meaning that their evaluation of intra-structural retargeting includes self-retargeting, i.e. retargeting with same source and target characters. In the following, we further evaluate self-retargeting apart from intra-structural retargeting.

3. https://github.com/DeepMotionEditing/deep-motion-editing

Finally, we measure the retargeting error with the MPJPE normalised by the height of the skeleton.

In the results provided by Aberman et al. [3], the retargeting error is averaged over target characters. However, we observed variations in the performance of our model depending on whether the characters involved have body proportions close to humans. The reason here is that some of these humanoid characters have body proportions very different from human, such as a head about the size of the body, and that we exclusively trained our model on motion data captured from human subjects. As a result, our model is a bit less accurate on non-human morphologies. In consequence, we detail our evaluation per character to emphasise the performance and limitations of our model when body proportions are close to human or not. In particular, BigVegas, Goblin and Mremireh have body proportions close to humans, while Vampire and Mousey body proportions differ, especially for the latter.

SELF-RETARGETING. Table 4.3 shows that our model performs well on all test characters for self-retargeting, even on non-human morphologies. Still, the retargeting error is significantly higher for Mousey compared to other characters. We notably outperform the *skeleton-aware networks* proposed by Aberman et al. [3], with an average retargeting error normalised by character height at 0.009, which is equivalent to an error of 1.62 cm for a 1.8 m tall character.

Table 4.3 – Quantitative evaluation of self-retargeting, i.e. from a source to a target character with the same skeleton topology and body proportions. We measure the retargeting error with the MPJPE normalised by the height of the skeleton, and provide results per character (middle) as well as overall average (right-most). Lower values means higher retargeting accuracy.

Model	Goblin	Mousey	Mremireh	Vampire	Average
Aberman et al. [3]	0.018	**0.015**	0.017	0.021	0.018
Ours - no SJS	0.046	0.066	0.045	0.042	0.050
Ours	**0.006**	**0.015**	**0.006**	**0.007**	**0.009**

INTRA-STRUCTURAL RETARGETING. As shown in Table 4.4, the performance of our model on intra-structural retargeting is a bit lower because non-human morphologies are encountered more often. In particular, the retargeting error when Mousey is either the source or target character is increased with respect to other characters, which points out that the ability of our model to generalise to morphologies very different from the ones in the training set is a bit limited. Still, its performance is competitive with respect to the state-of-the-art skeleton-aware networks proposed by Aberman et al. [3].

Table 4.4 – Quantitative evaluation of intra-structural retargeting, i.e. from a source to a target character with the same skeleton topology. We measure the retargeting error with the MPJPE normalised by the height of the skeleton, and provide results per pair of source and target characters (middle) as well as overall average (right-most). G_o, M_o, M_r and V_a stand for Goblin, Mousey, Mremireh and Vampire characters, respectively. Lower values means higher retargeting accuracy.

Model	$G_o \leftrightarrow M_o$	$G_o \leftrightarrow M_r$	$G_o \leftrightarrow V_a$	$M_o \leftrightarrow M_r$	$M_o \leftrightarrow V_a$	$M_r \leftrightarrow V_a$	Average
Aberman et al. [3]	**0.027**	**0.033**	**0.028**	**0.036**	**0.036**	**0.024**	**0.030**
Ours - no SJS	0.118	0.071	0.073	0.112	0.131	0.066	0.095
Ours	0.067	0.055	0.055	0.065	0.067	0.040	0.058

CROSS-STRUCTURAL RETARGETING. Table 4.5 confirms our conclusions on the performance of our model on human morphologies and its relative limitation on very different morphologies. Nevertheless, our approach outperforms skeleton-aware networks [3], while it does not require a specific encoder/decoder pair to be trained for each new skeleton topology. Surprisingly, our model is more accurate on cross-structural retargeting than on intra-structural retargeting, which is presumably easier. The performance on cross-structural retargeting suggests that our model effectively

Table 4.5 – Quantitative evaluation of cross-structural retargeting, i.e. from a source (BigVegas here) to a target character with different skeleton topologies. We measure the retargeting error with the MPJPE normalised by the height of the skeleton, and provide results per target character (middle) as well as overall average (right-most). Lower values means higher retargeting accuracy.

Model	Goblin	Mousey	Mremireh	Vampire	Average
Aberman et al. [3]	0.060	**0.047**	0.062	0.071	0.060
Ours - no SJS	0.072	0.128	0.076	0.076	0.088
Ours	**0.045**	0.060	**0.035**	**0.036**	**0.044**

captures abstract motion features which are shared across different skeleton topologies. The lower performance on intra-structural retargeting indicates that our model might wrongly transfer some morphological features. We further discuss the reasons to that in Section 4.5, and the possible directions to solve it. In Figures 4.9 and 4.10, we provide visual results of motion retargeting performed using our approach and skeleton-aware networks [3].

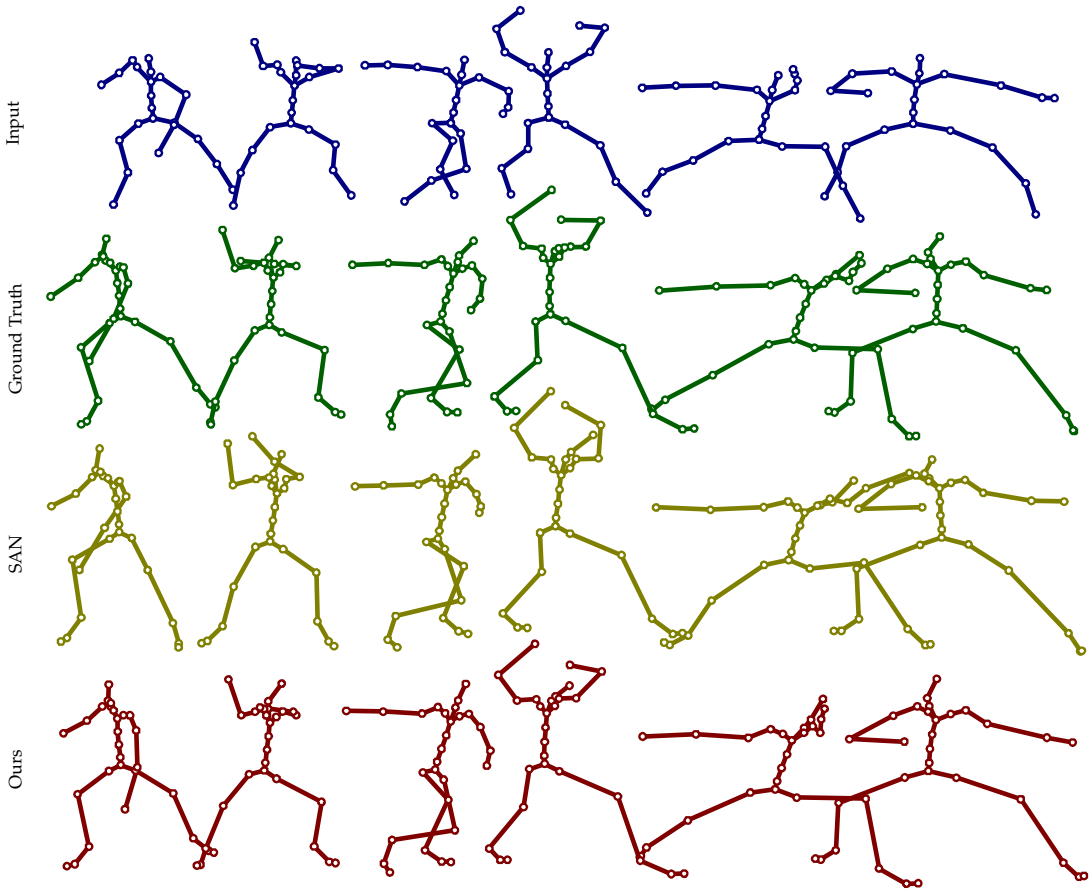


Figure 4.9 – Visual result of cross-structural retargeting. In this example, a motion sequence (1st row) drawn from Mixamo, consisting in fighting moves, is retargeted from *BigVegas* to *Vampire* using either Skeleton-Aware Networks (SAN) [3] (3rd row) or our model (4th row). The corresponding sequence for character *Vampire* (2nd row) is considered as ground truth when evaluating motion retargeting.

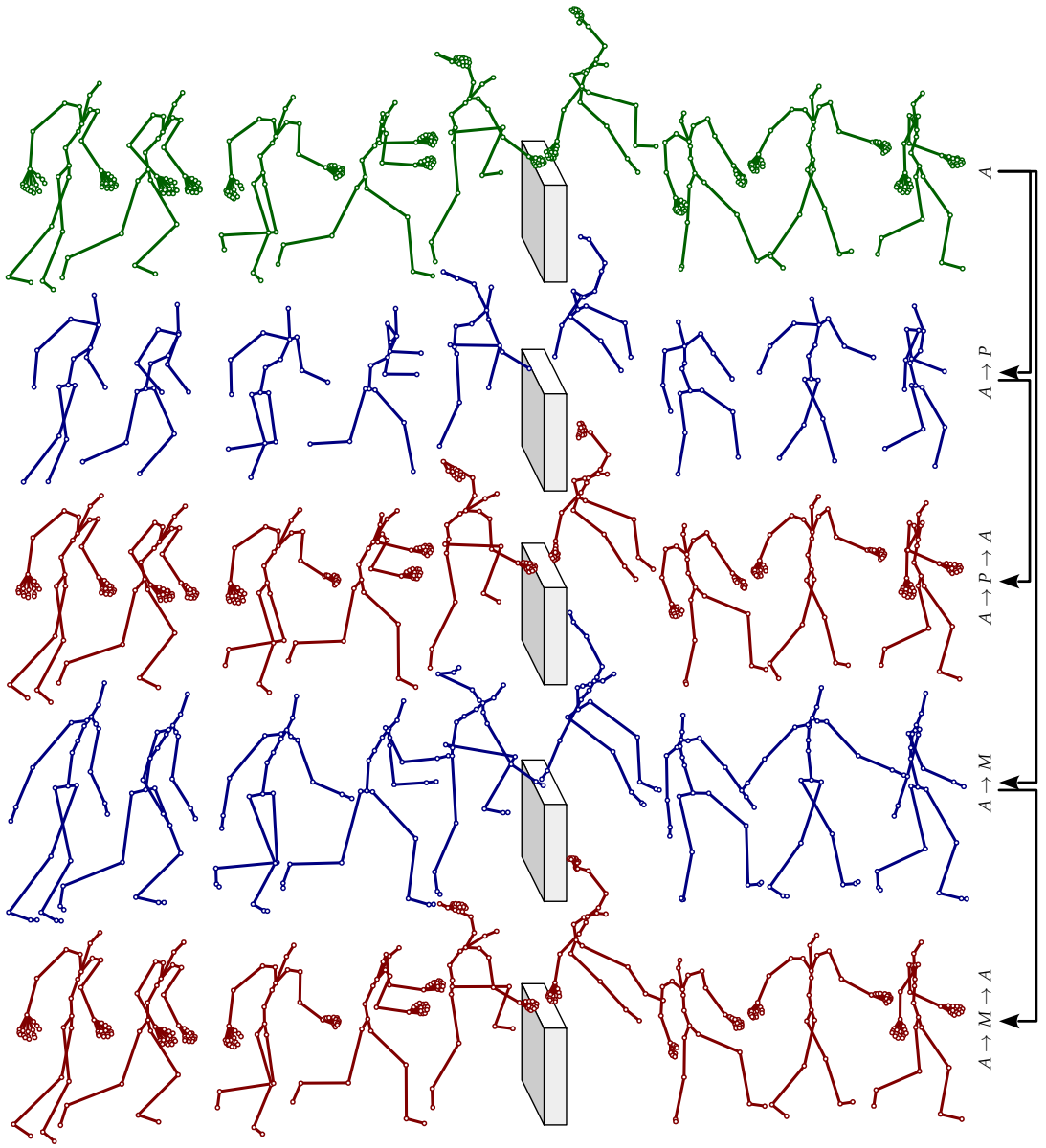


Figure 4.10 – Visual examples of motion retargeting. A motion sequence (see 1st row, in green) with AMASS skeleton topology A is retargeted to PSU-TMM100 and MPI-INF-3DHP topologies, noted P and M, respectively (see 2nd and 4th rows, respectively, in blue). Then, both are retargeted back to AMASS topology (see 3rd and 5th rows, respectively, in red). Note that our model performs well on MPI-INF-3DHP topology, even though it has never seen it during training.

STOCHASTIC JOINT SUBSAMPLING. Finally, Tables 4.3 to 4.5 show that our stochastic joint subsampling mechanism is effective, with performance of the variant Ours - no SJS consistently outperformed by our main variant. It is worth noting that the difference is larger on self-retargeting rather than intra-structural and cross-structural retargeting, and on characters with morphology close to human.

4.5 CONCLUSION

We have proposed a novel framework in which an autoencoder network based on transformers learns to encode diverse motion sequences with variable skeleton topologies and morphologies in a shared latent space, i.e. our unified deep motion representation. The reasons why our approach is successful are threefold: first, we rely on a skeleton template conditioning mechanism to tell both encoder and decoder which morphology and topology is given and targeted, respectively. Moreover, it acts as a spatial variant of positional encoding since we require skeleton templates to be normalised and consistently aligned neutral poses, and we concatenate intermediate features of motions and templates per-joint. Second, we apply during training a well-thought variant of dropout, consisting in randomly subsampling joints of input and output motion sequences and skeleton templates consistently. As our model is forced not to rely too much on specific joints, this mechanism encourages our deep motion representation to be shared across skeleton topologies, instead of representing each topology in a distinct region. Third, our autoencoder network follows a thoroughly designed transformer-based architecture, coupled with a large amount of motion data gathered from multiple existing databases with different skeleton topologies and morphologies.

We have demonstrated the ability of our model to accurately encode motion sequences, perform motion retargeting, and denoise motion sequences. Indeed, both motion retargeting and denoising can be performed reasonably well through encoding into our deep motion representation followed by decoding. This suggests that our deep motion representation is rather well structured to encode clean motion sequences with abstract features that are common to multiple topologies and morphologies. In

the case of retargeting in particular, the results obtained with our model are very close to the state-of-the-art skeleton-aware networks proposed by Aberman et al. [3]. However, both models reach an average normalised retargeting error ranging from 0.03 to 0.06 on cross-structural and intra-structural, corresponding to an abolute mean per joint position error of 5.4 cm to 10.8 cm for a 1.8 m tall character which is relatively consequent. When looking at visual results such as those illustrated in Figure 4.9, we might think that both skeleton-aware networks and our method have essentially reached the limitation of using Mixamo's source and target motion sequence pairs as ground truth for retargeting evaluation. If our thoughts are correct, further reducing the retargeting error evaluated over Mixamo would mostly be overfitting. Nevertheless, our deep motion representation unified across morphology and topology has the advantage over skeleton-aware networks [3] to not require additional encoder/decoder pairs to be trained for each new skeleton topology. This is particularly beneficial when few motion data are available for a given skeleton topology, or when training an additional encoder/decoder pair is not possible.

Beyond motion retargeting, our model might be useful for numerous applications, in a similar way that recent powerful natural language models are effective on multiple different natural language tasks. For instance, joint positions might be completed in a similar way to JUMPS, i.e. our approach presented in Chapter 3. In the temporal domain, this might include frames completion, motion prediction and in-betweening. Moreover, a well structured deep motion representation might also facilitate numerous editing applications, e.g. to affect the trajectory followed, the style expressed, the interactions with environment or other characters, etc. Last but not least, our deep motion representation, which is able to embed together motion sequences from different sources (e.g. databases or animation systems outputs), might enhance the performance and extend the scope of numerous approaches in skeletal character animation. On one hand, data-driven approaches might be applied on large general databases coupled with smaller specialised ones (e.g. annotated with style or contact labels) to increase supervision, which is currently difficult when skeleton topologies differ. On the other hand, existing methods and animation systems might be more easily used as

building blocks for novel methods of higher level and complexity since topological and morphological variations constitute a source of incompatibility.

It is tempting to think that our model might generalise to any morphology. However, our evaluation demonstrated that our model learnt on human characters is more effective on unseen human-like morphology rather than on exaggerated morphologies. Moreover, we showed that our model is less accurate on intra-structural retargeting, i.e. with source and target characters having the same topology. The reason here might be that the skeleton templates lack complexity, pushing our deep motion representation to not entirely be freed from some morphological features. When performing intra-structural retargeting, such wrongly encoded morphological features would be transferred, explaining the lower performance. Moreover, skeleton templates that we represented with joint positions have no mechanism to specify correspondences. For instance, a long half arm and a short full arm result in similar structure in skeleton templates even though the nature of their dynamics is different. In other words, skeleton templates do not tell anything about dynamics which leaves space for ambiguity when dealing with different morphologies and topologies. Increasing expressiveness of skeleton templates is a possible direction to improve our deep motion representation.

Finally, our approach, as other existing methods for motion retargeting, do not take care of foot contacts. This is an issue when transferring motion dynamics from one character toward another because annoying footskate artefacts (i.e. inconsistencies between feet motion and ground) are likely to be introduced. Even though approaches based on angular pose representations are more prone to produce such artefacts, it is still an issue with positional representations, such as in our case, since neither guarantees are given nor explicit constraints applied to ensure that legs movements are consistent with the global trajectory. In the next chapter, we present an approach intended to automatically solve such artefacts typically applied as a post-process.

