Learning-based interactive character animation

Léon Victor

N° d'ordre NNT : 2023ISAL0028

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

opérée au sein de

l'Institut National des Sciences Appliquées de Lyon

École Doctorale EDA 512
Mathématiques et Informatique de Lyon

Spécialité de doctorat : Informatique

Soutenue publiquement le 07/04/2023, par :

Léon Victor

Learning-based Interactive Character Animation Edition

Devant le jury composé de :

M. Jean-Philippe Vandeborre	Professeur	Institut Mines Télécom Nord Europe	Rapporteur
M. Renaud Séguier	Professeur	Centrale-Supelec	Rapporteur
MMe Sylvie Gibet	Professeure	Université Bretagne Sud	Examinatrice
M. Sébastien Mavromatis	Professeur	Université Aix Marseille	Examinateur
M. Bertrand Kerautret	Professeur	Université Lumière Lyon 2	Examinateur
MMe Véronique Eglin	Professeure	INSA Lyon	Directrice de thèse
MMe Saïda Bouakaz	Professeure	Université Lyon 1	Co-Directrice de thèse
M. Alexandre Meyer	Maître de Conférence	Université Lyon 1	Co-Directeur de thèse

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Département de la Formation par la Recherche et des Études Doctorales (**FEDORA**)

Bâtiment INSA direction, 1er étage
37, av. J. Capelle
69621 Villeurbanne Cédex
fedora@insa-lyon.fr

Référence : TH0955_Léon VICTOR

L'INSA Lyon a mis en place une procédure de contrôle systématique via un outil de détection de similitudes (logiciel Compilatio). Après le dépôt du manuscrit de thèse, celui-ci est analysé par l'outil. Pour tout taux de similarité supérieur à 10%, le manuscrit est vérifié par l'équipe de FEDORA. Il s'agit notamment d'exclure les auto-citations, à condition qu'elles soient correctement référencées avec citation expresse dans le manuscrit.

Par ce document, il est attesté que ce manuscrit, dans la forme communiquée par la personne doctorante à l'INSA Lyon, satisfait aux exigences de l'Etablissement concernant le taux maximal de similitude admissible.

Département FEDORA – INSA Lyon - Ecoles Doctorales

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **https://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr	**M. Stéphane DANIELE** C2P2-CPE LYON-UMR 5265 Bâtiment F308, BP 2077 43 Boulevard du 11 novembre 1918 69616 Villeurbanne **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **https://edeea.universite-lyon.fr** Sec. : Stéphanie CAUVIN Bâtiment Direction INSA Lyon Tél : 04.72.43.71.70 secretariat.edeea@insa-lyon.fr	**M. Philippe DELACHARTRE** INSA LYON Laboratoire CREATIS Bâtiment Blaise Pascal, 7 avenue Jean Capelle 69621 Villeurbanne CEDEX Tél : 04.72.43.88.63 **philippe.delachartre@insa-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.e2m2@univ-lyon1.fr	**Mme Sandrine CHARLES** Université Claude Bernard Lyon 1 UFR Biosciences Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69622 Villeurbanne CEDEX **sandrine.charles@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://ediss.universite-lyon.fr** Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.ediss@univ-lyon1.fr	**Mme Sylvie RICARD-BLUM** Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Raulin - 2ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tél : +33(0)4 72 44 82 32 **sylvie.ricard-blum@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	**M. Hamamache KHEDDOUCI** Université Claude Bernard Lyon 1 Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tél : 04.72.44.83.69 **hamamache.kheddouci@univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Yann DE ORDENANA Tél : 04.72.18.62.44 yann.de-ordenana@ec-lyon.fr	**M. Stéphane BENAYOUN** Ecole Centrale de Lyon Laboratoire LTDS 36 avenue Guy de Collongue 69134 Ecully CEDEX Tél : 04.72.18.64.37 **stephane.benayoun@ec-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bâtiment Direction INSA Lyon mega@insa-lyon.fr	**M. Jocelyn BONJOUR** INSA Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **https://edsciencessociales.universite-lyon.fr** Sec. : Mélina FAVETON INSA : J.Y. TOUSSAINT Tél : 04.78.69.77.79 melina.faveton@univ-lyon2.fr	**M. Christian MONTES** Université Lumière Lyon 2 86 Rue Pasteur 69365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Abstract

Animated characters are a crucial component of compelling and lively virtual worlds. Their motion is influenced by complex behaviors, and the production of expressive character animation heavily relies upon animators' skills, training and knowledge. Ever since computers have been used in this creative process, researchers have strived to develop intuitive, interactive and precise tools and interfaces to facilitate animators' work. In the last decade, deep learning methods have raised a lot of interest in the computer animation community, capitalizing on the increasing availability of real-life captured motion data to train generative models able to synthesize new movement. While the studied application allow for the generation of new realistic animation, the process unfortunately leaves little room for animators' control. This thesis therefore proposes to use neural networks to learn the subtle complexities carried by motion data and to use the extracted information in the design of user-centric animation tools which fit in the existing animation production workflow. We propose two approaches allowing a user to edit a character's pose at a given time frame, first by directly manipulating its skeleton, and second through higher-level pose parameters.

Contents

CHAPTER 1

Introduction

1.1 Context

Animated virtual characters are present everywhere in video media such as films, animated movies or video games. They can often be found in the spotlight, telling stories or embodying players, but also in the background, populating vivid worlds where they strengthen the spectators' immersion. The way these characters move, act around their world and interact with each other is an important point of focus for their creators, and producing high quality, expressive and diverse animation is a crucial aspect of the production process.

Humans have been interested in depicting moving scenes for as long as they have been drawing. Early animation thus has centuries of history, starting with side-by-side wall paintings, to puppets and shadow play, and up to the photorealistic digital humans of today. The modern understanding of the discipline arose recently, following the democratization of the cinematic industry during the late 19th century. The beginning of the 20th century saw the development of the traditional animation technique in which animation artists draw characters by hand, image per image. This period also marks the first public success of animated production, first through short-format cartoons, quickly followed by feature-length films. Nearing the end of the century, computers made their way in the animation creative process. In their first appearances digital tools were used to reduce the burden of repetitive work and improve production times, but still operated on flat 2D images. The focus however quickly shifted to the 3D world, driven by the fast growth of computing power and progress in computer graphics. Nowadays, the most common approach to animation production is entirely computer-based, and is the result of virtual cameras filming virtual scenes in which 3D models act, even in cases where the final image aims for a 2D look. With the rise of 3D video games, the growing scope of animation feature films, and the looming prospect

of high-quality immersive experiences in virtual reality, animation is in ever-increasing demand.

The human eye perceives a lot of information through motion: it is a complex behavior through which characters convey intent, emotions and personality. A same gesture can be performed differently depending on the unique identity of the performer, and a character can achieve the same action in a variety of ways. Creating high-quality motion that accurately depicts a character's actions *and* convey its personality and internal state is a difficult task, and animators undergo long and meticulous training to master their craft. The role of computer animation researchers is then to imagine and develop new tools and interfaces to help artists transform their imagination into tangible art, and bring their characters to life.

1.2 Motivation

In this thesis, we are interested in proposing a new set of such tools. We are also interested in the information present in existing motion data, and in ways it can be extracted then exploited in the design of the tools. Our approach thus falls in the "data-oriented" category of animation research, which has been a topic for the community since its very beginning.

During the last decade, neural networks and more specifically deep learning methods have seen an explosion in popularity as the computational power and amount of data they require have become accessible to larger audiences. They have proven to excel at learning to model complex data distributions from database of existing samples, and at approximating complicated functions. Their usage have quickly grown out of their original image classification root: they are now studied in application to all sorts of data types [Qin+20; Del+21], and towards various goals, up to synthesizing new unseen data from the modelled distribution. The animation field has not been spared, and, starting from 2015, a larger and larger amount of research has been dedicated to applying neural networks to motion data. However, as shown in the following literature review, the vast majority of existing work in this direction has focused on animation synthesis, i.e. the task of generating new motion, occasionally guiding the generation through a high level control signal such as an expected locomotion trajectory or a past context of existing motion. While these methods produce impressive results for on-the-fly synthesis, they are limited when it comes to artistic control. Animators creating new motion need precise control over their tools' output, so that they can author an animation that precisely fit their expectation for it. With existing methods, if the generated motion is not satisfying, the only solution is to retrain the whole models with a different dataset.

This observation motivates the objective of this thesis: exploiting the modelling power of deep learning methods to extract meaningful information from animation data, and using it to power tools which facilitate the artists' job while leaving them in control of the final output. We therefore propose methods that apply the insight learned from data to more fine-grained operations and fit in the existing animation pipeline. To achieve our goal, the proposed tools focus on being unintrusive, controllable, easy to use, and fast enough to allow interactive use.