5

UNDERPRESSURE: DEEP LEARNING FOR FOOT CONTACT DETECTION, GROUND REACTION FORCE ESTIMATION AND FOOTSKATE CLEANUP

Chapter abstract

Human motion synthesis and editing are essential to many applications like video games, virtual reality, and film post-production. However, they often introduce artefacts in motion data, which can be detrimental to the perceived realism. In particular, footskate is a frequent and disturbing artefact, which requires knowledge of foot contacts to be cleaned up. Current approaches to obtain foot contact labels rely either on unreliable threshold-based heuristics or on tedious manual annotation. In this chapter, we explore foot contact detection from motion data with deep learning. To this end, we first publicly release UNDERPRESSURE*, a novel motion capture database labelled with pressure insoles data serving as reliable knowledge of foot contact with the ground. Then, we design and train a deep neural network to estimate ground reaction forces exerted on the feet from motion data and then derive accurate foot contact labels. The evaluation of our model shows that we significantly outperform heuristic approaches usually used in character animation in terms of accuracy and robustness. We further propose to automatically remove footskate by solving foot contact constraints through an optimisation-based inverse kinematics approach that ensures ground reaction forces consistency. Beyond footskate cleanup, both the database and the method we propose could help to improve many approaches based on foot contact labels or ground reaction forces, including inverse dynamics problems like motion reconstruction and learning of deep motion models in motion synthesis or character animation. Our implementation, pre-trained model as well as links to database can be found at https://github.com/InterDigitalInc/UnderPressure.*

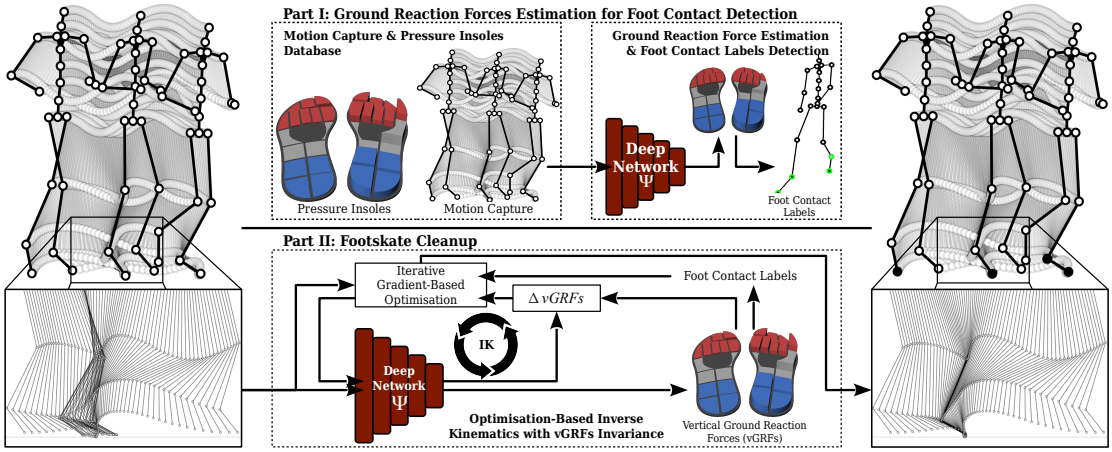


Figure 5.1 – We leverage UNDERPRESSURE, a novel publicly available dataset of motion capture synchronised with pressure insoles data, to learn a deep model for vertical ground reaction forces (vGRFs) estimation from motion data and foot contact detection. We further clean up footskate artefacts through an optimisation-based inverse kinematics algorithm while enforcing vGRF invariance.

5.1 INTRODUCTION

Perceived realism is central in animation; however, when synthesising or processing motion sequences, annoying artefacts can be introduced, such as *footskate* well-known in skeletal character animation. As seen in Section 2.4.1, It is a family of artefacts which can be defined as motion inconsistencies of the feet with respect to the ground such as when the feet are sliding on, passing through or floating above the ground while they should be static and on the ground. The problem of footskate is twofold: first, it easily appears when processing or synthesising motion sequences. Second, it is known to disturb human perception even at very low intensities [138] and to be highly detrimental to the realism of synthetic characters. In this chapter we explore foot contact detection and footskate artefacts removal, the former being a necessary step of the latter.

As of today, foot contacts are derived from motion sequences using simple heuristics, most commonly hand-crafted thresholds applied over the velocity and height of the feet to obtain binary contact labels. These approaches suffer from three main limitations.

First, the terrain height must be known everywhere, meaning that these approaches are generally not applicable to uneven or inclined terrain. Second, those thresholds have no universal optimal values and therefore must be manually tuned, ideally for every motion type, morphology and contact location (e.g. heel or toe). Finally, even with optimal values thresholding approaches are far from being 100% accurate and particularly lack of robustness, which implies that tedious manual checking is necessary when using these approaches. Especially in the context of footskate cleaning, foot contact detection must be robust to the presence of, at least, footskate artifacts.

In this chapter, we present a novel data-driven method for foot contact detection from motion sequences which significantly outperforms traditional heuristic approaches accuracy and robustness. First, we captured UNDERPRESSURE, a novel database publicly released and composed of motion sequences synchronised with pressure insoles data, from which vertical ground reaction forces (vGRFs) and accurate foot contact labels can be derived. Then, our key idea is to model vGRFs with a deep neural network, providing a finer representation of interactions of the feet with the ground than binary contact labels. Furthermore, vGRFs are directly related to the dynamics of motion unlike binary labels. To this end, we train our model to estimate the distribution of vGRFs exerted under the feet, from which foot contact labels can be calculated, encouraging robustness and generalisability since the network needs a relatively deep understanding of motion dynamics to accurately estimate vGRFs.

We evaluate our model against an OT baseline that relies on thresholding the velocity and height above ground of foot joints. In addition, we make an ablative study which considers learned linear and non-linear generalisations of the OT baseline as well as variants of the proposed model. We further experiment how these models behave in different challenging conditions representative of real-world applications.

Finally, we demonstrate the generalisability of our approach on motion sequences from other databases, as well as its integration in a fully automatic footskate cleanup workflow. The main novelty of this workflow is to leverage our deep neural network by enforcing invariance between vGRFs estimated from input and optimised motion sequences to better guide the IK optimisation used to clean footskate and maintain the consistency of interactions between feet and ground.

5.2 RELATED WORK

Early works in human animation already considered kinematic constraints such as foot contacts [20, 64, 86, 91, 92]. Foot contact labels are helpful in numerous applications: they are often necessary to clean up foot artefacts such as footskate [86], e.g. by enforcing foot constraints via IK [3, 52, 58, 121, 161, 163]; likewise, they are required to quantify such foot artefacts, e.g. for evaluation purposes [106, 161, 163, 207]; they are also helpful to mitigate the well-known problem of mean collapse / drift away in human motion prediction, particularly with deterministic and recurrent models [52, 55, 116, 185], to disambiguate human motion modelling [58, 59, 135, 136, 190] (see Section 2.3.1), and more generally to leverage contact-based loss functions for increased quality and robustness [58, 93, 132, 150, 178]; foot contact information is also deeply involved in character control based on physical simulation [1, 90, 189, 191, 201] where ground reaction forces are explicitly modelled in the physics engine, and also relates to foot-placement strategies that are a real challenge for locomotion policies [137]. In the following, we overview existing approaches for foot contacts labels detection (Section 5.2.1) and ground reaction forces estimation (Section 5.2.2), as well as existing databases of motion data labelled with information on foot contacts (Section 5.2.3).

5.2.1 *Foot Contact Labels Detection*

In Chapter 2, we have extensively reviewed existing approaches for foot contacts labels detection (see Section 2.4.1). In summary, both animation research and industry mostly rely directly or indirectly on simple heuristics with hand-crafted thresholds, applied to velocity and height of the feet as proposed by Bindiganavale and Badler [20] or Lee et al. [92]. However, they lack of temporal precision and are not reliable [91], in particular in the presence of noise or artefacts. As a result, they are not suited in the context of footskate cleanup without tedious manual checking and corrections. Beyond thresholds, researchers have explored learned contact detection models, e.g. KNN classifier [64] and neural networks [142, 150, 152, 156, 211]. However, these approaches

are learned from ground truth foot contacts that are either obtained with heuristics or manually annotated which limits the amount of data and hence the generalisability of the derived model.

5.2.2 *Ground Reaction Forces Estimation*

In physics, the force exerted by the ground on a body in contact, such as the human body, is called a ground reaction force (GRF). It is generally difficult to measure but nonetheless important in many fields of study including biomechanics, biomedical engineering and physics-based animation. Researchers in biomechanics and biomedical investigated GRF estimation from plantar pressure sensors [72, 123, 145], inertial and optical motion capture systems [39, 42, 75, 126], 3D accelerometers [94], and Kinect [38]. However, GRF estimation for biomechanics or biomedical applications is beyond the scope of our approach that uses vGRF distribution as a proxy representation and is intended for human animation applications. For more details on GRF estimation in biomechanics and biomedical engineering, we refer the reader to the systematic review by Ancillao et al. [12].

Early works in motion reconstruction leveraged pressure sensors to measure GRFs, because of their importance in dynamics. Ha et al. [50] formulated the problem as a per-frame optimisation of end-effector positions obtained from a hand tracking device, and linear and angular momentums measured with pressure platforms. Later, Zhang et al. [208] leveraged a pair of pressure sensing shoes as well as three depth cameras to develop a full-body motion reconstruction framework consisting in kinematic pose reconstruction followed by physics-based motion optimisation.

More recently, several approaches instead estimated GRFs from monocular images, starting with 2D and 3D pose estimation and then solving physical optimisation problems. Zell et al. [205] proposed to estimate inner and exterior forces by optimising camera parameters and 2D pose reconstruction with a linear combination of base poses in a first step, and then GRFs and inner joint torques to satisfy the equations of motion and resolve camera projection ambiguities. Li et al. [103] estimated 3D motion and forces between a subject and its environment by minimizing the discrepancy

between the observed and reprojected 2D poses, with priors on estimated 3D poses, trajectory smoothness and physical plausibility for regularisation. Rempe et al. [142] and Shimada et al. [152] proposed a similar pipeline but focusing on more dynamic and diverse human motions, without object interactions. Shimada et al. further corrected imbalanced stationary poses. Later on, they extended their approach with additional DNN [151]: *TPNet* first regresses target 3D poses and contact states from 2D key points. Then, *GRFNet* and *DyNet* iteratively estimate GRFs and proportional-derivative (PD) controller gain parameters in a dynamic cycle where the character pose is updated at each step after FK.

Different from motion reconstruction from images, Zell et al. [204] proposed a weakly-supervised approach to inverse dynamics. An MLP is trained to estimate GRFs, moments and joint torques from motion such that the input motion is reconstructed using forward dynamics in an optimisation loop. Motion capture data synchronised with force plates enable supervision during training: reconstructed ground reaction force and moment and joint torque divergences are penalised while GRF are minimised whenever feet are not in contact with the ground.

To the best of our knowledge, the closest work to the proposed method is the deep learning approach to improve computing stability in human pose estimation proposed by Scott et al. [148]. In this work, body dynamics analysis from joint positions is investigated while most papers on human pose estimation only focus on skeleton kinematics. A CNN called *PressNet* is proposed to estimate 2D foot pressure maps from joint positions and validated on a novel dataset of tai chi sequences (see Section 5.2.3). CoP and base of support (BoS) are computed from pressure values and used for validation. Unlike Scott et al., our work specifically targets foot contact detection and footskate cleanup, tasks that are relevant to human character animation. We explore more diverse motion sequences including different types of locomotion at different paces performed in different ways (forward, backward, sideways) as well as sequences in non-flat environments such as stair climbing and stepping on solid obstacles.

5.2.3 *Foot Contact & Ground Reaction Force Databases*

As of today, motion capture data annotated with accurate foot contact information are scarce. Researchers in biomechanics and biomedical engineering have released a few databases of motion capture with GRFs e.g. Kulbacki et al. [87], however most of these databases are not suitable for animation purposes as they typically focus on specific aspects of movement, or target stage and symptoms recognition of diseases in pathological subjects.

To the best of our knowledge, the closest database to the one we propose is PSU-TMM100. Recently released to the computer vision community by Scott et al. [148], this dataset provides videos from two views, motion capture markers, body joints and foot pressure recorded with insole sensors. It contains about 7.6 hours of data during which 10 subjects are performing 24-form simplified tai chi. Although similar in terms of scale and nature of the captured data, the database we propose has quite different types of motion. While PSU-TMM100 contains specific tai chi sequences mostly composed of slow body movements with long and stable foot supports, UNDERPRESSURE provides diversified sequences focused on but not restricted to locomotion at different paces including on non-flat environments (see Table 5.1), i.e. more challenging conditions for foot contacts detection.

5.3 DATABASE

In this work, we release UNDERPRESSURE, a motion capture database annotated with pressure insoles data, designed primarily for character animation purposes. In the following, we provide information about the capture, the motion characteristics, and the preprocessing steps.

We recorded 10 healthy adult volunteers (2F, 8M) with diverse morphologies aged between 21 and 55 years (32 ± 11 yr), weighing between 65 and 91 kilograms (79 ± 9 kg), and measuring between 167 and 187 centimetres (177 ± 5 cm). Each subject performed the same set of activities, including forward and backward locomotion at different paces, sitting, standing, passing obstacles, climbing stairs, as well as motions on uneven

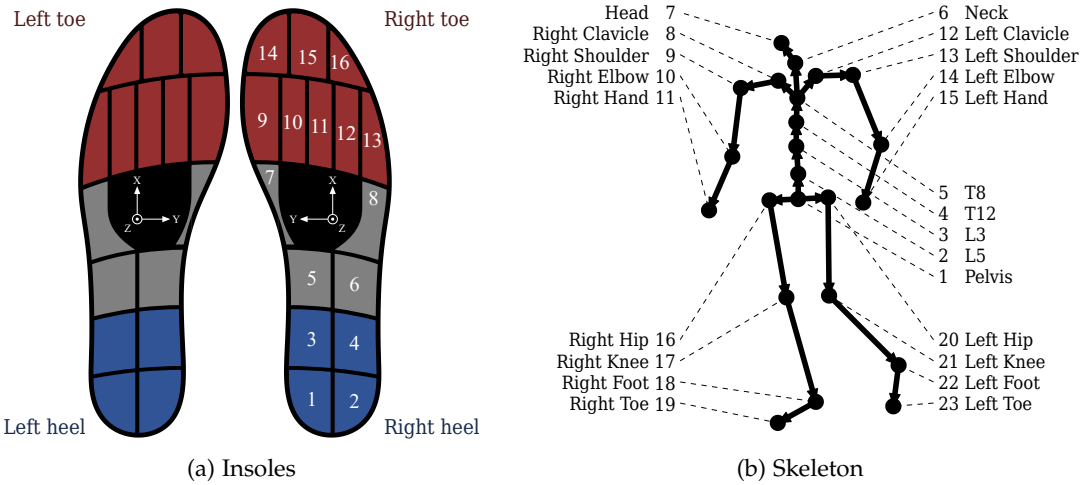
Table 5.1 – Motion sequence categories in UNDERPRESSURE.

Category	Motion Type	Duration [mn]
Locomotion, forward	slow walking	43.9
	normal walking	42.0
	fast walking	42.9
	running	30.1
Locomotion, backward	slow walking	21.1
	normal walking	21.8
	fast walking	20.8
	running	15.5
Locomotion, miscellaneous	running sideways	12.3
	hopping	13.9
	stairs 1 at a time	18.0
	stairs 2 at a time	13.9
Locomotion with obstacles	stepping on obstacles	5.7
	stepping over obstacles	11.4
	jumping over obstacles	10.5
Idle	leg stand-up	4.8
	sit-down	4.8
	crouched down	4.8
Total		338.2

terrain like going up and down stairs. The detailed composition of our dataset is provided in Table 5.1. Motion capture data for each subject last approximately 34 minutes, for a total of 5.6 hours of motion capture.

5.3.1 *Motion Capture*

Subjects were equipped with the *Xsens MVN Link* motion capture system [147]. The hardware consists of 17 inertial measurement units (IMUs) running at 240 Hz embedded in the *MVN Link* suit. Each IMU contains a 3D accelerometer, a gyroscope, and a magnetometer. Capturing was performed using the *Xsens MVN Animate* software, an engine customised for 3D character animation that combines tracking data of the 17 individual IMUs with a 23-segment biomechanical model (see Figure 5.2b) to obtain segment positions and orientations. We calibrated *MVN Animate* for each subject with height, arm span and shoe length measurements as inputs while other body dimensions and proportions were estimated through the calibration, as well as the orientation of



(a) Insoles (b) Skeleton

Figure 5.2 – Left) Pressure cell layout of *Moticon's OpenGo Sensor Insoles* [6]. Blue (1 to 4) and red (9 to 16) cells are the groups of cells used to compute heel and toe contacts, respectively. Axes at insole centres represent inertial measurement units. Right) *Xsens MVN*'s [147] motion capture skeleton with 23 joints.

motion trackers with respect to the corresponding segments. After *MVN Animate* processing, motion data consist of pose sequences with 23 segments sampled at 240 Hz.

5.3.2 *Foot Pressure*

In addition to motion capture, we recorded the spatial distribution of plantar foot pressures. To this end, subjects were also equipped with *Moticon's OpenGo Sensor Insoles* [6] placed into their shoes. Each insole has 16 plantar pressure sensors with a resolution of 0.25 N/cm^2 and a 6-axis IMU, both running at 100 Hz (see Figure 5.2a). Moreover, we weighed each subject with full equipment to enable vGRF normalisation and equipped subjects with the same shoes whose soles are thin and flexible insole for accurate and faithful pressure measures as well as for controlling the grip.

5.3.3 *Post-Capture Processing*

GROUND REACTION FORCES Captured data include motion sequences, plantar pressure distribution and foot acceleration. Since pressure is defined as the perpen-

dicular force per unit area, we additionally compute vGRF components by multiplying pressure values by the corresponding cell areas. The motivation here is that groups of vGRFs are easier to aggregate (by summation). We also normalise these values to express them as subject weight proportions.

FOOT CONTACT LABELS We derive ground truth foot contact labels deterministically from vGRFs. In this work, we consider two contact locations per foot, i.e. heels and toes as commonly done in human animation [58, 64, 151]. To compute contact labels, vGRF components are first smoothed with a Gaussian filter to avoid rapidly alternating labels due to threshold effects. Then, smoothed vGRF components are summed per contact location (see red and blue cells in Figure 5.2a), rescaled such that the sum of blue and red cell vGRFs is equal to total vGRF (to properly ignore grey cells which particularly suffer from noised measures and can be activated during either toes or heels contact) and then a threshold at 5% of the body weight is applied to obtain raw labels. Finally, raw labels are discarded whenever per-foot total vGRF (including grey cells) is below 10% of the body weight to avoid false positives triggered by noise. Contact phases shorter than 0.1 s are also discarded for the same reason. In the following, we refer to this binary contact labels calculation as the contact function Γ.

SYNCHRONISATION Since we jointly capture motion and foot pressure data with separate devices, our records must be accurately synchronised in absence of a genlock signal. To this end, subjects were asked to perform a simple control movement at the beginning and end of each capture sequence, consisting in an in-place double-leg jump. This allows us to match vertical acceleration peaks measured on pressure insole IMUs with peaks computed from motion captured foot positions. Although numerical differentiation is known to amplify high frequency noise, we found that the framerate of our motion capture data was sufficiently high and measurement noise was small enough for synchronisation.

DOWNSAMPLING AND TRIMMING After synchronising our data, we downsampled motion capture data from 240 Hz to 100 Hz using spherical linear interpolation to

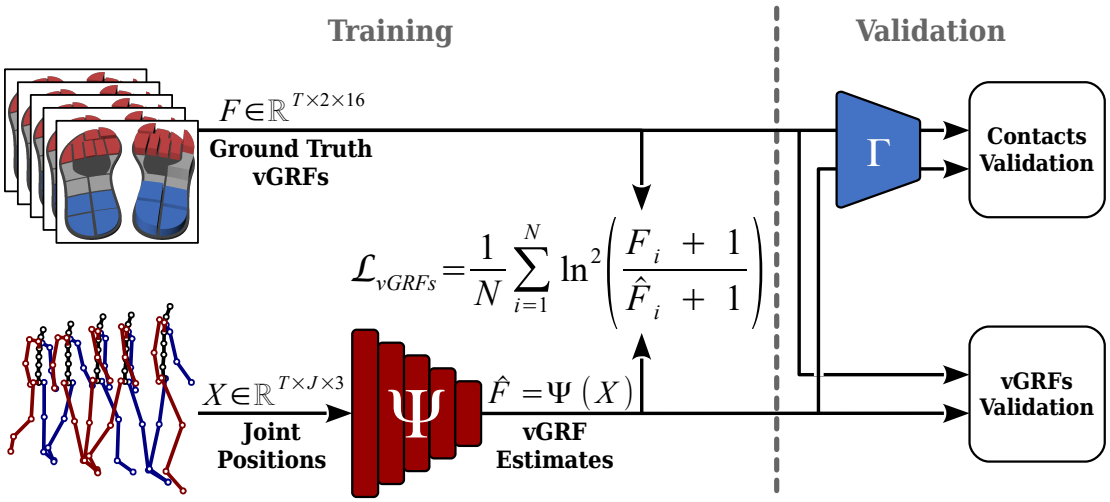


Figure 5.3 – Overview of our approach. Our database provides synchronised input motion sequences and target vertical ground reaction forces (vGRFs) to train our network Ψ depicted in red. The blue trapezoid represents the contact function Γ (see Section 5.3.3). At runtime, our network estimates vGRFs from which foot contact labels can be derived using the contact function Γ, both useful in many applications, e.g., reconstructing motion from images, cleaning footskate, finding suitable transition frames for motion blending, adapting animations to uneven terrain, and many more. As illustrated, both estimated vGRFs and detected contacts are evaluated in Sections 5.5.2 and 5.5.3, respectively.

match the framerate of pressure insole data, and trimmed the beginning and end of each sequence to remove the synchronisation patterns. Original motion sequences at 240 Hz are also provided in our database.

5.4 DEEP NEURAL NETWORK

In this section, we describe the proposed method to learn a deep model for vGRFs estimation from motion capture data. Learning vGRFs instead of binary contact labels encourages our deep neural network to more accurately model interactions between feet and ground, and enforces motion dynamics understanding. See Figure 5.3 for an overview of our approach.

5.4.1 *Data Representation*

INPUT At each frame t, the human pose $X_t \in \mathbb{R}^{J\times 3}$ is represented by the position of its $J = 23$ joints in a global Euclidean space. We design our deep network Ψ to output vGRFs and contact labels at each frame from a few surrounding input frames with padding when needed. The full input pose sequence is then $X \in \mathbb{R}^{T\times J\times 3}$, where T is a variable number of frames.

OUTPUT As previously described, our deep network Ψ estimates the vGRF distribution from motion data. At each frame, it outputs $\hat{F} = \Psi(X)$ with $\hat{F} \in \mathbb{R}^{T\times 2\times 16}$, i.e. 16 positive real-valued vGRF components, corresponding to the 16 insole pressure cells for each foot, expressed proportionally to subject weight.

5.4.2 *Network Architecture*

We designed our network Ψ to process variable-length sequences. To this end, the network is composed of four 1D temporal convolutional layers with 7-frame wide kernels, followed by three fully-connected layers applied at each frame independently to preserve the support variable-length sequence. Each convolutional or fully-connected layer is followed by ELUs as activation, except for the last one which is a softplus activation to output nonnegative vGRF components.

5.4.3 *Training and Inference*

During training, our network is iteratively exposed to sequences of human poses and tries to estimate corresponding vGRF components as depicted in Figure 5.3. To encourage robustness and smooth convergence, we make use of stochastic data augmentation. First, similar to random crops and rotations used on images in computer vision, we apply random vGRF-invariant transformations on input pose sequences including translations, horizontal rotations, scaling, and left-right mirroring.

For each sequence, we also randomly draw its skeleton which is then animated by joint angles to further robustify our network. To do so, we precomputed (offline) an singular value decomposition (SVD) basis of the skeletons captured in our database. At training time, we draw new skeletons by linearly combining precomputed singular vectors with randomly sampled weights. We then further edit these skeletons by randomly moving joint relative positions and rescaling bone lengths to obtain morphological variations. The resulting input motion sequences purposely suffer from artefacts since kinematic chains (i.e. from root joint to feet) have been randomly edited, which encourages the network to be resilient with respect to perturbed inputs. Joint positions are finally computed through forward kinematics and fed to the network.

To train our deep network Ψ, we minimise a reconstruction loss of vGRF components. Instead of the standard MSE, we use the mean squared logarithmic error (MSLE). It has the property to only focus on the relative difference between target and estimated values (see right-hand side of Equation (5.1)), which is convenient when the values considered can be several orders of magnitude apart. In our case, actual vGRFs can be strictly positive and arbitrarily low (e.g. during transition from the double leg stance to the single leg stance) as well as very high (e.g. during jump landing). The loss function used to train our network is then

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left(ln(F_i+1) - ln(\hat{F}_i+1)\right)^2 = \frac{1}{N}\sum_{i=1}^{N} ln^2\left(\frac{F_i+1}{\hat{F}_i+1}\right) \tag{5.1}$$

where F_i are the ground truth vGRF components, and $\hat{F}_i = \Psi(X)_i$ their estimated counterpart. Adding 1 to both F and $\hat{F}$ ensures that the loss is defined when vGRF component value goes to zero.

At inference, our deep network estimates vGRF components from joint positions as inputs. Then, foot contact labels can be calculated from vGRF estimates using the contact function Γ (see Section 5.3.3).

5.5 EVALUATION

In this section we present results of our method to assess estimated vGRFs and detected foot contacts. After providing implementation details necessary for reproducibility, we assess foot contact labels detection and vGRF estimation. Finally, we also evaluate foot contact detection performance on different perturbed motion sequences, simulating challenging conditions encountered in concrete applications.

5.5.1 *Implementation Details*

DATA. We split our database into training and testing sets. To ensure robust evaluation, the testing set is composed of the sequences performed by three out of the ten subjects (1F+2M, $\{S8, S9, S10\}$), representing approximately 30% of the overall database. For training, we further divide the remaining 70% to keep a validation set (about 10%) and use early stopping during training. Moreover, we split each training motion sequence into overlapping windows of $T = 240$ frames, i.e. 2.4 s.

ARCHITECTURE. The four convolutional layers at the beginning of our network have respectively 128, 128, 256, and 256 7-frame wide filters while the back-end FC layers have 256 neurons each. Dropout with probability $p = 0.2$ is applied before each FC layer. The total number of weights in our network is 1'137'792.

TRAINING. We implemented our deep neural network in Python using PyTorch. Training and validation were executed on an NVidia Tesla V100 GPU while other results were obtained either on an NVidia GeForce RTX 2060 GPU or on CPU. We trained our deep neural network through stochastic gradient descent (see Equation (5.1)) for about 2500 epochs (about 10 days) at each of which a new version of the training set was randomly generated (see Section 5.4.3). We used Adam optimisation algorithm [82] with a mini-batch size of 64, learning rate $\alpha = 3 \cdot 10^{-5}$ and hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

5.5.2 *Foot Contacts Detection*

METRIC. To evaluate foot contact detection, we use the F_1 score which is the harmonic mean of precision (fraction of correctly detected labels among detected labels) and recall (fraction of correctly detected labels among expected contact labels).

BASELINE. To evaluate our model against commonly used heuristics-based approaches using thresholds, we define an optimal thresholds (OT) baseline. It has two thresholds that are applied on foot height and velocity norm. To demonstrate the effectiveness of our approach, we set these two thresholds to the values which maximise the F_1 score over the training set, computed by recursive grid search. Note that in practical applications, such optimal values are not available. Therefore, this baseline constitutes an upper bound of threshold-based approaches.

ABLATIVE STUDY. We also investigate the relevance of our architecture with an ablative study. We first consider two linear models taking as inputs joint positions and velocities, as learned generalisations of the OT baseline: the former (*Linear-Feet*), takes foot and ankle joints as inputs while the latter (*Linear*) takes all joints. Then, we consider a 3-layer MLP, i.e. the architecture of the foot contact detection module in the style transfer framework proposed by Smith et al. [156], introducing non-linearities with respect to Linear-Feet and Linear models. These three models have real-valued outputs, for which positive values are considered as foot contacts with the ground, and are trained with binary cross-entropy (BCE). Finally, we explore two variants of our deep network. First, in *Ours-C* variant, we adapt our architecture to directly estimate foot contact labels instead of vGRFs by change the last layer and its activation (i.e. number of neurons and sigmoid instead of softplus) and train this adapted model with BCE. Then, in *Ours-C&F* variant, we combine *Ours-C* variant with the main proposed model *Ours*. Our architecture is adapted to output both contacts and vGRF components (convolutional layers are shared while FC layers are duplicated and separately learned for each output) and trained with both MSLE and BCE over vGRF and contacts, respectively.

FOOT CONTACT DETECTION RESULTS. Table 5.2 reports the F_1 score for each model of our ablative study and each motion category as well as overall results. First, *Linear* and *Linear-Feet* show improved performances with respect to the OT baseline, which tends to confirm that thresholds based approaches lack complexity. Increased performances obtained by *Linear* model with respect to *Linear-Feet* model suggests that foot contact detection also benefits from other body joints, pointing out one of the limitation of thresholds applied on foot joints. The higher F_1 score of the 3-layer MLP with respect to linear models confirms their limitations. Finally the proposed architecture further increases the detection accuracy with relatively small differences among variants.

Table 5.2 – F_1 score on foot contact labels detection of our method, its variants for ablative study purposes, and the OT baseline. Bold and underline respectively indicate per-column best and second best. Our method outperforms the OT baseline and its linear generalisations, and the proposed architecture seems relevant with respect to the 3-layer MLP.

Model	Walking	Running	Obstacles	Hopping	Stairs	Idle	Overall
OT baseline	0.927	0.869	0.926	0.859	0.882	0.826	0.909
Linear-Feet	0.936	0.863	0.921	0.824	0.906	0.812	0.913
Linear	0.937	0.868	0.926	0.855	0.925	0.912	0.923
3-layer MLP	0.940	0.883	0.926	0.882	0.947	0.921	0.930
Ours-C	0.946	0.917	0.941	<u>0.930</u>	<u>0.956</u>	0.941	0.942
Ours-C&F	<u>0.948</u>	<u>0.918</u>	<u>0.942</u>	0.923	0.954	<u>0.946</u>	<u>0.943</u>
Ours	**0.949**	**0.930**	**0.944**	**0.931**	**0.959**	**0.948**	**0.947**

TEMPORAL ANALYSIS OF FOOT CONTACT DETECTION. Up to now, we provided temporally global foot contact detection results. However, misdetected labels located closer to contact phase changes are less severe in the sense that they would result in less severe biases or artefacts in most applications, e.g. footskate cleanup. To this end, we provide a finer analysis in the following. In Figure 5.4, we plot the F_1 score against an increasing temporal tolerance to detection errors. In Figure 5.5, the distribution of false positive rates is given according to a normalised temporal frame where ground truth off-contact phase intervals are mapped to $[0,1]$. Both figures confirm the limitations of heuristics approaches represented by the OT baseline whose false positive rate is significant in the middle of off-contact phases. On the contrary, learned models show

convergence of accuracy close to 100% with increasing tolerance as well as low false positive rates in the middle of off-contact phases.

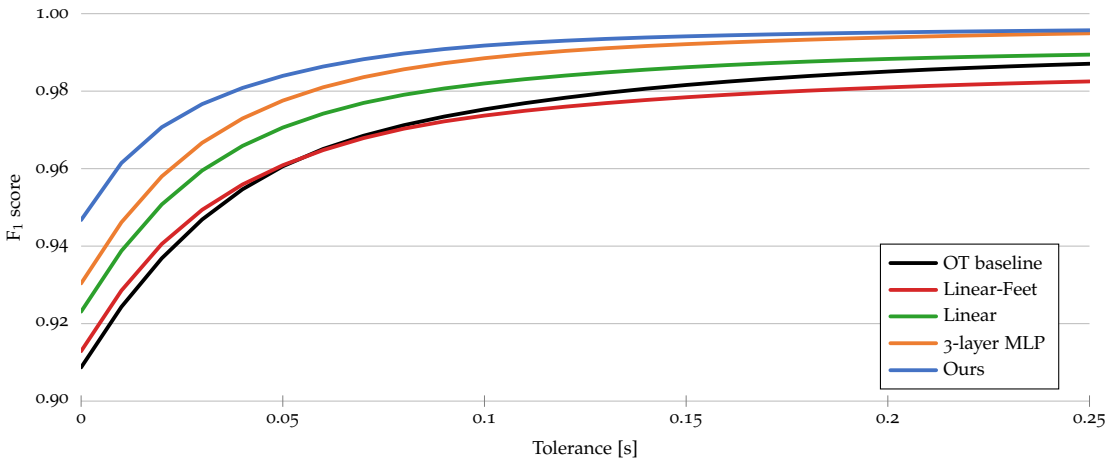


Figure 5.4 – F_1 score curves against temporal tolerance. Here we compute the F_1 score with tolerance $t \in [0, 0.25]$ by considering contact labels correct whenever they are located at most t seconds away from the closest contact phase. At $t = 0$, we fall back to the *Overall* column in Table 5.2. The optimal thresholds (OT) baseline and its linear generalisations have large errors far from contact phase changes, resulting in relatively low F_1 scores compared to our model even with high temporal tolerances.

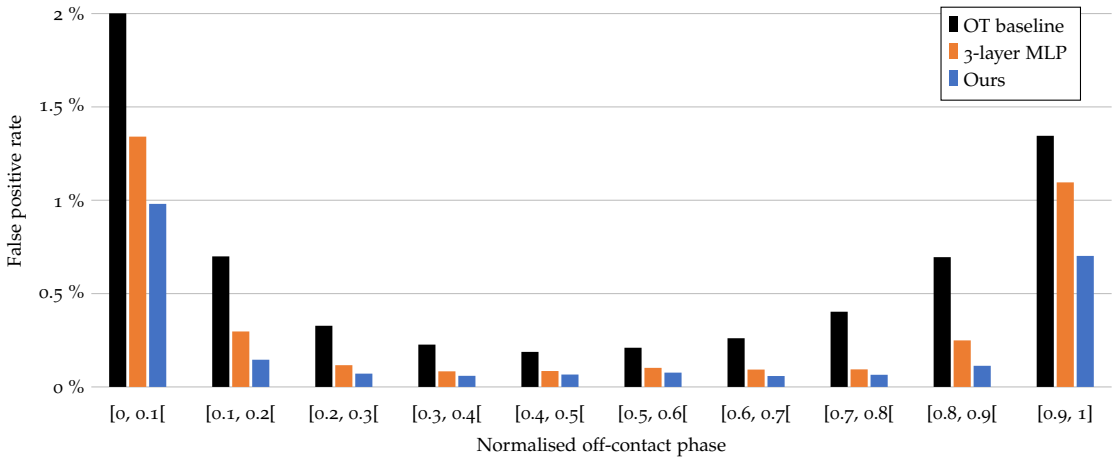


Figure 5.5 – False positive distribution during normalised off-contact phases, i.e. mapped to $[0, 1]$. Intuitively, the farther misdetected contacts are from contact phases, the more severe they are. False positive rates of learned models like ours quickly decrease outside contact phases while the optimal thresholds (OT) baseline keeps it significantly higher in-between contacts.

Table 5.3 – Root mean squared error (RMSE) of the estimated vGRF normalised by body weight. Estimating foot contacts in addition to vGRFs (*Ours-C&F*) seems slightly detrimental compared to estimating only vGRFs (*Ours*).

Model	Walking	Running	Obstacles	Hopping	Stairs	Idle	Overall
Ours-C&F	9.5%	14.8%	12.4%	14.3%	11.8%	13.1%	11.4%
Ours	**9.1%**	**14.3%**	**11.6%**	**14.1%**	**10.9%**	**11.9%**	**10.9%**

Table 5.4 – Median absolute deviation (MAD) in milimetres of CoP computed from estimated vGRF components. Similarly to estimated vGRF, modelling foot contact labels in addition to vGRFs (*Ours*) slightly affects CoP accuracy.

Model	Walking	Running	Obstacles	Hopping	Stairs	Idle	Overall
Ours-C&F	16.7	12.3	17.8	15.5	17.9	28.9	16.4
Ours	**13.3**	**11.2**	**15.1**	**12.3**	**13.9**	**25.9**	**13.4**

5.5.3 *vGRFs Estimation*

METRICS. In this section we assess performances of our method on vGRFs estimation from motion capture data. We use the root mean squared error (RMSE) of per-foot total vGRF which is mainly sensitive to global biases (at foot scale). In complement, we also evaluate the centre of pressure (CoP) computed from vGRF estimates, which is more sensitive to local errors (at pressure cell scale) since it is calculated as the weighted mean of vGRF component application points.

RESULTS. Table 5.3 gives the RMSE of the estimated vGRFs proportionally to the subjects' body weight while Table 5.4 provides the MAD (or median distance to ground truth) of the CoP. Moreover, Figure 5.6 depicts the distribution of offsets from predicted to ground truth CoP. As for contact detection, these results suggest that learning to model foot contact labels in addition to vGRF (*Ours-C&F*) reduces accuracy, although the performances of both variants are close. Note that other variants evaluated on contact detection in the previous section only detect contact labels and hence cannot be evaluated on vGRF estimation. Finally, visual results of vGRFs estimated using our main variant are depicted in Figure 5.7.

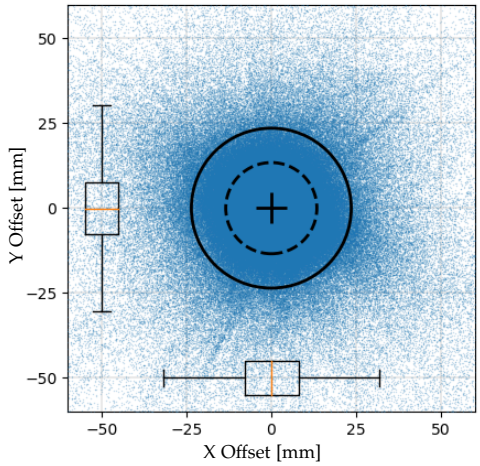


Figure 5.6 – Scatter plot of 2D offsets between centres of pressure computed from ground truth and estimated vGRFs. The concentric solid and dashed circles respectively represent the mean and median norm of 2D offsets.