1.3 Thesis contributions & organization

The work presented in this thesis revolves around the idea of leveraging existing data in the design of atomic operations in the animation workflow. We select and focus on the task of skeleton manipulation, during which the user edits a character's pose at a specific time stamp. This choice is made considering that it is an essential step in the process that, in its usual format, heavily relies on the animator's understanding of complex skeleton constraints. We present a new pipeline in which said constraints are learned and handled by a few neural networks, while the users can focus on editing a character's pose through simple and efficient manipulators.

Our contributions constitute the building blocks of this pipeline, which are presented throughout this thesis.

The first chapter gives an overview of the literature in computer animation, presenting the fundamental concepts, the efforts that lead to the current workflows used to produce animation, and the most recent experiments aiming to reduce the burden of animators, including animation data modelling with, among other methods, neural networks. Through this presentation it also illustrates the shortcomings of existing methods that our work attempts to resolve.

The second chapter presents the general idea of this thesis, which is to use neural networks to learn an alternative (latent) representation space for animation frame data, then edit latent pose representation, leaving the handling of complex skeleton constraints to the learned mapping functions. We present the dataset used to train our neural networks and discuss two approaches to building the latent space.

A first application of the space is then presented in the fourth chapter. We focus on the task of pose editing through the manipulation of some of its joints, similarly to a virtual puppet. Two methods are presented and discussed.

Chapter five investigates the animation style metaphor and its applications in the thesis' paradigm. We present a pose editing method in which the pose is modified following some high-level but objective pose parameters, using a learned editing function.

Finally, the fourth chapter concludes this thesis by summarizing our findings, discussing them and suggesting further improvements.

CHAPTER 2

Literature review

2.1 Introduction

Ever since the early days of computers, animators and researchers have looked for how they could be used to improve the animation process, which results in a large body of literature. The goal of this thesis being to explore how neural networks can be used to provide smart animation tools, it is a direct continuation of this past effort. This chapter gives an overview of the previous work our research builds upon.

Nowadays two dominant methods to produce animation coexist. Key-framed animation is the direct successor of traditional pen-and-paper animation: animators manipulate a character, setting its poses at selected time frames. The motion is created by playing these key-frames in sequence, interpolating between them if necessary. Motion Capture (MoCap) techniques instead focus on recording real-life actors moving, then applying the captured motion to virtual characters. The work presented in this thesis makes use of both methods: while we aim to develop tools that fit in the key-frame animation pipeline, we take advantage of the naturalness and relative accessibility of MoCap data to train neural networks. While our work depends on the existence of faithful motion data, the capture process itself is a broad topic, and is kept out of this report for shortness sake[1]. Similarly, the animation of character faces, expressions and speech motion are specific and widely studied subjects[SS18; HP22], out of the scope of this review. We instead focus on the different steps used to produce key-frame animation, to explain the context and the limitations we aim to alleviate.

The first section 2.2 gives an overview of what "animation data" entails, and how it is applied to animate character models. Then in 2.3 we discuss the different existing methods used to design an animation through the manipulation of key frames. Section 2.4 we present the efforts towards reusing existing animation data, leading to recent work dedicated to modelling it through deep learning methods. Finally, in section 2.6, we describe the different proposed approaches to edit an animation through the metaphor of its "style".

[1]For overviews of the state of the art in MoCap technologies one can refer to Zhu *et al.* [ZL16] or Desmarais *et al.* [Des+21] on real-time-capture and markerless approaches, respectively.

2.2 Skeletal animation

In computer graphics, virtual characters are represented by three-dimensional polygon meshes on which texture images and physically-based materials are applied. Their motion is driven by an underlying hierarchical structure of joints (or bones) which collectively form the skeleton (Fig. 2.1). When the skeleton moves, it drives the deformation of the mesh by applying its bones' transformations to the vertices, in a process called skinning [CHP89]. The influence of a given bone on a given vertex is specified by a weight factor that artists can tweak to achieve the desired deformation behavior. The modeling of the character and the skinning process are out of the scope of this thesis, as we are only interested in the skeleton and its motion.

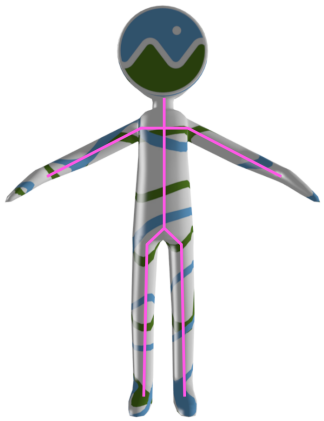


FIGURE 2.1: A textured 3D mesh with its underlying animation skeleton (in pink).

Animation data consists of a time series of skeleton configurations (key poses) at specific time stamps (key frames). A pose is usually defined by the transformation (translation, orientation, and scale) of each bone in the skeleton. At runtime, the computer interpolates between these poses to display a smooth animation (Fig. 2.2) [BW71]. Within that framework the role of an animator is then to design the key frames to produce the character's motion.

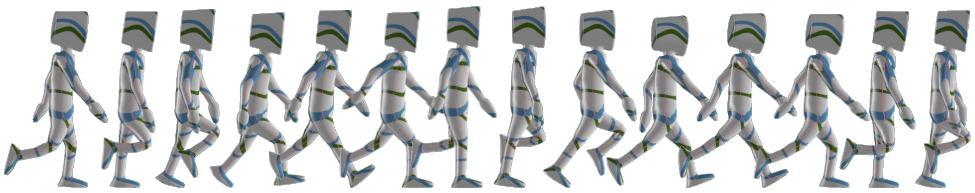


FIGURE 2.2: Interpolating between key frames creates a new pose for each frame, creating the illusion of a smooth animation.

This skeleton representation has multiple advantages: It is mainly a memory-efficient representation for animation data, as it stores only per-frame bones transformation and fixed vertices bone weights, instead of per-frame per-vertex transformations for mesh animation. Additionally, its simplicity makes it an effective interactive interface for animators, which can manipulate it much like a puppet to achieve desired poses [BW76].

2.3 Key-frame editing

The common key-frame animation synthesis pipeline can be split in two phases: posing and interpolation control. During the first, animators manipulate the skeleton to define key poses at specified key frames. In the second, they control the various processes used to fill the gaps between key-frames.

2.3.1 Pose manipulation

Virtual characters' skeletons can be highly complex: they are composed of many joints, each of which can have specific restrictions, such as orientation limitations. They must also respect many constraints specific to the character they represent. For instance, the distance between each pair of joint must remain constant; or can conditionally allow for a limited amount of stretch, and the limbs must never overlap when accounting for the morphology of the character. Manually parameterizing each joint of a skeleton to respect these constraints is next to impossible, even more so when multiple poses are aligned and must also respect temporal coherence.

A substantial part of the literature is then dedicated to describing methods which facilitate the manipulation of the skeleton. These tools are in direct contact with animators, so their focus is on ease-of-use as well as real-time interactivity. We review the different approaches to this kind of interfaces, starting with forward and inverse kinematics methods, to sketching interfaces, data-driven methods, and finally interfaces in three dimensions.

Forward and inverse kinematics

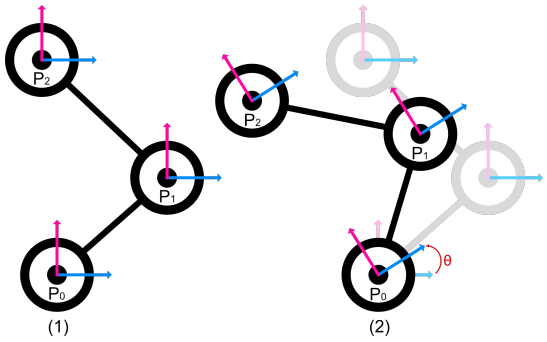


FIGURE 2.3: Forward Kinematics: A modification to the orientation of the joint P_0 at the starting position (1) is propagated down the chain, modifying the positions of both P_1 and P_2 in the resulting position (2).

The most straightforward skeleton manipulation tool is Forward Kinematics (FK): the parameterization of one joint in the chain is the result of combining those of all the joints up the chain. This way the modification of a joint's position or orientation is echoed down the chain to the end effector (see Fig. 2.3). Using this method, an animator can change a joint's orientation to edit the pose, ensuring the constant length of each bone.

The opposite technique, Inverse Kinematics (IK), finds a suitable parameterization of the chain given a target parameterization of the end-effector. This can be used as an effective interface for skeleton manipulation where animators can drag a joint over a desired position or orientation while pinning some others in place (*pin-and-drag*

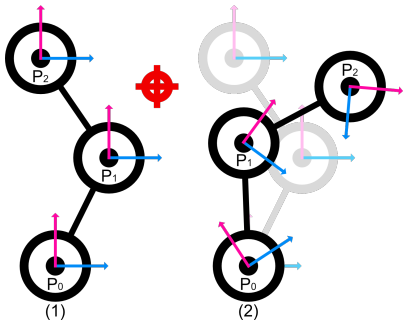


FIGURE 2.4: Inverse Kinematics: Given a target position for the end effector P_2 the IK method must find a proper parameterization of both P_0 and P_1. Multiple solutions exist.

interface [YN03]). The IK problem is an ill-posed one, as a given target configuration may have one, multiple, or no solution. A recent review of IK techniques in computer graphics [Ari+18] proposes to split the existing solutions in four families (Analytic, Numerical, Data-driven[2], and Hybrid) depending on their characteristics, analyzing their strong and weak points. The following section presents a restricted summary of their review focusing on the methods that are best suited for real-time IK applied to character skeletons.