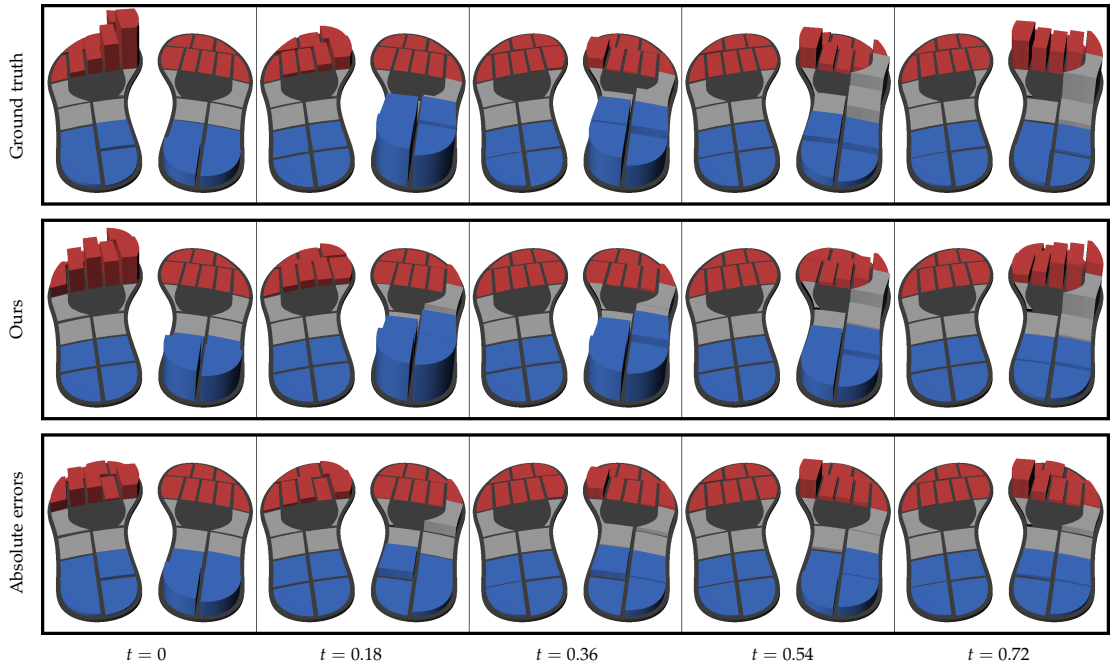


Figure 5.7 – Illustration of vGRF components at different timestamps during a walking cycle. The top, middle and bottom rows respectively depict the ground truth, vGRFs estimated from motion by our deep neural network and the corresponding absolute error.

DISCUSSION ON PRESSURE INSOLES ACCURACY. In biomechanics, force plates are generally considered gold standard to measure vGRF and CoP; however, the environment in which we captured the proposed database, i.e. spanning over a significant area including obstacles and stairs, prevented us from resorting on force plates to capture force or pressure data. As a result, our evaluation relies on pressure insole measures as ground truth. Existing works in biomechanics [74, 129] evaluated pressure insoles accuracy with respect to force plates, and tell us that vGRFs measured with pressure insoles suffer from a RMSE up to approximately 10% of the subject weight, being subject to variations depending on experimental conditions. Thus, Table 5.3 indicates that vGRF estimation errors of our deep neural network are approximately of the same order of magnitude as the measurement error this is expected with pressure insoles.

5.5.4 *Foot Contacts Detection in Challenging Conditions*

As explained in Section 5.2, applications requiring foot contact labels detection include quantifying or correcting foot artefacts. By definition, motion sequences considered in such applications are expected to be perturbed at least by footskate artefacts, and sometimes by other artefacts like noise or distortions. As a consequence, the performance of a method for foot contact detection is not only relevant on clean motion capture data (as evaluated in Section 5.5.2), but also on motion sequences suffering from perturbations of all kinds. In this section, we evaluate the performance of our model compared to our baseline and ablative variants of our model on perturbed motion sequences. To this end, we purposely introduce three types of artefacts in motion sequences from our database such that foot contact labels can still be used as ground truth to measure contact detection performance. First we add Gaussian noise to joint positions. Second we introduce distortions caused by going through a partially trained motion autoencoder. Finally, we generate footskate artefacts by blending different motion sequences from our database.

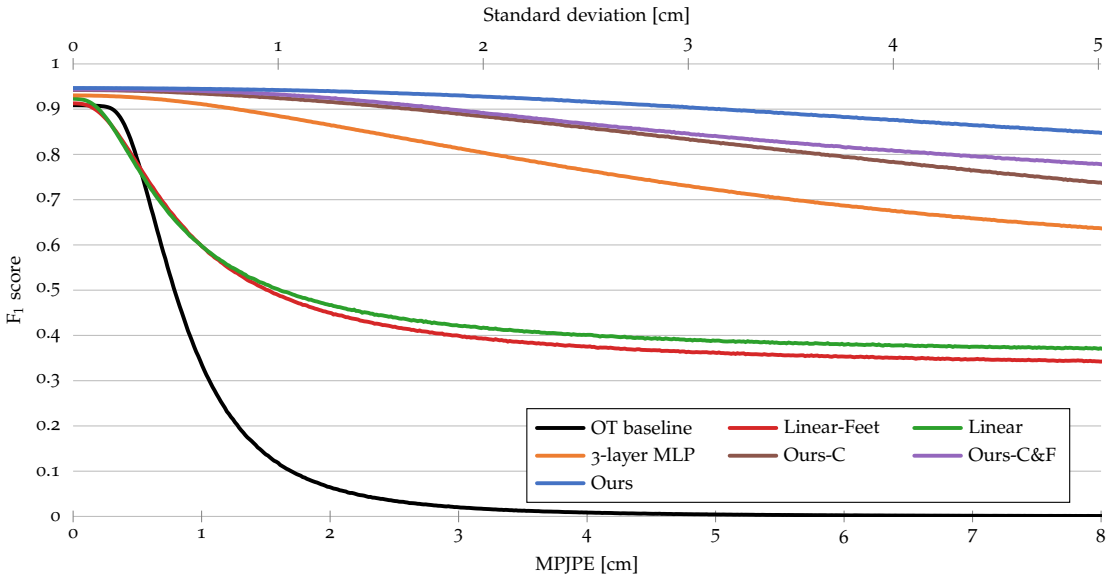


Figure 5.8 – F_1 score on foot contacts detection from motion sequences purposely noised with additive isotropic Gaussian noise. Each curve represents the F_1 score against the amount of noise introduced, measured with the MPJPE in centimetres indicated by the bottom horizontal axis, while the top horizontal axis gives the corresponding standard deviations of the Gaussian noise. The results indicate that our method is more robust to noise.

MOTIONS PERTURBED WITH GAUSSIAN NOISE. First, we simply evaluate how contact detection performances are affected by additive isotropic Gaussian noise. To this end, we first take with ground truth sequences from our test set and purposely add Gaussian noise to joint positions. Then we try to detect foot contacts from the noised motion sequences using various models evaluated in Section 5.5.2 and report their performances with the F_1 score. We repeat this procedure for an entire range of noise levels whose standard deviation varies from 0 to 5 centimetres which corresponds to a MPJPE ranging from 0 to 8 centimetres. Finally, we plot the resulting F_1 score curves in Figure 5.8.

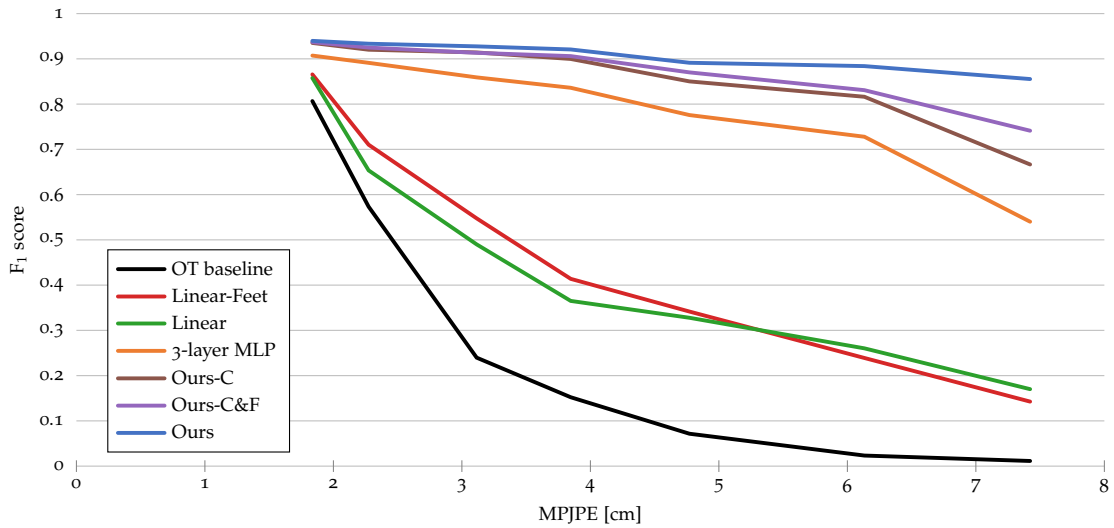


Figure 5.9 – F_1 score on foot contacts detection from motion sequences distorted by a motion autoencoder early-stopped at different epochs to emulate different amounts of distortion. Each curve displays the F_1 score against the amount of distortion introduced, measured with the MPJPE in centimetres. The results indicate that our method is more robust to distorted sequences.

MOTIONS DISTORTED BY AN AUTOENCODER. Second, to simulate perturbations more faithful to what is encountered in real-world applications, we evaluate detection performances on motion sequences that have been encoded and decoded by an autoencoder. To this purpose, we trained a convolutional autoencoder similar to the one proposed by Holden et al. [59] to reconstruct motion sequences from our test set for 100 epochs (about 10 minutes). Then we evaluate foot contact detection on motion sequences which have been encoded and decoded with our autoencoder. Moreover, to emulate different level of perturbations, we leveraged the fact that the amount of perturbations introduced when passing through the autoencoder continuously decrease during training. Hence, we ran this evaluation scheme at different epochs and plot in Figure 5.9 F_1 score curves obtained against the amount of perturbations measured at each of those epochs using the MPJPE.

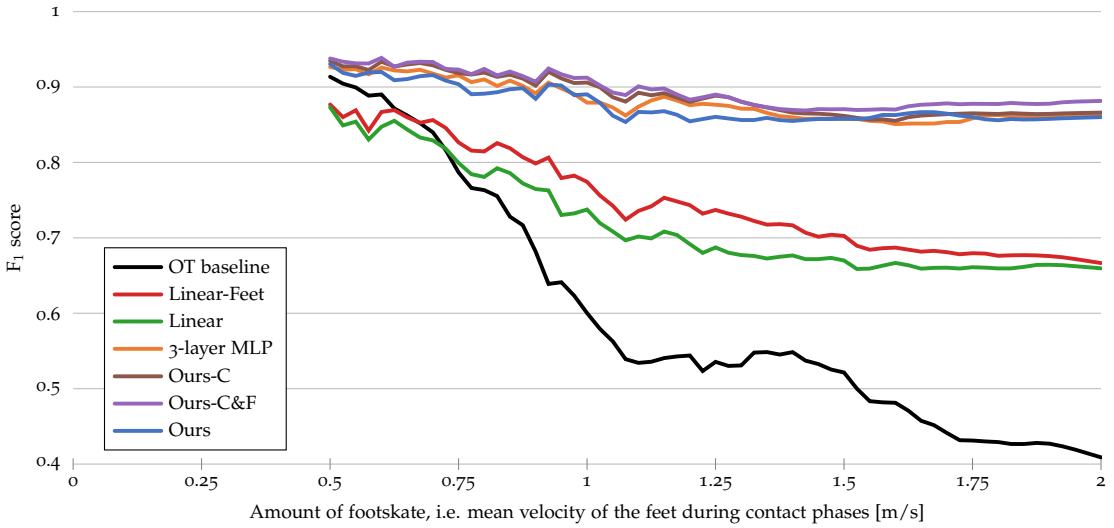


Figure 5.10 – F_1 score against amount of footskate. Test motions with matching foot contact patterns have been blended to purposely introduce footskate while preserving contact labels. Foot contact detection is then evaluated on such motions suffering from footskate, and compared to the ground truth contact labels. The amount of footskate introduced through blending is measured using the mean velocity of the feet during contact phases. Non-linear learned detection models keep reasonably high accuracy while accuracy of linear models significantly decrease and accuracy of optimal thresholds (OT) baseline quickly collapse.

FOOTSKATE GENERATED WITH MOTION BLENDING. To push further our evaluation toward concrete applications, we additionally evaluate contact detection on motion sequences that have been blended. Motion blending (or motion interpolation) is a well-known technique widely used in animation that consists in mixing existing motions with dynamic blending weights to create new motions. In our case, motion blending is interesting because it is known to easily introduce footskate artefacts in resulting motions, which can then help us to evaluate contact detection on such perturbed motions. To do so, we randomly picked pairs of chunks from motion sequences in our test set having identical foot contact patterns on either left or right foot. Making the reasonable hypothesis that foot contact patterns are preserved when motion sequences with identical foot contact patterns are blended together, we can then blend our random pairs of chunks and thus introduce footskate artefacts while still having ground truth foot contacts needed to evaluate foot contact detection. We constituted a

set of approximately 40 thousands blended chunks of 80 frames long (i.e. 0.8 s) and quantified the amount of footskate introduced using the mean horizontal velocity of the feet during contact phases [106, 161, 163, 207]. Then, we attempt to detect foot contacts using the same models as before, and measure the performances with the F_1 score. Finally, we obtain and plot corresponding F_1 score curves in Figure 5.10 using simple moving average.

As depicted in Figures 5.8 to 5.10, the proposed approach for foot contact labels detection is much more robust than threshold-based heuristics approaches represented by the OT baseline, regardless of the type of perturbation applied to motion sequences. Moreover, the improvement of modelling vGRFs instead of foot contact labels (*Ours* vs *Ours*-C) is much larger when facing perturbed motion sequences. Since vGRFs are much more related to motion dynamics than binary contact label, vGRFs estimation requires a deeper understanding of motion than contact labels detection. Indeed, approximately 90% of contact labels can be correctly detected with foot position and velocity thresholding (see Table 5.2), i.e. with almost no understanding. Then, in challenging conditions like perturbed input motion sequences, the performances of the variant modelling vGRFs are logically more stable since a deeper understanding of motion is intuitively more robust.

5.6 FOOTSKATE CLEANUP

In this section, we leverage our robust foot contact detection for a downstream task and propose a novel fully automatic workflow for footskate cleanup. After detecting foot contact labels, our goal here is remove footskate artefacts by setting the feet static on the ground during detected contact phases, which can be seen as a particular case of IK. Traditional IK algorithms often rely on numerical optimisation to reach the target(s). Few works based on deep learning have been proposed, such as the framework of Victor et al. [173] for interactive design which allows to edit end-effectors locations in a static pose while keeping a globally consistent pose. In our case, we rely on an hybrid optimisation-based approach in which our deep network is used to enforce preservation of vGRFs throughout the optimisation.

Let $\tilde{X} = (\tilde{Q}, S, \tilde{P})$ be the input motion sequence suffering from footskate, where $\tilde{Q}$ denote joint angles, S the skeleton and $\tilde{P}$ the global trajectory positions of the root joint. Our goal is to find Q and P such that $X = (Q, S, P)$ is the footskate-cleaned version. First, we compute $\tilde{F} = \Psi(FK(\tilde{Q}, S, \tilde{P}))$ and $C = \Gamma(\tilde{F})$, being respectively the vGRF components estimated by our deep neural network Ψ and the foot contact labels calculated using the contact function Γ defined in Section 5.3.3. The function $FK(\cdot)$ refers to forward kinematics, i.e. the computation of joint positions from joint angles, skeleton and global trajectory. After initialising Q and P with $\tilde{Q}$ and $\tilde{P}$, we iteratively optimise both for a small fixed number N of iterations to best satisfy foot contact constraints into a gradient-based optimisation loop. In the following paragraphs we describe each term $\mathcal{L}_i$ of the our loss function

$$\mathcal{L} = \omega_q \mathcal{L}_{quat} + \omega_f \mathcal{L}_{foot} + \omega_t \mathcal{L}_{traj} + \omega_v \mathcal{L}_{vGRFs} \tag{5.2}$$

where weights ω_i are hyperparameters to balance our objectives.

QUATERNIONS LOSS. We parameterise joint angles with unit quaternions, which are well-suited for such an optimisation since they are free of singularities, computationally efficient, and numerically stable [124]. To keep valid rotations with quaternions throughout the optimisation, $\mathcal{L}_{quat}$ penalise quaternion norm deviations from 1:

$$\mathcal{L}_{quat} = \sum_t \sum_j \left(\|Q_j(t)\| - 1 \right)^2 \tag{5.3}$$

where $Q_j(t)$ is the quaternion representing the orientation of joint j at time t.

FOOT CONTACTS LOSS. Satisfying foot contact constraints is not an easy task since we do not have directly access to actual contact locations with respect to the skeleton. Moreover, since *foot joints* are located above actual contact points, they are allowed to rotate around the latter during part of contact phases (e.g. when the heel is in contact with the ground, the ankle is constrained to rotate around). To alleviate these issues, we artificially insert *contact joints* in S corresponding to contact points, i.e. under

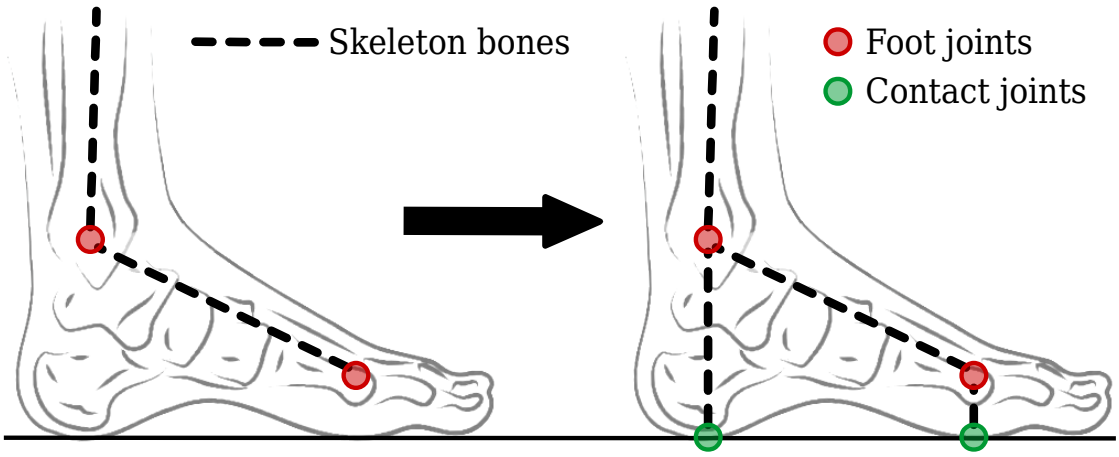


Figure 5.11 – Illustration of *contact joints* (green dots) inserted under corresponding *foot joints* at the ground level in the skeleton to properly enforce foot contact constraints (See details below).

foot joints at the ground level (see Figure 5.11). We assume that each contact joint is located vertically under its foot joint in the T-pose at the ground height, has constant orientation with respect to its foot joint, and is assumed to be static during contact phases. Then, imposing zero velocity on contact joints instead of foot joints during the contact phases better reflects the kinematics of the interaction with the ground, allowing slight rotation of foot joints located above the ground level. To this end, we minimise the mean squared distance from contact joints to contact positions during contact phases to faithfully constrain foot joint positions through forward kinematics:

$$\mathcal{L}_{foot} = \sum_{c=1}^{C} \sum_{t=t_0(c)}^{t_1(c)} \Pi(t) \left\| FK(Q,S,P)_{j_c}(t) - p_c \right\|^2 \tag{5.4}$$

where j_c and p_c are respectively the joint in contact with the ground during contact phase c and its contact position during the contact phase spanning from $t_0(c)$ to $t_1(c)$. Moreover, function $\Pi(\cdot)$ is a rectangular wave function with smoothed edges (synchronised with contact phases) to avoid sharp foot position changes.