Analytical solutions attempt to find a formulation of the problem which yields an exact solution given a target configuration [RR93; Kal08]. They have the advantage of being reliable and of having global solutions, but they can be computationally costly. Finding an analytical solution is also be next to impossible for complex, strongly-constrained systems, such as a character's skeleton. They therefore don't see much use in pose edition, but are good candidates for robotics.

Numerical solutions formulate a cost function for the problem and iteratively minimize it. They can be divided in three categories: Netwon, Heuristic, Jacobian. Newton solutions use the quasi-Newton method to approximate the objective function [NW99], but much like analytic solutions, they are too complex and computationally costly for the use-case at hand. Jacobian methods use the Jacobian of the system (a matrix of the partial derivatives of each joint's configuration with respect to their angles), as a linear approximation to the problem [Bus09]. The use of an approximation allows for lighter computations and faster inference, which made Jacobian solutions popular during the 1980s and 1990s. In the recent years they have however fallen out of favor due to the demanding environment of real-time interactive software. Heuristic methods do not try to generalize the IK problem and instead rely on simple operations that they can repeat iteratively to reach an acceptable solution. This process allows them to be extremely computationally efficient, at the cost of having no consideration for the naturalness of the produced pose. This drawback can be attenuated by extending the methods to respect more constraints, which is fortunately easy to do in most of them. Their speed make them the currently preferred solution in animation software, so we next describe in deeper details two of the most popular heuristic IK algorithms: Cyclic Coordinate Descent (CCD) and Forward and Backward Reaching Inverse Kinematics (FABRIK).

[2]Data-driven IK solutions are closely related to full-body posing techniques, so both have been grouped and are discussed in their own section (2.3.1) below.

The CCD algorithm [LY84; WC91] operates iteratively to minimize the distance and rotation error between the end effector and the target. At each step, the end-effector is rotated to align with the target. Then, each joint down to the root is rotated to minimize the error. The process is repeated until the end-effector reaches the target. The success of CCD can be attributed to its simplicity: it is easy to implement and has low computational cost. The algorithm has also received multiple improvements, allowing it to handle different kind of constraints [Wel93] or multi-chains problems [Shi+01; KM05; KMA05].

FABRIK [AL11] is an iterative solver that gradually modifies the positions of the chain's joints to minimize the distance between the target and the end effector, operating in two stages (forward and backward). In the forward stage, the end-effector is placed on the target. The other joints are updated one by one down to the root, and placed on the line between their current position and their child's new one, respecting the distance between them. In the backward stage, the same process is followed, only the other way around: the root is placed back on its original positions, and the joints *up* the chain are "pulled back" one by one to it. The two phases are repeated until the end effector is close enough to the target. The advantages of FABRIK are its simplicity and flexibility: the algorithm has been proven to always converge when the target is placed within reach [ACL16], and has been extended to handle many kinds of constraints [MC13; ACL16; TY17].

Hybrid solutions usually divide the IK task in sub-problems, and solve each of them with a specific method. For example, some approaches reduce the search space of a numerical solution by using a preliminary analytical step [LS99]. Others use data-driven methods for the same purpose: Agrawal *et al.* [Av16] learn a prior which is used to warm-start FABRIK, yielding more natural looking human poses.

Forward and inverse kinematics are at the core of most modern animation software and lead to precise results, but can be tedious to use from an artist's perspective. Other techniques and interfaces have then been studied to alleviate this problem.

Sketching interfaces

(A) Stick-figures [Dav+06] (B) Line of action [GCR13] (C) Generalized [Hah+15]

FIGURE 2.5: Sketch-based posing interfaces

In traditional pen-and-paper animation, it is common for animators to start posing by sketching a simplified version of the character to quickly get an idea of what their mental picture of the pose will look like. The draft is then progressively refined

until reaching a satisfying character pose and acceptable level of details. Computer animators use the same coarse-to-fine workflow [Las01], but the common posing tools are not as efficient as a quick sketch.

A body of work has then taken interest in translating this workflow to computer animation (some of the literature's methods are illustrated in Fig. 2.5). The main difficulty in adapting the sketch abstraction is the reconstruction of a full 3D model from a flat 2D drawing: the additional depth dimension cannot be precisely inferred, which is a source of ambiguity.

Early work along this idea required users to draw stick figures, which are commonly used as rough depictions of characters, and have the advantage of corresponding to the widely used skeleton model. The stick-figures could either be sketched from scratch or over an existing doodle, and either on paper [TBv04; CON05] or directly on the computer [Fek+95]. They are then used to generate a 3D pose.

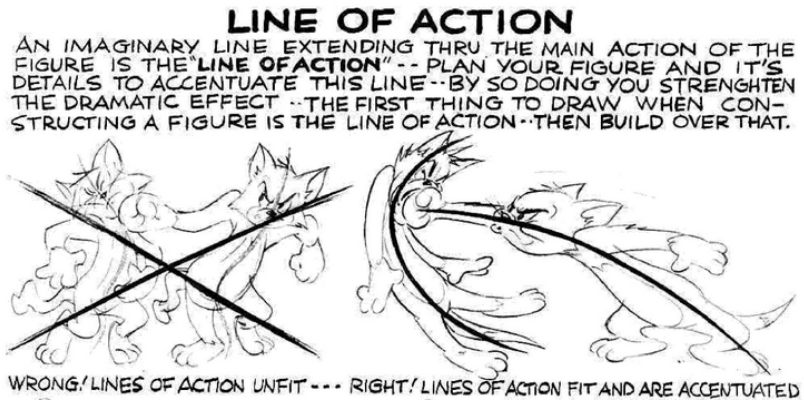


FIGURE 2.6: The concept of Line of Action ©Animation, P. Blair, 1948 [Bla48].

More recently, some investigated a simpler abstraction inspired by the workflow of traditional cartoonists [LB84; Bla48], the Line of Action (LoA). The LoA is a single smooth stroke over the character's body which indicates the general aesthetic shape of the pose, as illustrated by Fig. 2.6. It has been shown to be an intuitive interface to produce expressive poses, manipulating either the skeleton [GCR13] (illustrated in Fig. 2.5 (B)) or the character mesh directly [Özt+13]. The method has been generalized to allow the user to provide his own curves [Hah+15].

Data-driven Inverse Kinematic

A more recent group of methods leverage existing real-world animation data to generate poses which correspond to some provided user constraints. The general idea is to match the constraints by producing a new feasible parameterization extracted from a database, or to synthesize a new one similar to the database. We present data-driven IK methods here as a majority of them is interested in generating a full character pose based on the target parameterization of a few of the skeleton's joints, which is in turn applied to character posing.

The initial approach from Rose *et al.* [RSC01] uses radial basis functions to interpolate between multiple poses whose joints most closely match the provided targets. Grochow *et al.* [Gro+04] propose to model the probability distribution of the database using Gaussian Process Latent Variable Models. The new pose is generated at run time

by optimizing a function which represents the likelihood of a given pose to satisfy the constraints. The method was later improved to take into account the physical properties of motion [LHP05].

Another family of methods suggest searching large pose databases for the closest match. The main limitation is the complexity of the search: in a naïve interpretation it grows linearly with the size of the database. To reduce this complexity, the proposed methods suggest reducing the dimensionality of the data through PCA [CH05; RB09], or speeding up the searching with the help of kd-tree clustering [Krü+10; WTR11] or maximum a-priori (MAP) estimations [WC11]. Other learning methods have been applied to character posing, including hierarchical clustering [OH06], sparse dictionary learning[Lai+12] and multi-variate Gaussian distribution models [Hua+17].

3D posing interfaces

While all the previously described methods allow easier manipulation of character poses, they collectively face a final limitation: their goal is to handle a 3D object, the skeleton, through a 2D interface, the mouse and keyboard. Such indirect manipulations are notoriously hard to master and require the animators to be trained with a specific software interface. To mitigate this limitation, a group of methods focuses on direct manipulation interfaces. A first group of such work focuses on building tangible pose editing devices, in the form of actuated puppets [Kne+95; Yos+11; Jac+14], allowing animators to pose a physical representation of the character (see Fig. 2.7). Another one is interested in Virtual Reality interfaces [LCM20], which offer opportunities to solve the problem by embodying the user in a full 3D world. Interfaces have been explored where the user manipulates the skeleton in 3D through IK handles [Sum+11] or by tracking its hands and interpreting their motion as strokes similar to the Line of Action [GRC19].

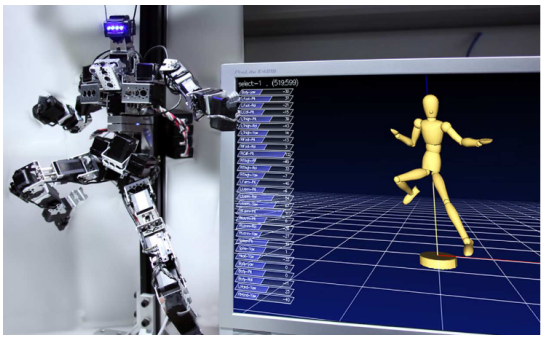


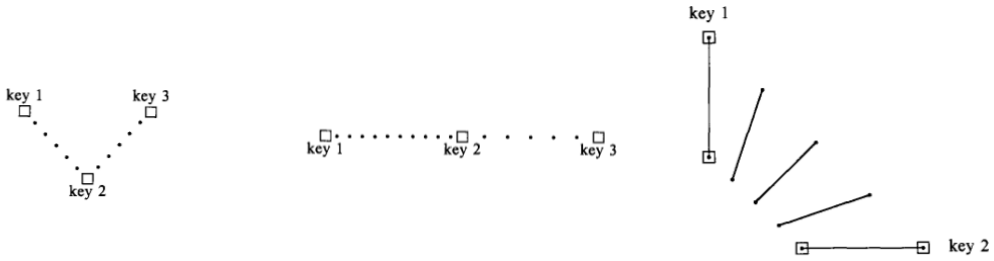
FIGURE 2.7: A physical pose editing interface through an actuated puppet. From Yoshizaki *et al.* [Yos+11].

2.3.2 In-betweening

Once key poses have been set, the software is tasked to generate intermediate poses (generally called *in-betweens*) between them to produce motion. To achieve this, the straightforward approach that comes to mind is to linearly interpolate between key-frames. Linear interpolations however suffer from limitations [KB84], as illustrated in Fig. 2.8:

- linear interpolation of orientation values provoke lengths distortions;
- sudden changes in the direction of motion might produce non-smooth animation;

- irregular time-placement of key-frames can lead to discontinuity in speed;



(A) Discontinuity in direction (B) Discontinuity in speed (C) Distortions in length when rotating

FIGURE 2.8: Linear interpolations limitations for animation, Kochanek 1984 [KB84]

A common solution to the first problem is to represent joint rotations by quaternions, and to interpolate using the spherical linear interpolation function (Slerp) [Sho85]. The second one was solved by smoothing the edges using cubic interpolation splines (using Catmull-Rom splines [KBB82] or B-splines [SB85]), and the third by allowing the user to parameterize them through control points [BH89; KB84]. Taken all together these solutions amount to a set of parametric curves (often called *animation curves*) that the animator can manipulate to control the interpolation process (Fig. 2.9).

This formulation of the animation curve is currently prevalent in production setups. The resulting motion's naturalness is however still dependent on the user's skill and carefulness, and so some more work has been dedicated to improving the process.

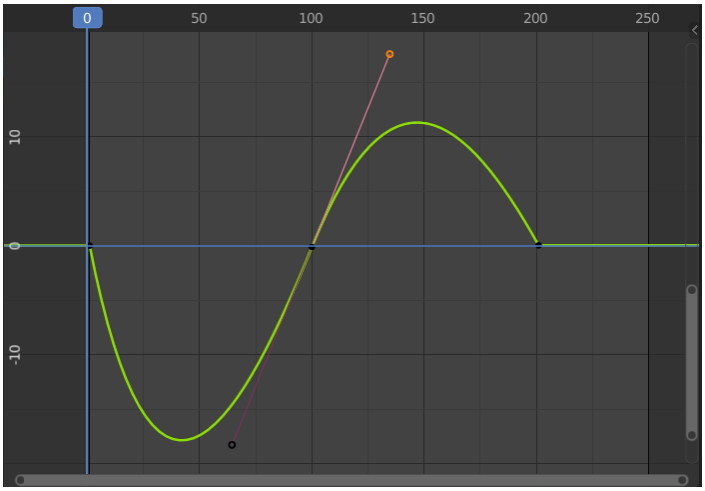


FIGURE 2.9: Animation curve in Blender. The curve represents the value over time for a single component, parameterized by 3 keyframes (black dots) and the interpolation splines at each of them. The control point of a spline is shown in orange.

A group of work propose to optimize the animation curves to satisfy all sorts of constraints. They have been applied to ensure that the motion remains physically plausible [SN88; LC95], and particular focus has been put on detecting and avoiding self collisions [Bad+94; Neb99].

More recently, another group focused on refining the curve editing process itself, making use of user-in-the-loop optimizations to gradually modify the curves with regard to the user input [KG18; CÖS19].

Data-oriented solutions have also attempted to improve the process, learning real-world constraints from motion capture data. The general idea is to generate the in-between frames while using this knowledge to ensure that the generated poses are all realistic. The first approaches used statistical modelling combined with optimization [CH07] or Gaussian processes [Gro+04]. Recently, recurrent neural networks (RNN) have been trained to produce in-betweening for a single specific character [Zv18; HP18]. Such methods are described further in the deep learning section below.

2.4 Reusing motion data

Producing animation through the interpolated key frames workflow as described in the previous section is a complicated process, and creating high quality motion remains a time-consuming task even for experimented artists. MoCap can allow arguably faster production, but the capture is not a light process. Actors, technicians and animators must be coordinated, and most production studios do not have the necessary technology on-site, so careful planning is mandatory. A missing or unfit animation clip arising in the middle of production involves re-shooting it and thus important time costs.

This complexity in the acquisition of motion data has led to an interest in ways to re-use existing animation and a fair amount of research has been devoted to the question.

2.4.1 Motion signal processing

The first group of work that proposed a solution to alter and reuse existing motion took inspiration from existing signal and image processing literature [BW95]. In these approach the motion of each animation parameter (such as a joint's rotation) is considered as a sampled signal, which can be filtered and decomposed in multiple resolution scales. The applications shown in the original include tweaking an animation by editing the values of individual bands, changing the timing of a clip (time-warping [WP95]) or using the signals as interpolation targets to blend multiple clips. Following work have shown applications in adaptation to random terrain or morphing the character's skeleton size [LS99; LS01], or altering the style characteristics of the original motion, by applying random noise to different resolution scales [Per95], or by applying a style transformation computed as the difference between a "neutral" motion clip and another "stylized" one [ABC96].

2.4.2 Motion blending

The goal of *motion blending* is to interpolate between two (or more) distinct animation clips to generate a seamless transition between them. While the blending operation can be used on its own to create new motion, it can also be used to alleviate the amount of original animation needed in a production. For example, if the short transitions between the longer clips of a character's animation set are automatically generated, animators can focus on refining the details of the most characteristic parts of its movement.

The very first approaches to blending used linear interpolations to satisfy user-provided constraints, such as IK targets, either directly in the animation parameter domain [GRG96; WH97], or in the spectral domain through Fourier decomposition [UAT95] or the aforementioned multi-resolution technique [BW95]. The main restrictions of these early approaches lies in the linear interpolation operator which cannot represent the subtlety of human motion and thus limits blending to similar input clips.

To alleviate this limitation, one family of techniques proposes to use Radial Basis Functions (RBF) to interpolate between motion clips. The seminal work of Rose *et al.* [RCB98] presents a new parameterization of motion as a composition of verbs (categories of motion) and adverbs (hand-labelled motion styles). Using RBFs to interpolate between verbs enables the system to generate variations of smooth transitions. Later improvements demonstrated constrained blending [RSC01] and improved performances by reducing data dimensionality [LT02; PSS02].

One limitation of these blending technique is their inability to respect constraints if a satisfying result is not present among their input motion. One other family thus propose to reverse the usual paradigm and generate the blending weights through a parametric constraint space [MK05; HK10]. In these inverse methods, the difficulty becomes finding an appropriate motion clip to blend to among a potentially large motion database. Another group of work then proposes to facilitate finding blending candidates by clustering similar clips together using k-Nearest neighbors [KG03]. The method has then received improvements in the form of pseudo-examples used to improve the motion space coverage [KG04].

A more in-depth analysis of some motion blending techniques, including numerical evaluations and comparisons, can be found in the review by Feng *et al.* [Fen+12].

2.4.3 Motion graphs

One of the objectives of data-driven animation synthesis is to organize a database of existing motion clips in a way which allow playing them back to back, on a same character, without breaking the animation. This type of organization is also important in interactive media such as video games: virtual avatars must be able to move and act across the world while responding to user input, which means seamlessly transitioning from the current animation to the one corresponding to the latest player input. The problem grows exponentially with the number of available actions available to the player, as each new task could be combined with the existing ones. To tackle this complexity, a group of work has studied the idea of building *animation graphs*, in which each edge represents an animation clip, and each node represent a pose, which represents a transition point between two (or more) clips. In this configuration, traversing the graph produces a single smooth animation in which each clip is played with no noticeable transition.

In their first appearances, motion graphs were fixed and structured by hand. Research focused on demonstrating their use, describing methods to search the graph to find paths satisfying user constraints, such as joints positions or a target root motion signal [MBC01; AF02; Lee+02].