TRAJECTORY LOSS. Since foot contact constraints are not guaranteed to be reachable by leg extensions, the global trajectory might be affected. However, it is closely related

to body dynamics and its optimisation must be carefully controlled to avoid artefacts. In particular, constraining trajectory positions might introduce implausible velocity changes, hurting motion dynamics realism, e.g. slowing down to reach some contact position might lead to speeding up afterwards to catch up the target trajectory. To this end, we minimise the trajectory velocity deviations instead of position deviations:

$$\mathcal{L}_{traj} = \sum_t (\|v(t)\| - \|\tilde{v}(t)\|)^2, \quad v(t) = \frac{P(t+\Delta T) - P(t)}{\Delta T} \tag{5.5}$$

where $v(t)$ is the global velocity of root joint at time t, computed from global trajectory position P by numerical differentiation.

VGRF INVARIANCE LOSS. The main novelty in our gradient-based optimisation approach is the use of our deep neural network through which gradients can flow. Since it is able to estimate consistent vGRF components from either clean or perturbed inputs, we further guide the optimisation by minimising the deviation between initially estimated vGRFs components $\tilde{F}$ and dynamically estimated vGRF components $F = \Psi(FK(Q,S,P))$:

$$\mathcal{L}_{vGRFs} = MSLE(F,\tilde{F}) \tag{5.6}$$

where the MSLE measures vGRFs deviation as previously (see Section 5.4.3).

We used the same motion sequences blended for evaluation purposes in Section 5.5.4 to test the proposed footskate cleanup workflow. With parameters $N = 100$, $\omega_q = 10^{-3}$, $\omega_f = 10^{-5}$, $\omega_t = 10^2$ and $\omega_v = 5 \cdot 10^{-5}$ (see Equation (5.2)), we achieve to significantly reduce the amount of footskate (the velocity of the feet during contact phases is approximately halved) while noticeably improving the realism of the sequences, assessed by visual inspection.

5.7 CONCLUSION

In this chapter, we presented a novel approach that improves the state-of-art of foot contact detection and footskate cleanup in human character animation. Building on UNDERPRESSURE, a novel motion capture database synchronised with pressure insoles which we publicly released, we proposed to learn the relationship between human motion and interactions between feet and ground with a deep neural network estimating vertical ground reaction forces, from which foot contact constraints can be derived and exploited for an effective removal of foot sliding artefacts.

As of today, the proposed database is one of a kind in animation as it provides synchronised motion capture and pressure insoles data for a wide variety of human motion. Our approach significantly outperforms thresholds-based heuristic approaches in detecting foot contact labels, which are limited in their capability to generalise. We show its robustness to perturbed input motion sequences, which is crucial in concrete use cases. We have shown that estimating forces greatly helps the accurate and robust detection of contacts. Finally, we demonstrate the usefulness of our foot contact detection approach for footskate cleanup, leveraging vGRF estimates to improve the quality of the results.

As described in Section 5.2, foot contact labels are relevant to a lot of tasks in human animation and are typically obtained with simple heuristics, or manually annotated on an exceptional basis. Thus, many of these downstream tasks could probably benefit from the improved accuracy of our foot contact detection approach. Moreover, motion reconstruction relies more and more on modelling physical interaction of the feet with the ground (see Section 5.2.2). Resolving inverse dynamics problems like ground reaction forces estimation from motion capture data could provide a valuable regularisation for such underconstrained problems as well as other tasks involving physics-based models. We believe our approach is a step in that direction.

6

CONCLUSION

In the last decade, artificial intelligence has revolutionised the topic of skeletal character animation as many others, especially through deep learning. Novel frameworks, e.g. leveraging variational autoencoder (VAE) [10, 35, 51, 195] or generative adversarial network (GAN) [19, 52, 112, 194], pushed generative motion synthesis up to unprecedented realism. Sophisticated schemes such as mixture-of-expert networks [207] or local motion phases [161] yielded breakthroughs in the control of virtual characters. The examples are numerous. Though, these advances still lack quality, flexibility and ability to generalise to fully outperform skilled animators' manual work quality and to replace them when dealing with tedious low-creativity tasks. Therefore, the purpose of this thesis is to tackle current obstacles preventing them from doing so.

In particular, we strived to alleviate the lack of high-quality motion data in the context of deep learning and the inclination of neural networks to introduce artefacts when synthesising or processing motion data. To this end, we first investigated the enhancement of 2D human pose sequences estimated from video using prior knowledge captured in a deep generative model. Then, we proposed a novel versatile deep representation of human motion, in which motion features are abstracted out from skeleton structure and morphology, allowing for gathering, processing or retargeting motion sequences with variable skeleton topologies. Finally, we tackled foot contact detection to automatically clean up footskate artefacts, well-known in character animation.

6.1 SUMMARY OF CONTRIBUTIONS

JOINT UPSAMPLING AND COMPLETION. Human pose estimation attempts to capture human body pose or motion from image or video, respectively. Requiring little equipment, it is a promising alternative to traditional motion capture systems which are expensive and cumbersome. Hence, capturing high-quality motion data from video

might vastly tackle the lack of human motion data, especially experienced with recent data-greedy models such as transformers. Since many 3D human pose estimation solutions rely on 2D pose estimation as a first step and solve inverse mapping from 2D to 3D afterwards, we presented in Chapter 3 a method to upsample and complete (e.g. missing joints due to occlusions) human joints in 2D as an intermediate step. Indeed, completed 2D pose sequences with higher spatial resolution disambiguate inverse mapping. Moreover, in such a use case, the higher resolution of joints would be reflected in estimated 3D pose sequences. To this end, we leveraged a deep generative model to learn the distribution 2D human pose sequences at high resolution of joints. At inference, completion and upsampling are performed by finding the point in the latent space of our model which best matches observed 2D low-resolution estimates, and then generating the corresponding completed high-resolution pose sequences. As our model is based on a GAN, this process is performed through optimisation in the latent space of our model, and high-resolution pose sequences are produced by the generator module of our model. We showed that our method is able to upsample joints in 2D pose estimates while keeping localisation accuracy of the underlying pose estimator. Moreover, prior knowledge on human motion learnt by our model allows to plausibly fill occluded joints (e.g. when a limb passes behind the body).

UNIFIED HUMAN MOTION REPRESENTATION. Another way to tackle the bottleneck constituted by the limited availability of motion data is to increase the interoperability of existing human motion databases, which are difficult to gather and exploit together. For example, combining a limited amount of motion data annotated with style labels with a larger general motion database might be more thoughtful when developing a style transfer method than acquiring a large amount of motion data annotated with style labels. However, combining existing databases is not straightforward since virtual characters often have different sets of joints and interconnections and might even be as tedious as acquiring a large amount of novel motion data. To overcome this issue, we proposed in Chapter 4 a novel deep representation of human motion unified across skeleton topologies and morphologies. We learnt this representation with a transformer-based neural network in an autoencoder scheme from a large amount of

motion data collected from different existing motion databases. We condition both encoder and decoder on skeleton topological and morphological features allowing to abstract "pure" motion features in our deep representation. The effectiveness of transformers to discover complex patterns from data is known to be unprecedented. In our case, it overcomes the difficulty to learn such a unified human motion representation, which itself makes the training of our transformer-based architecture possible. Indeed, transformers require large data volumes but our unified model can be learnt from the gathering of multiple existing motion databases. As a result, our deep motion representation constitutes an interesting space to embed and work with motion sequences from different sources. Moreover, the effectiveness of transformers coupled with the generality of the approach makes our model well-suited for a variety of downstream applications including motion retargeting, motion denoising and joint upsampling.

FOOT CONTACT DETECTION AND FOOTSKATE CLEANUP. Finally, we also investigated in this thesis the cleaning of artefacts. We believe that deep learning is also promising to effectively clean artefacts as post-processing rather than indefinitely increasing data quality, data volumes and deep model sizes to perfect the results obtained. In particular, we focused on footskate which is known to be detrimental to the perceived realism even at low intensities [138] and also easily introduced when processing motion data. Knowledge of foot contacts is required for footskate cleanup, however it is often obtained through manual annotation or using simple heuristics (e.g. hand-crafted foot velocity threshold). Therefore, we proposed a reliable foot contact detection solution on top of which we built an algorithm to automatically clean footskate artefacts. To this end, we first captured and publicly released a novel database of motion data synchronised with pressure insoles data, from which accurate foot contacts can be derived. Hence, this database provides a significant amount of motion data annotated with foot contacts compared to scarce existing motion data manually annotated. Then, we built a robust foot contact detection approach by training a deep neural network to estimate vertical ground reaction forces (vGRFs) from motion on our database. vGRFs serve as a proxy representation which captures physical interactions between feet and ground better than binary foot contact labels. At inference, binary foot contact labels

can be computed from vertical ground reaction forces (vGRFs). We demonstrate that our approach outperforms traditional heuristics-based solutions in terms of accuracy, generalisability and robustness. On top of our foot contact detection approach, we also presented a fully automatic workflow to remove footskate artefacts from motion sequences. Following recent approaches, footskate is removed by enforcing foot constraints (detected with our model) using an inverse kinematics (IK) algorithm, iteratively optimising joint angles to satisfy the constraints. We additionally leverage our deep model to preserve ground reaction forces during this optimisation process and keep motion dynamics globally consistent.

6.2 PERSPECTIVES AND FUTURE WORK

In this thesis, we have explored approaches in the general goal of improving deep learning based systems for skeleton character animation. In the following, we present possible research perspectives in the continuation of our work. First, we propose short-term perspectives, i.e. possible improvements and experiments which naturally stem from the work presented in this thesis. Then, we suggest a set of possible directions to extend our work inspired from existing literature in character animation and beyond. Finally, we open up the discussion to long-term perspectives requiring to solve several challenges and rethink part of the current paradigms in character animation.

6.2.1 *Short-Term Perspectives*

In Chapters 3 and 4 we have presented two approaches relying on deep motion representations. In the latter, we proposed an abstract motion representation independent of skeleton topology and morphology. However, as pointed out in Section 4.5, our model has some limitations with regard to morphological variations, including decreased performance on non-human morphologies.

A first perspective would be to investigate whether or not the proposed model is able to accurately encode both human and humanoid morphologies in a single

unified motion representation. In other words, is it possible to unify the motion of humanoid characters (i.e. two legs, two arms and a head but possibly exaggerated body proportions) in a deep representation learnt following our approach? Moreover, non-humanoid characters such as quadrupeds might also be explored in future work. A simple approach to do so that might be effective would be to diversify the training dataset beyond human motion sequences. However, other changes more involved might be needed to handle humanoid or other kind of characters, and are overviewed in the medium-term perspectives hereafter.

Another limitation of our work is that footskate artefacts are likely to be introduced in some cases. As of today, this issue concerns most, if not all, deep motion models used to produce or process motion data. A possible approach to improve in this direction would be to introduce knowledge about interactions between feet and ground. For instance, this could be possible by introducing some supervision with respect to foot contact or vertical ground reaction forces (vGRFs) during training. This can be directly implemented using the motion database proposed in Chapter 5 in which motion data is synchronised with pressure insole data. Several ways to add supervision are possible and can be combined. For example: ground truth foot contacts might be used to measure and minimise footskate in output motion sequences; vGRFs or foot contacts could be estimated from the latent representation of motion sequences; vGRFs or foot contacts consistency loss might be applied on vGRFs or foot contacts estimated from reconstructed or retargeted motion sequences. Moreover, the unified motion representation proposed in Chapter 4 and the foot contact detection model proposed in Chapter 5 might be directly combined to automatically clean footskate artefacts. The former would retarget motion sequences to the skeleton topology of the latter to obtain foot contacts needed to clean footskate.

Finally, the amount of footskate artefacts introduced is rarely evaluated. The reasons here are the scarcity of motion data annotated with accurate foot contacts, and the difficulty to obtain accurate ground truth foot contacts. The database and foot contact detection approach proposed in Chapter 5 could be leveraged to define some standard evaluation procedure of footskate.

6.2.2 *Medium-Term Perspectives*

A rising challenge in character animation is to propose methods that are versatile enough to be usable on characters with large topological and morphological variations, and possibly generalise beyond humanoid characters in some cases. Extending some of the deep motion representations proposed in this thesis to handle such large variations might therefore need to rethink the conditioning mechanism we proposed. In particular, we currently proposed to use neutral poses as skeleton templates, represented by their joint positions which might be limited to fully capture morphological and topological features. While this issue could be solved by a more thoughtful design, we believe that an elegant solution would consist in additionally learning a deep representation of skeleton topologies and morphologies together with a deep motion representation. Such an approach would have the benefit to avoid manually predefining skeleton templates while alleviating the limitations of our current approach. To further extend the scope of our approach, one might also need to consider personalising skeleton templates (either predefined or learnt) for each subject, i.e. more finely tuning our conditioning mechanism to better capture fine morphological variations, e.g. hip sexual dimorphism.

Beyond skeleton templates, novel deep learning models might be leveraged for improved learning. For instance, diffusion models [159] are very attractive to build deep representations useful for multiple tasks. Indeed, these models outperform GANs on image synthesis, and have also been promising in a variety of domains [197]. Moreover, diffusion models also improve on scalability and parallelisability. Finally, their progressive nature might be key to numerous applications that can be formulated as progressive transitions. These include most completion tasks, such as motion completion, in-betweening or joint upsampling, but also cleaning tasks such as denoising or footskate cleanup. The progressive nature of diffusion models might even extend the scope of current problems. For instance, motion-to-motion diffusion models (in analogy with image-to-image diffusion models [146]) could learn to progressively retarget a motion sequence from a character to another, notably allowing character morphing in motion sequences, i.e. progressively switching from one character to another over time.

There are also multiple perspectives in the direction of cleaning and preventing artefacts in output motion sequences. In the context of footskate cleanup in particular, one of the limitations comes from the binary representation of foot contacts which are then generally used to enforce foot constraints using inverse kinematics (IK) algorithms. Indeed, interactions between feet and ground are not trivial and modelling them as in or out of contact is overly simplified and conducts to manually tune IK algorithms for specific situations to avoid replacing footskate by other kinds of artefacts. A possible direction to solve this issue is to use a finer representation of feet and ground interactions, such as vGRFs, and design an IK algorithm specifically targeting footskate cleanup, e.g. by enforcing target profiles of vGRFs instead of binary in-contact phases. This is something that we have only slightly explored in this thesis, but which we believe shows promising potential for novel approaches. Another ambitious direction to prevent footskate artefacts would be to explicitly model feet and ground interactions together with motion, e.g. by building a common embedding of vGRFs and motion sequences similarly to visual semantic embedding in computer vision that are shared between text and images.

6.2.3 *Long-Term Perspectives*

In the long term, we envision that character animation might need to change paradigm to make animation synthesis and editing more accessible. This might include replacing current technical human-computer interfaces in animation systems by semantic controls more intuitive for designers and artists. In this direction, semantic embeddings are promising. Indeed, they have recently demonstrated impressive capabilities at the intersection of natural language processing and computer vision, e.g. OpenAI's novel image generation model *DALL·E 2* [141] which relies on a version of *GPT-3* [22] modified for image synthesis. In particular, semantic embeddings based on natural languages seem to be very efficient to yield powerful representations and capture necessary information for applications involving human-computer interactions. In character animation, a lot of applications include human-computer interactions at some point, such as motion synthesis and editing systems driven by artists and

animators or virtual characters controlled by gamers. For example, in analogy with capabilities of current image synthesis models, we could imagine to quickly produce draft motion sequences from textual descriptions, such as "a tall character running for 4 seconds then accelerating jump above an obstacle". However, building a semantic embedding for character animation based on natural language would require to solve several tough challenges. These challenges include the greediness in term of data volume of recent successful models such as GPT-3. In the case of images associated with text descriptors, billions of examples are available on the internet. On the contrary, motion data are currently much more scarce that images. Also, motion sequences are not often annotated with text descriptors in existing databases, needed to build such a semantic embedding of natural language and character motion. Another important challenge is that dedicated natural language model(s) might be necessary, as for DALL·E 2. Even though it might require less complexity than general natural language models like GPT-3, little work has been done in this direction. A recent attempt is the language-directed physical character controller proposed by Juravsky et al. [73].

We believe that the contributions presented in this thesis are a first step in such directions, and will open the door to novel approaches for animating virtual characters based on deep learning approaches.

BIBLIOGRAPHY

[1] Farzad Abdolhosseini et al. "On Learning Symmetric Locomotion". In: *International Conference on Motion, Interaction and Games*. Association for Computing Machinery (ACM), Oct. 2019. DOI: 10.1145/3359566.3360070 (cit. on p. 134).

[2] Kfir Aberman et al. "Learning Character-Agnostic Motion for Motion Retargeting in 2D". In: *ACM Transactions on Graphics* 38.4 (July 2019). DOI: 10.1145/3306346.3322999 (cit. on p. 100).

[3] Kfir Aberman et al. "Skeleton-Aware Networks for Deep Motion Retargeting". In: *ACM Transactions on Graphics* 39.4 (July 2020). DOI: 10.1145/3386569.3392462 (cit. on pp. 39, 40, 45, 48, 55, 63, 65, 67, 68, 98, 100, 118, 122–125, 128, 134).