In order to facilitate the preparation work required by motion graphs, various authors investigated ways to automatically build an animation graph from a motion database [KGP02; SH07; Zha+09; RZS10; Gle+08]. The automatically built graphs can be

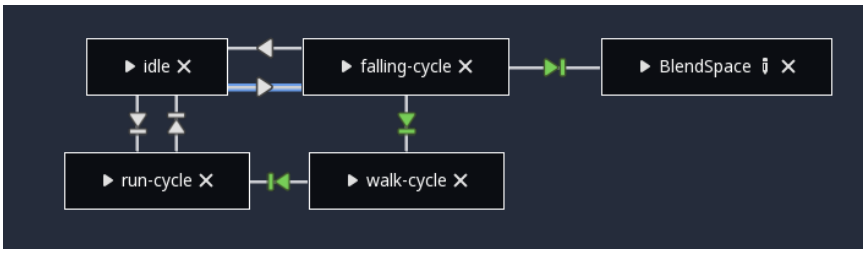


FIGURE 2.10: A (simple) animation graph in Godot Engine. Each node represents an animation clip, each edge a possible transition.

used in the same way as the manual ones, i.e. searched and traversed to satisfy user provided constraints.

The searching costs of a motion graph can grow exponentially along with the number of edges. Another group of work thus focuses on improving the process, using specialized path-search algorithms [SH07] or pre-computing cost functions [LL04; SMM05].

The "quality" of motion graphs have also been studied, with methods to evaluate the capabilities of a graph in a given scenario [RP07] or to enhance their connectivity [ZS09].

Some others propose alternative graph definitions where the graphs are denser and more structured. Shin and Oh [SO06] introduce "fat graphs" in which clips between which blending is possible are grouped together. In [HG07], both node and edges contain multiple animation clips. A walk through these *fat graphs* can produce even more motion by blending between available animation clip at each step. Having multiple options also allows for finer constraining, at the cost of search efficiency [MC12].

Motion graphs only allow transitions between clips when reaching nodes, which is limiting in applications where interactivity is paramount. Because they only use pre-existing animation data, coupling them with techniques which perturb the original clips (physics simulation, layered animation...) is difficult, as it would require straying away from pre-determined transitions. To alleviate these limitations, some work has proposed interactive controllers, learning to navigate the graph through reinforcement learning [TLP07; MP07].

Motion graphs are now widespread in the games industry. The graphs are however still hand-built, and used as state-machines to drive animation from character state (Fig. 2.10). Automatic construction approaches have indeed proven difficult to control, which could cause unacceptable latency in the response to player input. The graph construction itself has then become a time-consuming task for animators, which has led the research community to exploring other avenues for animation synthesis.

2.4.4 Motion fields and motion matching

Interactive applications require quick reaction to user input and external perturbations, for which motion graphs come lacking. Another family of approaches has then emerged: instead of modelling the single most suitable motion from any possible character state, as done with graphs, *motion fields* [Lee+10] suggest modelling a *set* of suitable motions. The field itself is composed of vectors containing the extracted instantaneous pose and velocities from each motion clip. At each frame, the most suitable motion clip according to the user input is selected interactively and blended into. The control

of the flow is based either on Markov Decision Processes (MDP) or reinforcement learning. As a result the current state is always a blend of multiple animations, staying in the general vicinity of them while immediately reacting to user input.

More recently, another method called *Motion Matching* has emerged from the video game industry. First presented by a team from Ubisoft [BC15; Cla16; Zad16] as a greedy simplification of motion fields, the method keeps the same main idea: at every frame, search for the best suited transition over an entire motion capture database. Motion Matching drops the complex task of constructing the field, as well as the control methods. Instead, the whole database is kept in memory, and the best matching clip at each time step is selected through nearest-neighbor search. The simplicity and controllability of motion matching has quickly drawn attention from many game studios [Har18; Zin19; Büt19] which extended the method to suit their need.

The main limitation of Motion Matching is the memory footprint of the animation database, which scales linearly with the amount of data. The academic field has started to take interest in the method and provided attempts to handle this problem. For instance Holden *et al.* [Hol+20] replace different parts of the process with neural networks, removing the need to store animation data in-memory.

2.4.5 Statistical modeling

One last family of approaches proposes to construct statistical models of animation data. A proper model can be helpful in analyzing and describing the subtle complexity of motion data, but most importantly in our case, it can be used to generate new motion with respect to its underlying distribution. Borrowing various modeling techniques from other fields and leveraging the increasing amount of available material, the models presented by researchers have grown more and more precise and powerful.

Principal Components Analysis (PCA)

The first statistical models of motion extend existing mathematical models of 3D objects deformations to incorporate motion. They rely on Principal Components Analysis (PCA) to transform animation data in uncorrelated variables and select the most representative ones, resulting in a more compact and general representation. For example, Alexa *et al.* [AM00] apply PCA on single animation clips directly, in order to learn a new representation decoupled from the geometry. Such representations of motion data have also been used to speed up the search process in a motion database [FF05]. A popular use of PCA has been to use it to reduce key frame data dimensionality, then to model the probability of frame-to-frame transitions (temporal dynamics) with variations of Markov Models [Bow00; BH00; MH00; GJH01].

The reduced subspace representation of motion data obtained through PCA can also be useful in order to reduce the size of the search space of optimization methods. This property has been leveraged to optimize motion to respect various constraints such as physical rules [SHP04], key frames and trajectories [CH07], or sketching interfaces [MCC09].

Gaussian Processes

Gaussian processes and their derivatives have also been used to model the distribution of motion data. Thanks to their probabilistic nature they can be used to generate any

data points instead of being strictly limited to inter or extrapolations. The seminal work of Grochow *et al.* [Gro+04] used Gaussian Process Latent Variable Models (GPLVM) to model the *pose* space and demonstrated their purpose with an interactive pose editing interface. Further work employed recurrent variants of Gaussian Processes to predict future frames [WFH08], automatically respect physical constraints [YL10; WMC11] or build character controllers [Lev+12]. Gaussian methods have produced impressive results, but have not seen much adoption outside the academic field. This can be attributed to their inherently high memory cost, since they need access to the full motion database in memory in order to be used.

2.5 Deep Learning

From 2015 and on, encouraged by their success in related fields, deep learning methods appeared in the animation community. They proved to be effective at learning complicated patterns from raw, real-life data, with the additional advantage of not requiring the training data to still be available after training. On top of that, their inference process usually consists in a set of straight-forward deterministic operations that can be executed fast enough for real-time applications. Thanks to their general popularity across research fields and these interesting properties, deep learning-based methods quickly grew to represent a large portion of new work in computer animation research (see Fig. 2.11). A succession of reviews have been published [WCW14; AP19; Mou+22] in attempts to keep up with this fast growth.

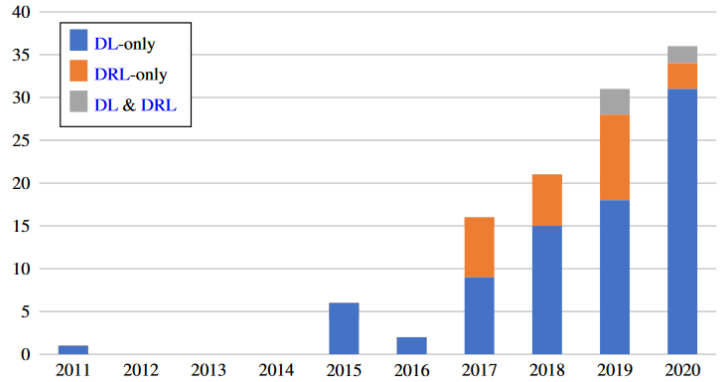


FIGURE 2.11: Histogram of the volume of peer-reviewed publications in human skeletal animation using Deep Learning (DL) or Deep Reinforcement Learning (DRL) over the past decade. From Mourot *et al.* [Mou+22]

Deep learning methods have been applied to a broad array of animation tasks, and many network architectures have been studied; the combinations and intertwining of both making classifying existing work difficult. In this thesis we choose to separate both part and give an overview of the applications in a first section, only surveying the architectures in a second one.

2.5.1 Applications of neural networks in animation

Motion synthesis The first applications of deep learning in animation followed the general idea of previous work with statistical models: model the distribution of motion data, and generate new motion sequences as varied as possible, whilst ensuring

the naturalness of produced motion and in some case respecting some user-provided constraints. In the second half of the 2000s, Taylor *et al.* [THR06; TH09] pioneered the use of neural networks with this task, studying variations of Restricted Boltzmann Machines (RBMs) to predict the next pose in a motion clip. Over the years, the task remained one of the most studied, but has been derived in a series of flavors.

On one end of the spectrum, *deterministic generation* applications' goal is to extrapolate an existing motion to closely predict its near future [Fra+15]. The difficulty then lies in the accurate reconstruction of a truncated motion sequence.

On the other, *generative synthesis* refers to the task of producing realistic motion sequences from a random seed [Hol+15]. Here, the objective is not reconstruction but rather the diversity and realism of the produced animation.

Between the two exist a myriad of similar tasks, whose objective vary between reconstruction accuracy and variety. But they most notably differ in how the synthesis of new animation is constrained. The most common one is to enforce the match with a past context to synthesize various follow-ups to an existing clip [Fra+15]. The context can however also be a future one, or past and future context can be enforced at the same time [Ber+15; Zv18; Har+20] (in which cases the task becomes similar to motion in-betweening, discussed earlier). Constraints are also used to provide some control to a user, or to restrict the possibility space of the generation, in which cases the synthesis in constrained by a user-provided trajectory signal [Hab+17], action types [Gho+17; Gho+20], or the character's environment [Cao+20].

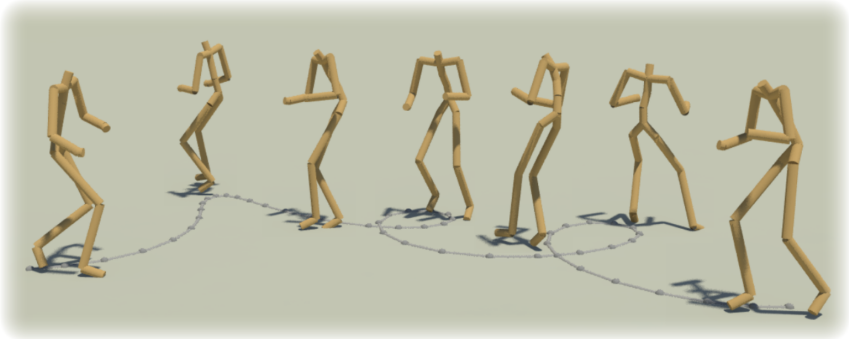


FIGURE 2.12: Motion synthesis constrained on a user-provided trajectory signal, from Holden *et al.* [Hol+15].

Motion control The next task attempts to use deep learning to create character controllers which react interactively to user input, as well as the character's virtual environment. Two categories of solutions to this problem respectively train joint-actuated, physically simulated agents [Pen+17; Pen+18] and musculoskeletal models of the human body [Gvv13; Nak+18], to act in a physically simulated world. As these techniques are not directly related to the topic of this thesis, we focus on the last category, which aims to generate the animation data used to drive a skeleton.

This family of approaches has been pioneered by Holden *et al.* [HKS17]. In their work the user is able to provide a phase function describing the alternating footsteps of the character, and at each frame the network can provide the next based on the current pose, the current phase, and the desired direction vector, providing an impressive character locomotion controller (see Fig. 2.13). This approach was later improved,

notably by using multiple expert networks focusing on different phases, orchestrated by a gating network [Zha+18; Sta+19; Sta+20].

While most of the early approaches to deep-learning-based motion control focused entirely on locomotion, more recent work have started taking interest in other tasks such as dribbling with a basket ball [LLL18; Sta+21].

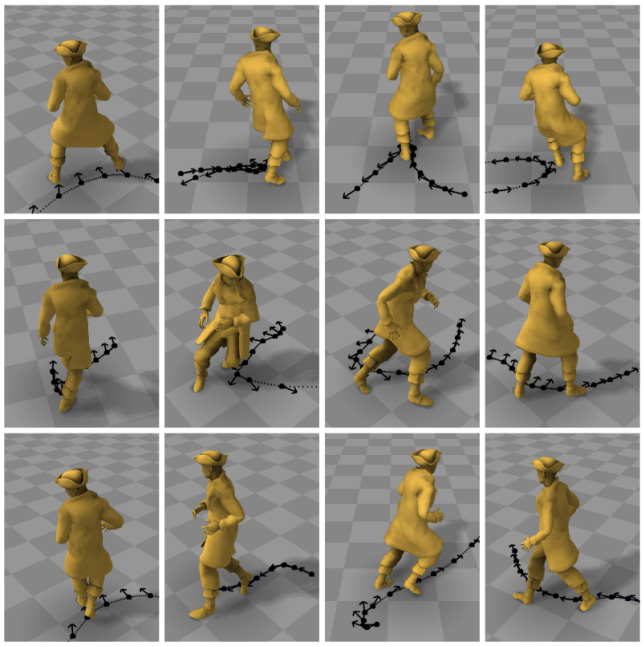


FIGURE 2.13: Character control using neural networks, from Holden *et al.* [HKS17]. At each frame the network generate the next one based on the user's input on a gamepad controller.

Motion editing The final part of the deep learning literature for character animation focus on editing, rather than generating, motion. The first of such applications were a byproduct of techniques developed for other tasks. Many early neural methods leveraged learned "motion manifolds" to which real animation data is mapped to and from. Because the decoding operation is trained to produce realistic motion, projecting corrupted or noisy data to the manifold and back to motion space can reproduce and clean up the missing information [HSK16; Büt+17; Wan+21]. Some work has also been dedicated to refining this property, by modifying the training loss to penalize bone-length error [Li+19] or by optimizing the latent space [LAT21].

Some work has also investigated how deep neural networks could help in retargeting animation, so that a motion sequence created for a character could be adapted to another one with a different morphology (see Fig. 2.14) [Vil+18; Kim+20; Lim19; Abe+20].

A last part of the literature is interested in using deep neural networks to extract and transfer the style of a given animation to another. Style being of the focuses of this thesis, these methods are described along with their non-neural equivalents in a later section (2.6.3).

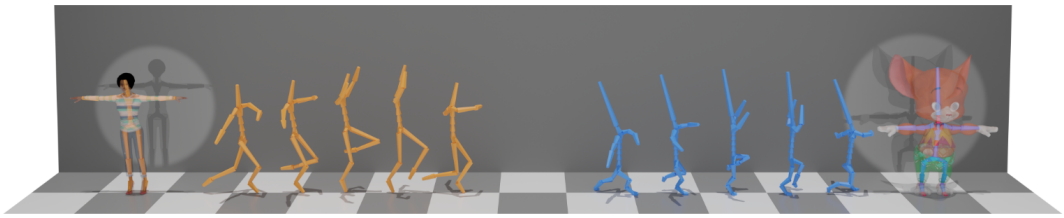


FIGURE 2.14: Neural networks have been used to retarget existing animation to a different skeleton. From Aberman *et al.* [Abe+20].

2.5.2 Network architectures

With their immense popularity, deep learning methods have been extensively studied during the past decade, inevitably spawning a broad array of models and architectures. The animation community has not been spared, and many of those have been applied to motion data, one of its underlying difficulty being its spatio-temporal aspect: the temporal coherency of a sequence of frame is important, but so is the relations between each joints' parameters and the kinematic structure of the skeleton.

Recurrent Neural Networks One of the first architectures applied to animation data is RNNs, in which the encoding of a pose is fed back to the network when encoding the subsequent ones, to model the temporal dynamics of motion sequences. The seminal work from Fragkiadaki *et al.* [Fra+15] uses an encoder-decoder to construct a latent representation of poses, then a RNN to model the dynamics between them. This general architecture was later extended to include structural information on the skeleton [Jai+16; AKH19; GC19], and improved by changing the pose representation from joint angles in Euler representation to joint velocities [MBR17] or quaternions representation [Pav+20].

The main limit of RNNs appear when attempting to generate longer motion sequences is *pose collapse*, which translates in the network progressively generating the same pose over and over. To reduce the problem, following approaches suggest feeding the training network with either noisy poses [Fra+15; XLM19; KGB19] its own predictions [Li+18; Gop+19].

Convolutional Neural Networks Around the same time, the network used proposed by Holden *et al.* [Hol+15; HSK16] proposed a derivative of CNNs on fixed-width sequences of motion, using 1-dimensional convolutions on individual pose features to model their temporal dynamics.

Graph Convolutional Networks Later, some others leveraged the hierarchical structure of the skeleton through (GCN). In these approaches the skeleton is represented as a graph rather than a kinematic tree, and the weights of its adjacency matrix are learned by the network [Büt+17; Mao+19; CSY20; MLS20; Li+20].

Variational Autoencoders Generative models focus on modelling the underlying distribution of a dataset, in order to be able to draw new samples from it and generate unseen data. Among this family, VAEs were the first to be applied to animation data [Hab+17; Yan+18; Du+19; Ali+20]. Much like autoencoders, VAEs learn to non-linearly map data samples to and from a latent representation. They are in addition trained to constrain the latent samples to be part of a pre-defined distribution, which

provides them their generative power: drawing samples from the known distribution only yields valid motion, similar to but not exactly part of the distribution of the training data.

Adversarial approaches Next, a series of work took interest generating diverse motion using another family of generative models: GANs [BKL18; LA18; Wan+20]. In this case, a generator model is trained to produce a new motion sample from a random input. In parallel, a discriminator network is trained, whose task is to distinguish real motion sequences and generated ones. During training, the generator's goal is to fool the discriminator into classifying its generated sequences as real ones, and the quality and variety of the generation improves.