[4] Kfir Aberman et al. "Unpaired Motion Style Transfer from Video to Animation". In: *ACM Transactions on Graphics* 39.4 (July 2020). DOI: 10.1145/3386569.3392469 (cit. on pp. 48, 63, 70).

[5] Adobe. *Mixamo*. Accessed: 2021-09-16. 2021. URL: https://www.mixamo.com (cit. on pp. 41, 42, 63).

[6] Moticon ReGo AG. *OpenGo Sensor Insole Specification*. Accessed: 2021-12-06. 2021. URL: https://moticon.com/wp-content/uploads/2021/09/OpenGo-Sensor-Insole-Specification-A4SQ-RGB-EN-03.02.pdf (cit. on p. 139).

[7] Emre Aksan, Manuel Kaufmann, and Otmar Hilliges. "Structured Prediction Helps 3D Human Motion Modelling". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society/Computer Vision Foundation (CVF), Oct. 2019, pp. 7143–7152. DOI: 10.1109/ICCV.2019.00724 (cit. on p. 44).

[8] Omid Alemi, William Li, and Philippe Pasquier. "Affect-Expressive Movement Generation with Factored Conditional Restricted Boltzmann Machines". In: *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*.

IEEE Computer Society, Sept. 2015, pp. 442–448. DOI: 10.1109/ACII.2015.7344608 (cit. on pp. 36, 51).

[9] Omid Alemi and Philippe Pasquier. "WalkNet: A Neural-Network-Based Interactive Walking Controller". In: *17th International Conference on Intelligent Virtual Agents*. Springer International Publishing, Aug. 2017, pp. 15–24. DOI: 10.1007/978-3-319-67401-8_2 (cit. on pp. 36, 59, 61).

[10] Sadegh Aliakbarian et al. "A Stochastic Conditioning Scheme for Diverse Human Motion Prediction". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2020, pp. 5223–5232. DOI: 10.1109/CVPR42600.2020.00527 (cit. on pp. 46, 50, 52, 53, 160).

[11] Eduardo Alvarado, Damien Rohmer, and Marie-Paule Cani. "Generating Upper-Body Motion for Real-Time Characters Making their Way through Dynamic Environments". In: *Computer Graphics Forum* 41.8 (Sept. 2022). DOI: 10.1111/cgf.14633 (cit. on p. 62).

[12] Andrea Ancillao et al. "Indirect Measurement of Ground Reaction Forces and Moments by Means of Wearable Inertial Sensors: A Systematic Review". In: *Sensors* 18.8 (Aug. 2018). DOI: 10.3390/s18082564 (cit. on p. 135).

[13] Mykhaylo Andriluka et al. "2D Human Pose Estimation: New Benchmark and State of the Art Analysis". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2014, pp. 3686–3693. DOI: 10.1109/CVPR.2014.471 (cit. on p. 88).

[14] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research (PMLR), Aug. 2017, pp. 214–223. URL: https://proceedings.mlr.press/v70/arjovsky17a.html (cit. on p. 81).

[15] Bruno Artacho and Andreas Savakis. "UniPose: Unified Human Pose Estimation in Single Images and Videos". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foun-

dation (CVF), June 2020, pp. 7033–7042. DOI: 10.1109/CVPR42600.2020.00706 (cit. on p. 76).

[16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". July 2016. eprint: 1607.06450 (cit. on p. 48).

[17] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". Sept. 2014. eprint: 1409.0473 (cit. on p. 54).

[18] Jianmin Bao et al. "CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Oct. 2017, pp. 2764–2773. DOI: 10.1109/ICCV.2017.299 (cit. on p. 78).

[19] 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)*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2018, pp. 1499–1509. DOI: 10.1109/CVPRW.2018.00191 (cit. on pp. 46, 50, 54, 160).

[20] Rama Bindiganavale and Norman I. Badler. "Motion Abstraction and Mapping with Spatial Constraints". In: *Modelling and Motion Capture Techniques for Virtual Environments*. Springer International Publishing, Mar. 1998, pp. 70–82. DOI: 10.1007/3-540-49384-0_6 (cit. on pp. 65, 134).

[21] Michael Bronstein et al. "Geometric Deep Learning: Going beyond Euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (Nov. 2017), pp. 18–42. DOI: 10.1109/MSP.2017.2693418 (cit. on p. 44).

[22] Tom B. Brown et al. "Language Models are Few-Shot Learners". May 2020. eprint: 2005.14165 (cit. on p. 166).

[23] Judith Bütepage et al. "Deep representation learning for human motion prediction and classification". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, July 2017, pp. 1591–1599. DOI: 10.1109/CVPR.2017.173 (cit. on pp. 44, 63, 64).

[24] Zhe Cao et al. "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.1 (Jan. 2019), pp. 172–186. DOI: 10.1109/TPAMI.2019.2929257 (cit. on pp. 66, 76).

[25] Jinxiang Chai and Jessica K. Hodgins. "Constraint-Based Motion Optimization Using a Statistical Dynamic Model". In: *ACM Transactions on Graphics* 26.3 (July 2007), 8–17. DOI: 10.1145/1276377.1276387 (cit. on p. 56).

[26] Prashanth Chandran et al. "Facial Animation with Disentangled Identity and Motion using Transformers". In: *Computer Graphics Forum* 41.8 (Sept. 2022). DOI: 10.1111/cgf.14641 (cit. on p. 102).

[27] Prashanth Chandran et al. "Shape Transformers: Topology-Independent 3D Shape Models Using Transformers". In: *Computer Graphics Forum* 41.2 (May 2022), pp. 195–207. DOI: 10.1111/cgf.14468 (cit. on pp. 102, 107, 109).

[28] Chung-Cheng Chiu and Stacy Marsella. "A style controller for generating virtual human behaviors". In: *Tenth International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems (AAMAS), May 2011, pp. 1023–1030. URL: http://www.ifaamas.org/Proceedings/aamas2011/papers/D7_G77.pdf (cit. on pp. 36, 51).

[29] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. "On the Relationship between Self-Attention and Convolutional Layers". Nov. 2019. eprint: 1911.03584 (cit. on p. 102).

[30] Qiongjie Cui, Huaijiang Sun, and Fei Yang. "Learning Dynamic Relationships for 3D Human Motion Prediction". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2020, pp. 6519–6527. DOI: 10.1109/CVPR42600.2020.00655 (cit. on p. 45).

[31] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *2019 Annual Conference of the North American Chapter of he Association for Computational Linguistics*. Association for Computa-

tional Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/n19-1423 (cit. on p. 109).

[32] Laurent Dinh, David Krueger, and Yoshua Bengio. "NICE: Non-linear Independent Components Estimation". Oct. 2014. eprint: 1410.8516 (cit. on p. 55).

[33] Yuzhu Dong et al. "Adult2child: Motion Style Transfer using CycleGANs". In: *International Conference on Motion, Interaction and Games*. Association for Computing Machinery (ACM), Oct. 2020. DOI: 10.1145/3424636.3426909 (cit. on pp. 48, 63, 71).

[34] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". Oct. 2020. eprint: 2010.11929 (cit. on p. 102).

[35] Han Du et al. "Stylistic Locomotion Modeling and Synthesis using Variational Generative Models". In: *International Conference on Motion, Interaction and Games*. Association for Computing Machinery (ACM), Oct. 2019. DOI: 10.1145/3359566.3360083 (cit. on pp. 50, 52, 160).

[36] Yinglin Duan et al. "Single-Shot Motion Completion with Transformer". Mar. 2021. eprint: 2103.00776 (cit. on pp. 48, 50, 57, 102).

[37] Alaaeldin El-Nouby et al. "XCiT: Cross-Covariance Image Transformers". In: *35th International Conference on Neural Information Processing Systems*. Vol. 34. Curran Associates Inc., Dec. 2021. URL: https://proceedings.neurips.cc/paper/2021/file/a655fbe4b8d7439994aa37ddad80de56-Paper.pdf (cit. on p. 111).

[38] Moataz Eltoukhy et al. "Prediction of ground reaction forces for Parkinson's disease patients using a kinect-driven musculoskeletal gait analysis model". In: *Medical Engineering & Physics* 50 (Oct. 2017), pp. 75–82. DOI: 10.1016/j.medengphy.2017.10.004 (cit. on p. 135).

[39] Gert S. Faber et al. "Estimating 3D L5/S1 moments and ground reaction forces during trunk bending using a full-body ambulatory inertial motion capture system". In: *Journal of Biomechanics* 49.6 (Nov. 2015), pp. 904–912. DOI: 10.1016/j.jbiomech.2015.11.042 (cit. on p. 135).

[40] Hao-Shu Fang et al. "RMPE: Regional Multi-Person Pose Estimation". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Oct. 2017, pp. 2353–2362. DOI: 10.1109/ICCV.2017.256 (cit. on p. 88).

[41] Katerina Fragkiadaki et al. "Recurrent Network Models for Human Dynamics". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Dec. 2015, pp. 4346–4354. DOI: 10.1109/ICCV.2015.494 (cit. on pp. 57, 62, 78).

[42] Andrew C. Fry et al. "Validation of a motion capture system for deriving accurate ground reaction forces without a force plate". In: *Big Data Analytics* 1 (Sept. 2016). DOI: 10.1186/s41044-016-0008-y (cit. on p. 135).

[43] Leon Gatys, Alexander Ecker, and Matthias Bethge. "Image Style Transfer Using Convolutional Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2016, pp. 2414–2423. DOI: 10.1109/CVPR.2016.265 (cit. on p. 69).

[44] Michael Gleicher. "Retargetting Motion to New Characters". In: *25th International Conference on Computer Graphics and Interactive Techniques*. Association for Computing Machinery (ACM), Mar. 1998, pp. 33–42. DOI: 10.1145/280814.280820 (cit. on pp. 23, 67, 98, 100).

[45] Aman Goel, Qianhui Men, and Edmond Shu-lim Ho. "Interaction Mix and Match: Synthesizing Close Interaction using Conditional Hierarchical GAN with Multi-Hot Class Embedding". In: *Computer Graphics Forum* 41.8 (Sept. 2022). DOI: 10.1111/cgf.14647 (cit. on pp. 43, 47, 50, 54).

[46] Ian Goodfellow et al. "Generative Adversarial Nets". In: *27th International Conference on Neural Information Processing Systems*. The MIT Press, Dec. 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423.pdf (cit. on pp. 78, 81).

[47] F. Sebastian Grassia. "Practical Parameterization of Rotations Using the Exponential Map". In: *Journal of Graphics Tools* 3.3 (Apr. 1998), pp. 29–48. DOI: 10.1080/10867651.1998.10487493 (cit. on pp. 35–37).

[48] Ishaan Gulrajani et al. "Improved Training of Wasserstein GANs". In: *31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2017, pp. 5767–5777. URL: https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf (cit. on pp. 81, 82, 87).

[49] Xiao Guo and Jongmoo Choi. "Human Motion Prediction via Learning Local Structure Representations and Temporal Dependencies". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.1 (July 2019). DOI: 10.1609/aaai.v33i01.33012580 (cit. on pp. 43, 47).

[50] Sehoon Ha, Yunfei Bai, and C. Karen Liu. "Human Motion Reconstruction from Force Sensors". In: *2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Association for Computing Machinery (ACM), Aug. 2011, 129–138. DOI: 10.1145/2019406.2019424 (cit. on p. 135).

[51] Ikhsanul Habibie et al. "A Recurrent Variational Autoencoder for Human Motion Synthesis". In: *British Machine Vision Conference 2017*. BMVA Press, Sept. 2017, pp. 1–12. DOI: 10.5244/C.31.119 (cit. on pp. 50, 52, 78, 160).

[52] Félix Gingras Harvey et al. "Robust Motion In-betweening". In: *ACM Transactions on Graphics* 39.4 (July 2020). DOI: 10.1145/3386569.3392480 (cit. on pp. 46, 50, 54, 56–58, 65, 66, 78, 134, 160).

[53] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Dec. 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123 (cit. on p. 23).

[54] Chris Hecker et al. "Real-Time Motion Retargeting to Highly Varied User-Created Morphologies". In: *ACM Transactions on Graphics* 27.3 (Aug. 2008), 1–11. DOI: 10.1145/1360612.1360626 (cit. on p. 118).

[55] Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. "MoGlow: Probabilistic and controllable motion synthesis using normalising flows". In: *ACM Transactions on Graphics* 39.6 (Nov. 2020). DOI: 10.1145/3414685.3417836 (cit. on pp. 46, 50, 55, 134).

[56] Alejandro Hernandez Ruiz, Jürgen Gall, and Francesc Moreno. "Human Motion Prediction via Spatio-Temporal Inpainting". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society/Computer Vision Foundation (CVF), Oct. 2019, pp. 7133–7142. DOI: 10.1109/ICCV.2019.00723 (cit. on p. 78).

[57] Daniel Holden. "Robust Solving of Optical Motion Capture Data by Denoising". In: *ACM Transactions on Graphics* 37.4 (July 2018). DOI: 10.1145/3197517.3201302 (cit. on pp. 63, 64).

[58] Daniel Holden, Taku Komura, and Jun Saito. "Phase-Functioned Neural Networks for Character Control". In: *ACM Transactions on Graphics* 36.4 (July 2017). DOI: 10.1145/3072959.3073663 (cit. on pp. 36, 40, 49, 59, 65, 66, 134, 140).

[59] Daniel Holden, Jun Saito, and Taku Komura. "A Deep Learning Framework for Character Motion Synthesis and Editing". In: *ACM Transactions on Graphics* 35.4 (July 2016). DOI: 10.1145/2897824.2925975 (cit. on pp. 41, 42, 50, 52, 55, 63, 69, 134, 152).

[60] Daniel Holden et al. "Fast Neural Style Transfer for Motion Data". In: *IEEE Computer Graphics and Applications* 37.4 (Aug. 2017), pp. 42–49. DOI: 10.1109/MCG.2017.3271464 (cit. on pp. 48, 63, 70).

[61] Daniel Holden et al. "Learned Motion Matching". In: *ACM Transactions on Graphics* 39.4 (July 2020). DOI: 10.1145/3386569.3392440 (cit. on pp. 40, 59, 62).

[62] Daniel Holden et al. "Learning Motion Manifolds with Convolutional Autoencoders". In: *SIGGRAPH Asia 2015 Technical Briefs*. Association for Computing Machinery (ACM), Mar. 2015. DOI: 10.1145/2820903.2820918 (cit. on pp. 60, 63, 64).

[63] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and Locally Consistent Image Completion". In: *ACM Transactions on Graphics* 36.4 (Aug. 2017). DOI: 10.1145/3072959.3073659 (cit. on p. 78).

[64] Leslie Ikemoto, Okan Arikan, and David Forsyth. "Knowing When to Put Your Foot Down". In: *2006 Symposium on Interactive 3D Graphics and Games*. Association

for Computing Machinery (ACM), Mar. 2006, 49–53. DOI: 10.1145/1111411.1111420 (cit. on pp. 65, 134, 140).

[65] Eldar Insafutdinov et al. "DeeperCut: A Deeper, Stronger, and Faster Multi-person Pose Estimation Model". In: *14th European Conference on Computer Vision (ECCV)*. Springer International Publishing, Oct. 2016, pp. 34–50. DOI: 10.1007/978-3-319-46466-4_3 (cit. on p. 89).

[66] Catalin Ionescu et al. "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (July 2014), pp. 1325–1339. DOI: 10.1109/TPAMI.2013.248 (cit. on pp. 41, 50, 63, 83, 112, 113).

[67] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, July 2017, pp. 5967–5976. DOI: 10.1109/CVPR.2017.632 (cit. on p. 55).

[68] Ashesh Jain et al. "Structural-RNN: Deep Learning on Spatio-Temporal Graphs". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2016, pp. 5308–5317. DOI: 10.1109/CVPR.2016.573 (cit. on p. 43).

[69] Deok-Kyeong Jang and Sung-Hee Lee. "Constructing Human Motion Manifold with Sequential Networks". In: *Computer Graphics Forum* 39.2 (July 2020), pp. 487–496. DOI: 10.1111/cgf.14028 (cit. on pp. 36, 46, 63, 64).

[70] Deok-Kyeong Jang, Soomin Park, and Sung-Hee Lee. "Motion Puzzle: Arbitrary Motion Style Transfer by Body Part". In: *ACM Transactions on Graphics* 41.3 (June 2022). DOI: 10.1145/3516429 (cit. on pp. 63, 70).

[71] Ananya Harsh Jha et al. "Disentangling Factors of Variation with Cycle-Consistent Variational Auto-encoders". In: *15th European Conference on Computer Vision (ECCV)*. Springer International Publishing, Sept. 2018, pp. 829–845. DOI: 10.1007/978-3-030-01219-9_49 (cit. on p. 82).

[72] Yihwan Jung et al. "Ground reaction force estimation using an insole-type pressure mat and joint kinematics during walking". In: *Journal of Biomechanics* 47.11 (May 2014), pp. 2693–2699. DOI: 10.1016/j.jbiomech.2014.05.007 (cit. on p. 135).

[73] Jordan Juravsky et al. "PADL: Language-Directed Physics-Based Character Control". In: *SIGGRAPH Asia 2022*. Association for Computing Machinery (ACM), Dec. 2022. DOI: 10.1145/3550469.3555391 (cit. on p. 167).

[74] Maria Jönsson et al. "Foot centre of pressure and ground reaction force during quadriceps resistance exercises; a comparison between force plates and a pressure insole system". In: *Journal of Biomechanics* 87 (Mar. 2019), pp. 206–210. DOI: 10.1016/j.jbiomech.2019.03.004 (cit. on p. 150).

[75] Angelos Karatsidis et al. "Estimation of Ground Reaction Forces and Moments During Gait Using Only Inertial Motion Capture". In: *Sensors* 17.1 (Dec. 2016). DOI: 10.3390/s17010075 (cit. on p. 135).

[76] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2019, pp. 4396–4405. DOI: 10.1109/CVPR.2019.00453 (cit. on p. 110).

[77] Tero Karras et al. "Alias-Free Generative Adversarial Networks". In: *35th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2021, pp. 852–863. URL: https://proceedings.neurips.cc/paper/2021/file/076ccd93ad68be51f23707988e934906-Paper.pdf (cit. on p. 110).

[78] Tero Karras et al. "Analyzing and Improving the Image Quality of StyleGAN". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2020, pp. 8107–8116. DOI: 10.1109/CVPR42600.2020.00813 (cit. on p. 110).

[79] Tero Karras et al. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *International Conference on Learning Representations*. May 2016. eprint: 1710.10196 (cit. on pp. 55, 85).

[80] Manuel Kaufmann et al. "Convolutional Autoencoders for Human Motion Infilling". In: *2020 International Conference on 3D Vision (3DV)*. IEEE Computer Society, Nov. 2020, pp. 918–927. DOI: 10.1109/3DV50981.2020.00102 (cit. on pp. 50, 56).