The general idea of using a discriminator to increase the variety of the generated samples has been applied to all kind of other architectures, often to alleviate the lack of encoding capacity of vanilla GANs [KGB19; WCX21; HGM19]. Ensuring that neither the generator nor the discriminator outperform the other during training is however a delicate task, which prevented more general adoption of the idea.

Attention and Transformers Another class of approaches make use of attention mechanisms in their networks' architecture to allow them to learn to focus on smaller but more important patterns in the skeleton data [MLS20; ZPK20]. This method has also recently been generalized for sequences using the popular Transformer architecture [Cai+20; Aks+21].

Miscellaneous Nowadays, new architecture keep emerging in other fields, yielding ever more impressive results. They inevitably also raise interest in the animation community, and very recent work is interested in modelling motion data using newer architectures such as normalizing flows [HAB20], Neural Radiance Fields (NeRF) [He+22], or Diffusion Models [Zha+22].

2.6 Motion style

Humans are very sensitive to the way each other act, and we are able to perceive a lot of information on the internal state of an individual simply from observing the way he or she moves [BS07]. Conversely, we are also able to tell right away if a virtual character's motion is not "right", although explaining *why* it is not so might not be that easy [MMK12].

The methods described in the previous sections facilitate and speed up the creation of character animation. However, producing motion that can convey to spectators the same kind of information about the character's personality, intents or emotional state as a real person's movement remains a daunting task. Crafting such animation require precise knowledge in physiology and anatomy on the animator's side, so tools to help them in the process have been studied. In this section, we are interested on methods that rely on the metaphor of *motion style* as a creative interface.

Nuances in the ways characters (and especially humans) move, and their relations with how we perceive them has been studied extensively, but no consensus was found on a way to categorize them. Thus, we first summarize the different definitions of "style" used in the computer animation community at large. We then expand on the various systems that have been proposed to analyze and quantify motion and its style,

and finally review the actual style-based motion synthesis and editing found in the literature.

2.6.1 Definition(s)

The notion of *motion style*, while commonly used throughout the literature, does not always have the same definition [BS79; Gal92]. A recent review by Ribet *et al.* [RWV21] acknowledges this observation and goes over the literature, proposing a taxonomy of the definitions they find. They separate them into three larger categories: style as a component of motion, style as a variation of motion, and style as individual-related features.

In the first definition, style is considered as a *secondary* motion which can be applied on top of a *primary* one [MC90; EA16]. Among this group, some also consider the existence of a "neutral animation" over which any style could be layered [ABC96; Cre+20].

In the second, style is considered as a variation around a general motion theme, although the concept of variation itself is also vaguely defined. Some consider related actions to be different styles in the same theme (i.e. walking, running, strutting are all style variations of the same theme, which could be "locomotion") [BH00; CL06]. Some generalize the variations to differences in speed or frequency [Ma+10]. Finally, some others see variations in styles as interpersonal differences, encoding an individual's specific way of moving [Urt+04; WFH07; TMD12].

The last definition, which is perhaps the most common one, is to define styles as discrete semantic features pertaining to an individual's characteristics [Bar+13; Hol+17; AYB17; Cre+17; KL19]. The review further separates the considered features in several categories:

- Emotions ("happy", "sad", "angry", etc.)[3];
- Biological features (age, gender);
- Physical states ("weak", "strong", "injured");
- Personality features ("cool", "confident", "stressed")
- Behaviors ("catwalk", "childlike", "zombie")

2.6.2 Analysis and quantification

Creating systems to accurately describe, document and quantify human motion has long been a topic of interest, with early work appearing in the fields of anthropology [Bir55], psychology [EF69; McN92] and linguistics [Ken80]. Motion notation systems are also interesting to the computer animation community as they can provide meaningful abstractions for the notion of motion style. In the following section, we review the most commonly used ones in the literature.

Laban Movement Analysis (LMA)

The LMA system, developed by dance theorist R. Laban [LU71], is widely used by dancers, choreographs and actors, but also computer scientists and roboticists. It

[3]Defining and classifying emotions is also a complex topic [KK81].

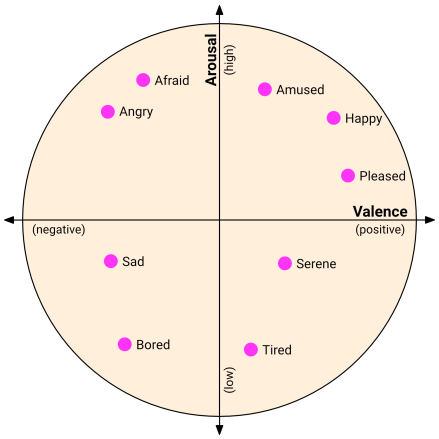


FIGURE 2.15: Graphical representation of the 2D arousal-valence scale. Positions of affective terms are placed indicatively.

corresponds to a framework in which motion is described in four separate categories, each category depending on the others. A complete description of LMA is the subject of complete books, in this thesis we merely present a short overview [Gro95]. The categories are as follows:

Body. Relates to structural and physical characteristics of the body. It describes the body parts that are moving, which are in contact, or which are influenced by others. It also describes if and when the body parts are moving simultaneously or sequentially.

Effort. Pertains to the dynamic changes in the expression of movement. It is further separated in four qualitative elements: space (direct or indirect), weight (light or strong), time (quick or sustained), and flow (free or bound). The combination of these elements creates "states" which can be used to describe the instant of change, the rhythmic variations of motion.

Shape. Represents the overall shape by taken the body (such as curvatures and symmetries) and the changes of this shape over time

Space. Describes spatial patterns and trajectories, as well as how the body occupies its surrounding space

To better describe the analyzed motion, a notation system, dubbed Labanotation, has been developed along LMA [Gue05].

The Arousal-Valence scale

The Arousal-valence scale (sometimes Russell's Circumplex Model (RCM) [Ari+17]) is another model used to represent emotions, attributed to A. Russel [Rus80]. In this model emotions are placed on a 2-dimensional plane where the axis represent arousal and valence. Low arousal is associated with boredom, high arousal with excitement, while valence goes from sadness to happiness. Emotions can then be placed in a fuzzy manner on the plane as shown in Fig. 2.15.

Motion descriptors

A segment of the research in computer vision and computer animation has also been dedicated to quantitatively measuring motion, in order to analyze, recognize or compare motion. Early work in this direction operate on video captures and take

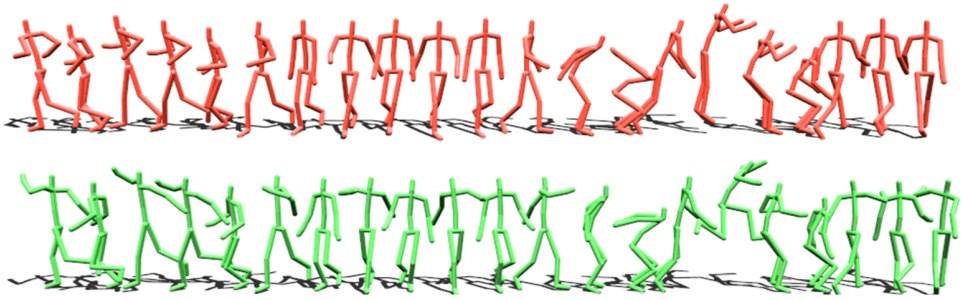


FIGURE 2.16: Illustration of the style transfer task. A random motion clip in the "neutral" style (top) is transformed to the "proud" style (bottom) [Xia+15].

inspiration from LMA to design quantitative features related to the shape and space qualities [CMV04]. Other approaches rely on emotion models from the psychological field [Pel09; Glo+11], or define descriptors specific to dance motion [Ala+12].

Some authors then leveraged the growing availability of motion capture data by designing geometric features describing the relationship between skeleton joints over time [MRC05], or LMA-related features [HTY05; Kap+13], although no consensus has been found.

In a recent work, Larboulette and Gibet propose to solve this issue by reviewing the available computable descriptors and aggregating them, providing a useful toolbox of formalized formulas [LG15].

2.6.3 Generating stylized motion

The many factors that make an animation belong or not belong to a style are difficult to explain, and even more difficult to reproduce during the creation process. Being able to easily generate stylized animation, and to modify the style of an existing one, would then be an interesting tool to provide to animators. It would allow users to easily experiment with different styles or to generate a wide array of variations from an existing sequence.

In this section we give an overview of the different tools following this idea proposed by the research community. The methods are separated in three categories: style transfer, style-based synthesis and style-based editing.

Style transfer

The first applications of the notion of "style" in computer animation was style transfer, i.e. the task of modifying an existing motion clip to match the style of another one, without modifying its content. This is similar to motion blending (see 2.4.2), although an added difficulty comes from the new explicit separation of aesthetic characteristics and functional role of a specific motion, which is inspired from contemporary work in related fields [TF96; TF00].

Amaya *et al.* [ABC96] present the first approach to the task by computing a style translation as the difference between a neutral and a stylized animation. Extending this idea, some others learned to transfer the style using statistical models such as Linear Time Invariant models [HPP05] or Independent Component Analysis [SCF06].