[81] SangBin Kim et al. "Motion Retargetting based on Dilated Convolutions and Skeleton-specific Loss Functions". In: *Computer Graphics Forum* 39.2 (July 2020), pp. 497–507. DOI: 10.1111/cgf.13947 (cit. on pp. 48, 63, 67, 68, 100, 118, 122).

[82] Diederik P. Kingma and Jimmy Lei Ba. "Adam: A method for stochastic optimization". Dec. 2014. eprint: 1412.6980 (cit. on pp. 87, 118, 144).

[83] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations*. Apr. 2014. URL: https://openreview.net/forum?id=33X9fd2-9FyZd (cit. on pp. 51, 78).

[84] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations*. Apr. 2017. URL: https://openreview.net/forum?id=SJU4ayYgl (cit. on p. 45).

[85] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. "Motion Graphs". In: *29th International Conference on Computer Graphics and Interactive Techniques*. Vol. 21. 3. Association for Computing Machinery (ACM), July 2002, pp. 473–482. DOI: 10.1145/566654.566605 (cit. on p. 23).

[86] Lucas Kovar, John Schreiner, and Michael Gleicher. "Footskate Cleanup for Motion Capture Editing". In: *2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Association for Computing Machinery (ACM), Mar. 2002, 97–104. DOI: 10.1145/545261.545277 (cit. on pp. 64, 134).

[87] Marek Kulbacki, Jakub Segen, and Jerzy Paweł Nowacki. "4GAIT: Synchronized MoCap, Video, GRF and EMG Datasets: Acquisition, Management and Applications". In: *6th Asian Conference on Intelligent Information and Database Systems*. Springer International Publishing, Apr. 2014, pp. 555–564. DOI: 10.1007/978-3-319-05458-2_57 (cit. on p. 137).

[88] Richard Kulpa, Franck Multon, and Bruno Arnaldi. "Morphology-independent representation of motions for interactive human-like animation". In: *Computer Graphics Forum* 24.3 (Oct. 2005), pp. 343–351. DOI: 10.1111/j.1467-8659.2005.00859.x (cit. on p. 118).

[89] Shigeru Kuriyama et al. "Context-based Style Transfer of Tokenized Gestures". In: *Computer Graphics Forum* 41.8 (Sept. 2022). DOI: 10.1111/cgf.14645 (cit. on pp. 48, 63, 71, 102).

[90] Taesoo Kwon, Yoonsang Lee, and Michiel van de Panne. "Fast and Flexible Multilegged Locomotion Using Learned Centroidal Dynamics". In: *ACM Transactions on Graphics* 39.4 (July 2020). DOI: 10.1145/3386569.3392432 (cit. on p. 134).

[91] Benoît Le Callennec and Ronan Boulic. "Robust Kinematic Constraint Detection for Motion Data". In: *2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Association for Computing Machinery (ACM), Mar. 2006. URL: https://dl.acm.org/doi/10.5555/1218064.1218103 (cit. on pp. 65, 134).

[92] Jehee Lee et al. "Interactive Control of Avatars Animated with Human Motion Data". In: *29th International Conference on Computer Graphics and Interactive Techniques*. Association for Computing Machinery (ACM), Mar. 2002. DOI: 10.1145/566570.566607 (cit. on pp. 65, 134).

[93] Kyungho Lee, Seyoung Lee, and Jehee Lee. "Interactive Character Animation by Learning Multi-Objective Control". In: *ACM Transactions on Graphics* 37.6 (Dec. 2018). DOI: 10.1145/3272127.3275071 (cit. on pp. 40, 46, 59, 62, 134).

[94] Gustavo Leporace et al. "Residual Analysis of Ground Reaction Forces Simulation during Gait Using Neural Networks with Different Configurations". In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE Computer Society, Nov. 2015, pp. 2812–2815. DOI: 10.1109/EMBC.2015.7318976 (cit. on p. 135).

[95] Chen Li et al. "Convolutional Sequence to Sequence Model for Human Dynamics". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2018, pp. 5226–5234. DOI: 10.1109/CVPR.2018.00548 (cit. on p. 44).

[96] Chuan Li and Michael Wand. "Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks". In: *14th European Conference on Computer Vision (ECCV)*. Springer International Publishing, Oct. 2016, pp. 702–716. DOI: 10.1007/978-3-319-46487-9_43 (cit. on p. 55).

[97] Jiefeng Li et al. "CrowdPose: Efficient Crowded Scenes Pose Estimation and A New Benchmark". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2019, pp. 10855–10864. DOI: 10855-10864 (cit. on p. 88).

[98] Peizhuo Li et al. "GANimator: Neural Motion Synthesis from a Single Sequence". In: *ACM Transactions on Graphics* 41.4 (July 2022). DOI: 10.1145/3528223.3530157 (cit. on pp. 38, 50, 55).

[99] Shu-Jie Li et al. "A Perceptual-Based Noise-Agnostic 3D Skeleton Motion Data Refinement Network". In: *IEEE Access* 8 (Mar. 2020), pp. 52927–52940. DOI: 10.1109/ACCESS.2020.2980316 (cit. on pp. 47, 60, 63, 64, 114).

[100] Shu-Jie Li et al. "An Iterative Solution for Improving the Generalization Ability of Unsupervised Skeleton Motion Retargeting". In: *Computers & Graphics* 104 (Apr. 2022), pp. 129–139. DOI: 10.1016/j.cag.2022.04.001 (cit. on pp. 63, 67, 100, 118, 122).

[101] Shu-Jie Li et al. "Bidirectional recurrent autoencoder for 3D skeleton motion data refinement". In: *Computers & Graphics* 81 (Apr. 2019), pp. 92–103. DOI: 10.1016/j.cag.2019.03.010 (cit. on pp. 47, 63, 64, 114).

[102] Yanran Li et al. "Efficient convolutional hierarchical autoencoder for human motion prediction". In: *The Visual Computer* 35.6 (June 2019), pp. 1143–1156. DOI: 10.1007/s00371-019-01692-9 (cit. on p. 44).

[103] Zongmian Li et al. "Estimating 3D Motion and Forces of Person-Object Interactions from Monocular Video". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foundation (CVF), Mar. 2019, pp. 8640–8649. DOI: 10.1109/CVPR.2019.00884 (cit. on p. 135).

[104] Jongin Lim, Hyung Jin Chang, and Jin Young Choi. "PMnet: Learning of Disentangled Pose and Movement for Unsupervised Motion Retargeting". In: *30th British Machine Vision Conference 2019*. BMVA Press, Sept. 2019, pp. 1–13. URL: https://bmvc2019.org/wp-content/uploads/papers/0997-paper.pdf (cit. on pp. 48, 63, 67, 100, 118, 122).

[105] Kevin Lin, Lijuan Wang, and Zicheng Liu. "End-to-End Human Pose and Mesh Reconstruction with Transformers". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2021, pp. 1954–1963. DOI: 10.1109/CVPR46437.2021.00199 (cit. on p. 107).

[106] Hung Yu Ling et al. "Character Controllers using Motion VAEs". In: *ACM Transactions on Graphics* 39.4 (July 2020). DOI: 10.1145/3386569.3392422 (cit. on pp. 40, 49, 59, 61, 134, 154).

[107] Jun Liu et al. "NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10 (May 2019), pp. 2684–2701. DOI: 10.1109/TPAMI.2019.2916873 (cit. on p. 41).

[108] Zhenguang Liu et al. "Towards Natural and Accurate Future Motion Prediction of Humans and Animals". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2019, pp. 9996–10004. DOI: 10.1109/CVPR.2019.01024 (cit. on p. 36).

[109] Suhas Lohit and Rushil Anirudh. "Recovering Trajectories of Unmarked Joints in 3D Human Actions Using Latent Space Optimization". In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE Computer Society, Jan. 2021, pp. 2341–2350. DOI: 10.1109/WACV48630.2021.00239 (cit. on pp. 48, 63, 64).

[110] Matthew Loper et al. "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Transactions on Graphics* 34.6 (Oct. 2015). DOI: 10.1145/2816795.2818013 (cit. on p. 42).

[111] Naureen Mahmood et al. "AMASS: Archive of Motion Capture As Surface Shapes". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society/Computer Vision Foundation (CVF), Oct. 2019, pp. 5441–5450. DOI: 10.1109/ICCV.2019.00554 (cit. on pp. 41, 42, 112, 113).

[112] Alireza Makhzani et al. "Adversarial Autoencoders". Nov. 2015. eprint: 1511.05644 (cit. on pp. 54, 160).

[113] Wei Mao et al. "Learning Trajectory Dependencies for Human Motion Prediction". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society/Computer Vision Foundation (CVF), Oct. 2019, pp. 9488–9496. DOI: 10.1109/ICCV.2019.00958 (cit. on p. 45).

[114] Timo von Marcard et al. "Recovering Accurate 3D Human Pose in The Wild Using IMUs and a Moving Camera". In: *15th European Conference on Computer Vision (ECCV)*. Springer International Publishing, Sept. 2018, pp. 614–631. DOI: 10.1007/978-3-030-01249-6_37 (cit. on pp. 41, 42).

[115] Mathieu Marsot et al. "Correspondence-free online human motion retargeting". Feb. 2023. eprint: 2302.00556 (cit. on p. 100).

[116] Julieta Martinez, Michael J. Black, and Javier Romero. "On human motion prediction using recurrent neural networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, July 2017, pp. 4674–4683. DOI: 10.1109/CVPR.2017.497 (cit. on pp. 47, 134).

[117] Ian Mason, Sebastian Starke, and Taku Komura. "Real-Time Style Modelling of Human Locomotion via Feature-Wise Transformations and Local Motion Phases". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5.1 (May 2022). DOI: 10.1145/3522618 (cit. on pp. 59, 61).

[118] Ian Mason et al. "Few-shot Learning of Homogeneous Human Locomotion Styles". In: *Computer Graphics Forum* 37.7 (Oct. 2018), pp. 143–153. DOI: 10.1111/cgf.13555 (cit. on pp. 36, 40, 59, 61, 65).

[119] Dushyant Mehta et al. "Monocular 3D Human Pose Estimation In The Wild Using Improved CNN Supervision". In: *2017 International Conference on 3D Vision*

(3DV). IEEE Computer Society, Oct. 2017, pp. 506–516. DOI: 10.1109/3dv.2017.00064 (cit. on pp. 86, 112, 113, 116).

[120] Qianhui Men et al. "GAN-based reactive motion synthesis with class-aware discriminators for human–human interaction". In: *Computers & Graphics* 102 (Feb. 2022), pp. 634–645. DOI: 10.1016/j.cag.2021.09.014 (cit. on pp. 43, 50, 54).

[121] Jianyuan Min and Jinxiang Chai. "Motion graphs++: a compact generative model for semantic motion analysis and synthesis". In: *ACM Transactions on Graphics* 31.6 (Nov. 2012). DOI: 10.1145/2366145.2366172 (cit. on pp. 23, 52, 65, 66, 134).

[122] Jianyuan Min, Yen-Lin Chen, and Jinxiang Chai. "Interactive Generation of Human Animation with Deformable Motion Models". In: *ACM Transactions on Graphics* 29.1 (Dec. 2009). DOI: 10.1145/1640443.1640452 (cit. on p. 56).

[123] Pauline Morin et al. "Foot contact detection through pressure insoles for the estimation of external forces and moments: application to running and walking". In: *Computer Methods in Biomechanics and Biomedical Engineering* (June 2021), pp. 1–2. URL: https://hal.inria.fr/hal-03273616/document (cit. on p. 135).

[124] Lucas Mourot et al. "A Survey on Deep Learning for Skeleton-Based Human Animation". In: *Computer Graphics Forum* 41.1 (Nov. 2021). DOI: 10.1111/cgf.14426 (cit. on p. 155).

[125] Lucas Mourot et al. "UnderPressure: Deep Learning for Foot Contact Detection, Ground Reaction Force Estimation and Footskate Cleanup". In: *Computer Graphics Forum* 41.8 (Sept. 2022). DOI: 10.1111/cgf.14635 (cit. on pp. 112, 113).

[126] Marion Mundt et al. "Prediction of ground reaction force and joint moments based on optical motion capture data during gait". In: *Medical Engineering & Physics* 86 (Oct. 2020), pp. 29–34. DOI: 10.1016/j.medengphy.2020.10.001 (cit. on p. 135).

[127] Meinard Müller et al. *Documentation Mocap Database HDM05*. Tech. rep. June 2007. URL: https://resources.mpi-inf.mpg.de/HDM05/07_MuRoClEbKrWe_HDM05.pdf (cit. on pp. 41, 42, 50, 63).

[128] Masaki Nakada et al. "Deep Learning of Biomimetic Sensorimotor Control for Biomechanical Human Animation". In: *ACM Transactions on Graphics* 37.4 (July 2018). DOI: 10.1145/3197517.3201305 (cit. on p. 43).

[129] Kosuke Nakazato, Peter Scheiber, and Erich Müller. "A Comparison of Ground Reaction Forces Determined by Portable Force-Plate and Pressure-Insole Systems in Alpine Skiing". In: *Journal of Sports Science & Medicine* 10.4 (Dec. 2011), pp. 754–762. URL: https://pubmed.ncbi.nlm.nih.gov/24149570/ (cit. on p. 150).

[130] Ferda Ofli et al. "Berkeley MHAD: A Comprehensive Multimodal Human Action Database". In: *2013 IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE Computer Society, Jan. 2013, pp. 53–60. DOI: 10.1109/WACV.2013.6474999 (cit. on p. 42).

[131] Boris N. Oreshkin et al. "Motion In-betweening via Deep Delta-Interpolator". Jan. 2022. eprint: 2201.06701 (cit. on pp. 38, 48, 50, 57, 102).

[132] Soohwan Park et al. "Learning predict-and-simulate policies from unorganized human motion data". In: *ACM Transactions on Graphics* 38.6 (Nov. 2019). DOI: 10.1145/3355089.3356501 (cit. on pp. 65, 134).

[133] Soomin Park, Deok-Kyeong Jang, and Sung-Hee Lee. "Diverse Motion Stylization for Multiple Style Domains via Spatial-Temporal Graph-Based Generative Model". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 4.3 (Sept. 2021). DOI: 10.1145/3480145 (cit. on pp. 63, 71).

[134] Deepak Pathak et al. "Context Encoders: Feature Learning by Inpainting". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2016, pp. 2536–2544. DOI: 10.1109/CVPR.2016.278 (cit. on p. 78).

[135] Dario Pavllo, David Grangier, and Michael Auli. "QuaterNet: A Quaternion-based Recurrent Model for Human Motion". In: *British Machine Vision Conference 2018*. BMVA Press, Sept. 2018. URL: https://dblp.org/rec/conf/bmvc/PavlloGA18 (cit. on pp. 37, 39, 40, 134).

[136] Dario Pavllo et al. "Modeling Human Motion with Quaternion-Based Neural Networks". In: *International Journal of Computer Vision* 128.4 (Apr. 2020), pp. 855–872. DOI: 10.1007/s11263-019-01245-6 (cit. on pp. 37, 39, 40, 134).

[137] Xue Bin Peng and Michiel van de Panne. "Learning Locomotion Skills Using DeepRL: Does the Choice of Action Space Matter?" In: *16th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Association for Computing Machinery (ACM), July 2017. DOI: 10.1145/3099564.3099567 (cit. on p. 134).

[138] Martin Pražák, Ludovic Hoyet, and Carol O'Sullivan. "Perceptual Evaluation of Footskate Cleanup". In: *2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Association for Computing Machinery (ACM), Aug. 2011, 287–294. DOI: 10.1145/2019406.2019444 (cit. on pp. 28, 65, 132, 162).

[139] Jia Qin, Youyi Zheng, and Kun Zhou. "Motion In-Betweening via Two-Stage Transformers". In: *ACM Transactions on Graphics* 41.6 (Dec. 2022). DOI: 10.1145/3550454.3555454 (cit. on pp. 38, 48, 50, 58, 78, 102).

[140] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *International Conference on Learning Representations*. May 2016. eprint: 1511.06434 (cit. on pp. 80, 87).

[141] Aditya Ramesh et al. "Hierarchical Text-Conditional Image Generation with CLIP Latents". Apr. 2022. eprint: 2204.06125 (cit. on p. 166).

[142] Davis Rempe et al. "Contact and Human Dynamics from Monocular Video". In: *16th European Conference on Computer Vision (ECCV)*. Springer International Publishing, Sept. 2020, pp. 71–87. DOI: 10.1007/978-3-030-58558-7_5 (cit. on pp. 66, 134, 136).

[143] Olinde Rodrigues. "Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendants des causes qui peuvent les produire". In: *Journal de Mathématiques Pures et Appliquées* 5 (Dec. 1840), pp. 380–440. URL: https://eudml.org/doc/234443 (cit. on p. 36).

[144] Charles Rose et al. "Efficient Generation of Motion Transitions Using Spacetime Constraints". In: *23rd Annual Conference on Computer Graphics and Interactive Techniques*. Association for Computing Machinery (ACM), July 1996, 147–154. DOI: 10.1145/237170.237229 (cit. on p. 56).

[145] Hossein Rouhani et al. "Ambulatory assessment of 3D ground reaction force using plantar pressure distribution". In: *Gait & Posture* 32.3 (June 2010), pp. 311–316. DOI: 10.1016/j.gaitpost.2010.05.014 (cit. on p. 135).

[146] Chitwan Saharia et al. "Palette: Image-to-Image Diffusion Models". Nov. 2021. eprint: 2111.05826 (cit. on p. 165).

[147] H. Martin Schepers, Matteo Giuberti, and Giovanni Bellusci. *Xsens MVN: Consistent Tracking of Human Motion Using Inertial Sensing*. Tech. rep. P.O. Box 559, 7500 Enschede, The Netherlands: Xsens Technologies B.V., Mar. 2018. URL: https://www.researchgate.net/publication/324007368_Xsens_MVN_Consistent_Tracking_of_Human_Motion_Using_Inertial_Sensing (cit. on pp. 138, 139).

[148] Jesse Scott et al. "From Image to Stability: Learning Dynamics from Human Pose". In: *16th European Conference on Computer Vision (ECCV)*. Springer International Publishing, Sept. 2020, pp. 536–554. DOI: 10.1007/978-3-030-58592-1_32 (cit. on pp. 112, 113, 136, 137).

[149] Amir Shahroudy et al. "NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2016, pp. 1010–1019. DOI: 10.1109/CVPR.2016.115 (cit. on pp. 41, 50, 63).

[150] Mingyi Shi et al. "MotioNet: 3D Human Motion Reconstruction from Monocular Video with Skeleton Consistency". In: *ACM Transactions on Graphics* 40.1 (Sept. 2020). DOI: 10.1145/3407659 (cit. on pp. 40, 48, 63, 66, 134).

[151] Soshi Shimada et al. "Neural Monocular 3D Human Motion Capture with Physical Awareness". In: *ACM Transactions on Graphics* 40.4 (July 2021). DOI: 10.1145/3450626.3459825 (cit. on pp. 136, 140).

[152] Soshi Shimada et al. "PhysCap: Physically Plausible Monocular 3D Motion Capture in Real Time". In: *ACM Transactions on Graphics* 39.6 (Dec. 2020). DOI: 10.1145/3414685.3417877 (cit. on pp. 48, 63, 66, 134, 136).

[153] David I. Shuman et al. "The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains". In: *IEEE Signal Processing Magazine* 30.3 (Apr. 2013), pp. 83–98. DOI: 10.1109/MSP.2012.2235192 (cit. on p. 45).

[154] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. "HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion". In: *International Journal of Computer Vision* 87.1 (Aug. 2009). DOI: 10.1007/s11263-009-0273-6 (cit. on p. 42).

[155] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529 (Jan. 2016), 484–489. DOI: 10.1038/nature16961 (cit. on p. 23).

[156] Harrison Jesse Smith et al. "Efficient Neural Networks for Real-time Motion Style Transfer". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2.2 (July 2019). DOI: 10.1145/3340254 (cit. on pp. 63, 66, 70, 134, 145).

[157] Leslie N. Smith. "Cyclical Learning Rates for Training Neural Networks". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE Computer Society, Mar. 2017, p. 10. DOI: 10.1109/WACV.2017.58 (cit. on p. 118).

[158] Paul Smolensky. *Information processing in dynamical systems: Foundations of harmony theory*. Tech. rep. Feb. 1986, pp. 194–281. URL: https://apps.dtic.mil/sti/citations/ADA620727 (cit. on p. 51).

[159] Jascha Sohl-Dickstein et al. "Deep Unsupervised Learning Using Nonequilibrium Thermodynamics". In: *32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research (PMLR), July 2015, 2256–2265. URL: http://proceedings.mlr.press/v37/sohl-dickstein15.pdf (cit. on p. 165).

[160] Sebastian Starke, Ian Mason, and Taku Komura. "DeepPhase: Periodic Autoencoders for Learning Motion Phase Manifolds". In: *ACM Transactions on Graphics* 41.4 (July 2022). DOI: 10.1145/3528223.3530178 (cit. on pp. 59, 60).

[161] Sebastian Starke et al. "Local Motion Phases for Learning Multi-Contact Character Movements". In: *ACM Transactions on Graphics* 39.4 (July 2020). DOI: 10.1145/3386569.3392450 (cit. on pp. 40, 49, 59–62, 65, 134, 154, 160).

[162] Sebastian Starke et al. "Neural Animation Layering for Synthesizing Martial Arts Movements". In: *ACM Transactions on Graphics* 40.4 (Aug. 2021). DOI: 10.1145/3450626.3459881 (cit. on pp. 59, 61, 62).

[163] Sebastian Starke et al. "Neural State Machine for Character-Scene Interactions". In: *ACM Transactions on Graphics* 38.6 (Nov. 2019). DOI: 10.1145/3355089.3356505 (cit. on pp. 40, 49, 59, 60, 65, 66, 134, 154).

[164] Ke Sun et al. "Deep High-Resolution Representation Learning for Human Pose Estimation". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2019, pp. 5686–5696. DOI: 10.1109/CVPR.2019.00584 (cit. on p. 76).

[165] Xiangjun Tang et al. "Real-Time Controllable Motion Transition for Characters". In: *ACM Transactions on Graphics* 41.4 (July 2022). DOI: 10.1145/3528223.3530090 (cit. on pp. 50, 57).

[166] Yongyi Tang et al. "Long-Term Human Motion Prediction by Modeling Motion Context and Enhancing Motion Dynamics". In: *Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence (IJCAI), July 2018, pp. 935–941. DOI: 10.24963/ijcai.2018/130 (cit. on p. 47).

[167] Graham W. Taylor and Geoffrey E. Hinton. "Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style". In: *26th International Conference on Machine Learning*. Association for Computing Machinery (ACM), June 2009, pp. 1025–1032. DOI: 10.1145/1553374.1553505 (cit. on pp. 36, 51).

[168] Graham W. Taylor, Geoffrey E. Hinton, and Sam T. Roweis. "Modeling Human Motion Using Binary Latent Variables". In: *19th International Conference on Neural Information Processing Systems*. The MIT Press, Sept. 2006, pp. 1345–1352. DOI: 10.7551/mitpress/7503.003.0173 (cit. on pp. 36, 51).

[169] Sam Toyer et al. "Human Pose Forecasting via Deep Markov Models". In: *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE Computer Society, Dec. 2017, pp. 1–8. DOI: 10.1109/DICTA.2017.8227441 (cit. on pp. 50, 52).

[170] Carnegie-Mellon University. *CMU Graphics Lab Motion Capture Database*. Accessed: 2021-09-16. 2003. URL: http://mocap.cs.cmu.edu (cit. on pp. 41, 42, 50, 59, 63).

[171] Simon Fraser University. *SFU Motion Capture Database*. Accessed: 2021-09-16. 2019. URL: https://mocap.cs.sfu.ca (cit. on p. 42).

[172] Ashish Vaswani et al. "Attention is All You Need". In: *31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2017, 6000–6010. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cit. on pp. 47, 111).

[173] Léon Victor, Alexandre Meyer, and Saïda Bouakaz. "Learning-based pose edition for efficient and interactive design". In: *Computer Animation and Virtual Worlds* 32.3-4 (June 2021), e2013. DOI: 10.1002/cav.2013 (cit. on p. 154).

[174] Léon Victor, Alexandre Meyer, and Saïda Bouakaz. "Pose Metrics: A New Paradigm for Character Motion Edition". Jan. 2023. eprint: 2301.06514 (cit. on p. 71).

[175] Ruben Villegas et al. "Contact-Aware Retargeting of Skinned Motion". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society/Computer Vision Foundation (CVF), Oct. 2021, pp. 9700–9709. URL: https://openaccess.thecvf.com/content/ICCV2021/papers/Villegas_Contact-Aware_Retargeting_of_Skinned_Motion_ICCV_2021_paper.pdf (cit. on p. 100).

[176] Ruben Villegas et al. "Neural Kinematic Networks for Unsupervised Motion Retargetting". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2018, pp. 8639–8648. DOI: 10.1109/CVPR.2018.00901 (cit. on pp. 46, 63, 67, 100, 118, 122).

[177] Borui Wang et al. "Imitation Learning for Human Pose Prediction". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society/Computer Vision Foundation (CVF), Oct. 2019, pp. 7124–7133. DOI: 10.1109/ICCV.2019.00722 (cit. on p. 47).

[178] He Wang et al. "Spatio-temporal Manifold Learning for Human Motions via Long-horizon Modeling". In: *IEEE Transactions on Visualization and Computer Graphics* 27.1 (Aug. 2019). DOI: 10.1109/TVCG.2019.2936810 (cit. on pp. 44, 63, 64, 134).

[179] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. "Gaussian Process Dynamical Models for Human Motion". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (Dec. 2008), pp. 283–298. DOI: 10.1109/TPAMI.2007.1167 (cit. on p. 56).

[180] Qi Wang et al. "Adversarial learning for modeling human motion". In: *The Visual Computer* 36 (June 2020), pp. 141–160. DOI: 10.1007/s00371-018-1594-7 (cit. on pp. 46, 50, 54, 63, 70).

[181] Qi Wang et al. "Transferring Style in Motion Capture Sequences with Adversarial Learning". In: *26th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. European Symposium on Artificial Neural Networks (ESANN), Apr. 2018. URL: https://hal.archives-ouvertes.fr/hal-02100672/ (cit. on pp. 46, 63, 70).

[182] Yingying Wang and Michael Neff. "Deep Signatures for Indexing and Retrieval in Large Motion Databases". In: *8th ACM SIGGRAPH Conference on Motion in Games*. Association for Computing Machinery (ACM), Nov. 2015, pp. 37–45. DOI: 10.1145/2822013.2822024 (cit. on pp. 43, 63, 64).

[183] Yumeng Wang, Wujun Che, and Bo Xu. "Encoder–decoder recurrent network model for interactive character animation generation". In: *The Visual Computer* 33 (June 2017), pp. 971–980. DOI: 10.1007/s00371-017-1378-5 (cit. on pp. 46, 59, 62).

[184] Zhenyi Wang et al. "Learning Diverse Stochastic Human-Action Generators by Learning Smooth Latent Transitions". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.7 (Apr. 2020), pp. 12281–12288. DOI: 10.1609/aaai.v34i07.6911 (cit. on pp. 46, 50, 54).

[185] Zhiyong Wang, Jinxiang Chai, and Shihong Xia. "Combining Recurrent Neural Networks and Adversarial Training for Human Motion Synthesis and Control". In: *IEEE Transactions on Visualization and Computer Graphics* 27.1 (Sept. 2019), pp. 14–28. DOI: 10.1109/TVCG.2019.2938520 (cit. on pp. 46, 50, 54, 63–65, 134).

[186] Gail Weiss, Yoav Goldberg, and Eran Yahav. "On the Practical Computational Power of Finite Precision RNNs for Language Recognition". In: *56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, July 2018, pp. 740–745. DOI: 10.18653/v1/P18-2117 (cit. on p. 47).

[187] Andrew Witkin and Michael Kass. "Spacetime Constraints". In: *15th Annual Conference on Computer Graphics and Interactive Techniques*. Association for Computing Machinery (ACM), Aug. 1988, 159–168. DOI: 10.1145/54852.378507 (cit. on p. 56).

[188] Andrew Witkin and Zoran Popović. "Motion Warping". In: *22nd International ACM Conference on Computer Graphics and Interactive Techniques*. Association for Computing Machinery (ACM), Mar. 1995, pp. 105–108. DOI: 10.1145/218380.218422 (cit. on p. 23).

[189] Jungdam Won and Jehee Lee. "Learning Body Shape Variation in Physics-based Characters". In: *ACM Transactions on Graphics* 38.6 (Nov. 2019). DOI: 10.1145/3355089.3356499 (cit. on p. 134).

[190] Shihong Xia et al. "Realtime Style Transfer for Unlabeled Heterogeneous Human Motion". In: *ACM Transactions on Graphics* 34.4 (July 2015). DOI: 10.1145/2766999 (cit. on pp. 42, 134).

[191] Zhaoming Xie et al. "ALLSTEPS: Curriculum-driven Learning of Stepping Stone Skills". In: *Computer Graphics Forum* 39.8 (Oct. 2020). DOI: 10.1111/cgf.14115 (cit. on p. 134).

[192] Yuliang Xiu et al. "Pose Flow: Efficient Online Pose Tracking". In: *British Machine Vision Conference 2018*. BMVA Press, Sept. 2018. URL: https://dblp.org/rec/conf/bmvc/XiuLWFL18.html (cit. on pp. 76, 88, 91, 92).

[193] Jingwei Xu et al. "Hierarchical Style-based Networks for Motion Synthesis". In: *16th European Conference on Computer Vision (ECCV)*. Springer International Publishing, Sept. 2020, pp. 178–194. DOI: 10.1007/978-3-030-58621-8_11 (cit. on pp. 47, 50, 56).

[194] Sijie Yan et al. "Convolutional Sequence Generation for Skeleton-Based Action Synthesis". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society/Computer Vision Foundation (CVF), Oct. 2019, pp. 4393–4401. DOI: 10.1109/ICCV.2019.00449 (cit. on pp. 50, 54, 160).

[195] Xinchen Yan et al. "MT-VAE: Learning Motion Transformations to Generate Multimodal Human Dynamics". In: *15th European Conference on Computer Vision (ECCV)*. Springer International Publishing, Sept. 2018, pp. 276–293. DOI: 10.1007/978-3-030-01228-1_17 (cit. on pp. 50, 52, 160).

[196] Dongseok Yang, Doyeon Kim, and Sung-Hee Lee. "LoBSTr: Real-time Lower-body Pose Prediction from Sparse Upper-body Tracking Signals". In: *Computer Graphics Forum* 40.2 (June 2021), pp. 265–275. DOI: 10.1111/cgf.142631 (cit. on pp. 63, 66).

[197] Ling Yang et al. "Diffusion Models: A Comprehensive Survey of Methods and Applications". Sept. 2022. eprint: 2209.00796 (cit. on p. 165).

[198] Raymond A. Yeh et al. "Semantic Image Inpainting with Deep Generative Models". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*

(CVPR). IEEE Computer Society, July 2017, pp. 6882–6890. DOI: 10.1109/CVPR.2017.728 (cit. on pp. 78, 85).

[199] Jiahui Yu et al. "Generative Image Inpainting with Contextual Attention". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2018, pp. 5505–5514. DOI: 10.1109/CVPR.2018.00577 (cit. on p. 78).

[200] Moonwon Yu et al. "Fast Terrain-Adaptive Motion Generation using Deep Neural Networks". In: *SIGGRAPH Asia 2019 Technical Briefs*. Association for Computing Machinery (ACM), Mar. 2019, pp. 57–60. DOI: 10.1145/3355088.3365157 (cit. on pp. 50, 56).

[201] Wenhao Yu, Greg Turk, and C. Karen Liu. "Learning Symmetric and Low-Energy Locomotion". In: *ACM Transactions on Graphics* 37.4 (July 2018). DOI: 10.1145/3197517.3201397 (cit. on p. 134).

[202] Ye Yuan and Kris Kitani. "DLow: Diversifying Latent Flows for Diverse Human Motion Prediction". In: *16th European Conference on Computer Vision (ECCV)*. Springer International Publishing, Sept. 2020, pp. 346–364. DOI: 10.1007/978-3-030-58545-7_20 (cit. on p. 46).

[203] Chuanqi Zang, Mingtao Pei, and Yu Kong. "Few-shot Human Motion Prediction via Learning Novel Motion Dynamics". In: *Twenty-Ninth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence (IJCAI), Sept. 2020, pp. 846–852. DOI: 10.24963/ijcai.2020/118 (cit. on p. 44).

[204] Petrissa Zell, Bodo Rosenhahn, and Bastian Wandt. "Weakly-supervised Learning of Human Dynamics". In: *16th European Conference on Computer Vision (ECCV)*. Springer International Publishing, Sept. 2020, pp. 68–84. DOI: 10.1007/978-3-030-58574-7_5 (cit. on p. 136).

[205] Petrissa Zell, Bastian Wandt, and Bodo Rosenhahn. "Joint 3D Human Motion Capture and Physical Analysis from Monocular Videos". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE Computer Society, July 2017, pp. 17–26. DOI: 10.1109/CVPRW.2017.9 (cit. on p. 135).

[206] Feng Zhang et al. “Distribution-Aware Coordinate Representation for Human Pose Estimation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2020, pp. 7093–7102. DOI: 10.1109/CVPR42600.2020.00712 (cit. on p. 76).

[207] He Zhang et al. “Mode-Adaptive Neural Networks for Quadruped Motion Control”. In: *ACM Transactions on Graphics* 37.4 (July 2018). DOI: 10.1145/3197517.3201366 (cit. on pp. 40, 49, 59, 60, 134, 154, 160).

[208] Peizhao Zhang et al. “Leveraging Depth Cameras and Wearable Pressure Sensors for Full-Body Kinematics and Dynamics Capture”. In: *ACM Transactions on Graphics* 33.6 (Nov. 2014). DOI: 10.1145/2661229.2661286 (cit. on p. 135).

[209] Yi Zhou et al. “On the Continuity of Rotation Representations in Neural Networks”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society/Computer Vision Foundation (CVF), June 2019, pp. 5738–5746. DOI: 10.1109/CVPR.2019.00589 (cit. on pp. 37, 38).

[210] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Oct. 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244 (cit. on p. 71).

[211] Yuliang Zou et al. “Reducing Footskate in Human Motion Reconstruction with Ground Contact Constraints”. In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE Computer Society, Mar. 2020, pp. 448–457. DOI: 10.1109/WACV45572.2020.9093329 (cit. on pp. 48, 63, 66, 134).

Université de Rennes

TITRE : APPRENTISSAGE PROFOND POUR L'ANIMATION DU SQUELETTE DE PERSONNAGES VIRTUELS: ÉDITION DE LA TOPOLOGIE, SQUELETTO-MORPHOSE ET NETTOYAGE

Mot clés : animation, personnages virtuels, apprentissage profond, mouvement humain

Résumé : L'apprentissage profond a révolutionné l'animation de personnages durant la dernière décennie. Des modèles novateurs et sophistiqués ont permis d'obtenir un réalisme sans précédent. Cependant, ces avancées ne permettent pas encore de remplacer les animateurs sur toutes les tâches fastidieuses et peu créatives. L'objectif de cette thèse est de s'attaquer aux obstacles qui les en empêchent. En particulier, nous avons abordé le manque de données de mouvement de qualité et la tendance des réseaux de neurones à introduire des artéfacts lors du traitement de données de mouvement. Nous avons d'abord exploré l'amélioration de séquences de poses humaines 2D estimées à partir de vidéos en utilisant des connaissances a priori apprises par un modèle génératif profond. Ensuite, nous avons abstrait le mouvement de la topologie et de la morphologie dans une représentation profonde, afin de rassembler, de traiter ou de squeletto-morphoser des séquences de mouvement avec des topologies et des morphologies variables. Enfin, nous nous sommes attaqués à la détection des contacts des pieds avec le sol dans le but de nettoyer automatiquement les artéfacts de glissement des pieds.

TITLE: DEEP LEARNING FOR SKELETAL CHARACTER ANIMATION: TOPOLOGY EDITING, RETARGETING AND CLEANING

Keywords: character animation, deep learning, human motion

Abstract: Deep learning has revolutionised skeletal character animation in the last decade. Novel frameworks and sophisticated schemes pushed toward to an unprecedented realism. Though, these advances still lack quality, flexibility and ability to generalise to replace animators when dealing with tedious low-creativity tasks. Therefore, the purpose of this thesis is to tackle current obstacles preventing from doing so. In particular, we strived to alleviate the lack of high-quality motion data in the context of deep learning and the inclination of neural networks to introduce artefacts when processing motion data. To this end, we first investigated the enhancement of 2D human pose sequences estimated from video using prior knowledge captured in a deep generative model. Then, we abstracted out motion features from skeleton topology and morphology in a deep motion representation, with the goal of gathering, processing or retargeting motion sequences with variable skeleton topologies and morphologies. Finally, we tackled foot contact detection to automatically clean up footskate artefacts, well-known in character animation.