In a similar approach some others operate in the spectral domain to extract dynamic characteristic of stylized motion [UAT95; YM16].

Neural networks have also been used to learn to transfer style. Early solutions train specialized networks per style, using RBF neural networks [EA14] or mixtures of autoregressive models [Xia+15] to describe the difference between an input and an output style, that can later be used for style transfer. Inspired by successful style transfer methods for images [GEB16], Holden *et al.* transfer the style from one motion to another by optimizing the latent representation of the output motion clip to match the Gram matrix of the input one's [HSK16]. They later achieve the same result faster by replacing the optimization process by another neural network [Hol+17]. A final group uses supervised setups to learn style characteristics for a semantic label, which can then be used to apply the related style to another motion sequence [Wan+18; Smi+19].

Synthesis

"Stylistic synthesis", in the context of computer animation, refer to the generation of new motion based on a style information, which could be given through examples, or higher level label values.

A first group of work extend existing interpolation approaches to match a user-provided style constraint. In their seminal work, Rose *et al.* [RCB98] use such a method by parameterizing the interpolated motion in verb (content) and adverb (style) classes. Grochow *et al.* [Gro+04] train per-style GPLVMs to solve the IK problem while generating style-constrained poses. By interpolating the probability distributions of the models they are also able to mix-and-match styles to produce new ones.

Another group employ statistical modeling to produce new motion, but replace or augment the generation prior with a style variable. In this setting, both supervised and unsupervised approach have been employed. Unsupervised approaches take an example stylized motion and extrapolate the available data to produce new sequences in the same style. To achieve this, Brand *et al.* [BH00] first describe a learned state machine using Hidden Markov Models, in which motion sequences are organized by shared style characteristics. During synthesis, a style variable vector input can be used to vary the style of the generated motion. This approach was then declined a number of variations, using GPLVMs [WFH07], PCA [Urt+04], or CRBMs [TH09]. More recently, VAEs have used for their ability to model the probabilistic nature of human motion and to generate large variation of motion in the same style [Du+19] The supervised approach instead generate motion from a higher-level style label which is learned from previously labeled dataset. Torresani *et al.* [THB06] learn a multidimensional style space using hand-labeled Laban notations, then a mapping from this space to animation parameters. In the same spirit Min *et al.* [MLC10] use multilinear analysis to model in two dimensions, "identity" and "style" after PCA reduction, allowing them to synthesize new parameterized motion. Generative probabilistic neural networks such as invertible flows have also been used [Wen+21] and demonstrated their ability to model the style of a given input and generate variations on its theme.

Finally, taking the style metaphor from a biomechanical simulation point of view, Liu *et al.* [LHP05] have tackled problem by defining style in terms of muscle preferences and musculoskeletal parameters, which are constrained during the simulation to generate stylized motion.

Editing

The last group of work gathers methods that provide animators with tools to manipulate motion through style parameters. The idea, first formulated by Bishko in 1992 [Bis92], revolves around the observation that the previously described notation systems (Laban's in this context) are parametric, and so are computer animations, and so a mapping between them could be found. As she puts it: "It is possible to create animation software based on these parameters. This would create a metaphor of the dancer's movement processes, allowing animators a closer relationship to the physical feeling of movement while maintaining the kinematic process that is the animator's craft".

The initial implementation of this idea by Chi *et al.* [Chi+00] thus hand-craft mappings from Laban Effort and Shape parameters to animation parameters, and show that users can manipulate them to edit animation. This approach has then been extended, first with a natural language interface [ZCB00], then by replacing the arbitrary mappings with learned ones [EA10]. Durupinar *et al.* [Dur+16] for example label a motion dataset with Laban notations as well as OCEAN personality traits through a user study, then learn the mappings from OCEAN to Laban, then to Laban to motion parameters. This way, users can edit animation through empirical "personality" knobs[4]. In the same spirit, Aristidou *et al.* [ACC15; Ari+17] learn a bidirectional mapping from Laban features to the Arousal-Valence scale, then describe an editing framework in which a user can modify a motion clip by manipulating the emotions on the scale.

2.7 Conclusion

Creating high-quality, appealing motion is a complex task. Animating a character according to one's vision, while depicting the desired action and conveying the right emotion requires an advanced knowledge of anatomy and physics rules. The literature depicts a wide variety of approaches to create and edit motion data, ranging from precise, user-controlled skeleton manipulation, to automatic generation by extrapolating a database.

Learning-based approaches have received their fair share of interest, and deep learning methods in particular have shown impressive potential at modelling animation data. However, as shown by this review of the literature, they have until now been applied to high-level tasks, and their output is difficult to control from a user point of view. This is the limitation that this thesis focuses upon: our goal is to leverage the powerful modelling capacities of neural network in the context of expressive and controllable animation tools.

[4] OCEAN is a taxonomy that has been studied in psychology to define the personality of individuals through five semantic categories [Gol90]. OCEAN scores can be derived from the words used to describe a person. The model's relevance and methodological bases have however been criticized [Blo95].

CHAPTER 3

A latent space for pose editing

3.1 Introduction

As highlighted in our review of the literature, neural networks have recently proven capable of modeling the complexity of motion data, and of synthesizing convincing new animation by generalizing from a database of examples. They have been successfully

used to predict missing frames in motion clips, to generate locomotion animation from a high level control input, or to retarget an animation to a new skeleton. These methods produce impressive results, however they share the common drawback of being difficult to control. Animation is a precise craft, and animators expect to be able to tweak the output of their tools to fit their artistic intent. And in these existing applications, said output is pretty much out of their hands: if the generated motion does not meet their expectations, the only solution is to record some more motion capture data, add it to the dataset, re-train the models and cross fingers in hope that the desired behavior will appear.

Leveraging the subtle information learned by the networks to provide animators with smarter tools while leaving them in control therefore appears like an interesting venue for research. This observation motivates the work presented in this thesis, which proposes to explore the application space for such neural networks-powered animation editing tools. Among the different building blocks of the key-frame animation pipeline, we focus on *pose editing*.

A large part of the difficulty in posing a character comes from the required understanding of its motion range, possibilities and limitations, which are difficult to grasp and replicate. This information is however inherently contained in motion capture data. We therefore propose a group of methods for data-driven pose editing with the goal of shifting more of the task of assessing realism from the artist to the computer, and to provide easier access to non-experts.

The general idea of this thesis is to rely on neural networks to automatically learn the constraints from MoCap data. In this chapter, we first give a general overview of the system (3.2), which creates a latent space on which poses can be mapped for edition, while the handling of subtle constraints is handled by the mapping functions. In section 3.3 we present the pose dataset used to train our neural networks, and discuss our choices in data representation. Finally, section 3.4 describe two different approaches in the construction of the latent pose editing space, and the network architectures used in both cases.

3.2 System overview

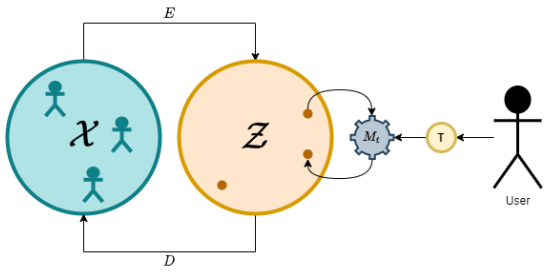


FIGURE 3.1: High-level overview of the latent space editing system. The functions E and D map real poses sampled from $\mathcal{X}$ to the latent space $\mathcal{Z}$. The function M_t operates on latent sample to edit poses with regard to a user-provided target value T.

A *character pose* in animation data represents the parameterization of the character's underlying skeleton at a given frame. Formally, the set of possible poses $\mathcal{X}$ for a given skeleton is a subset of the space of skeleton parameterizations $\mathcal{S} = \mathbb{R}^{J \times P}$, with J the

number of joints making up the skeleton and P the number of values parameterizing a single joint (i.e. position, rotation, scale, etc. The choice of P is discussed in the next section). Any skeleton configuration in $\mathcal{S}$ does not correspond to a visually correct or even plausible pose. Those are only produced by parameterizations that respect a set of constraints. Some of said constraints are straightforward, such as coherent bone lengths and limits on joints orientations, but some others are not, for example the sense of equilibrium expressed by a pose, the intent and emotions it gives off. These constraints are what makes pose editing difficult: in order to manipulate a pose while remaining in the realm of possible poses, one must specify and respect them.

Much like some other approaches from the data-driven family, we attempt to circumvent the problem by learning the implicit constraints directly from ground-truth data. A high-level illustration of our system can be seen in Fig. 3.1. We construct an alternative pose editing space, $\mathcal{Z} = \mathbb{R}^d$ (d is a hyperparameter designing the dimension of a pose's latent representation). Two transformation functions are also defined to map samples from one space to the other: the Encoder $E : \mathcal{X} \to \mathcal{Z}$ and Decoder $D : \mathcal{X} \to \mathcal{Z}$. We also introduce $M_t : \mathcal{Z} \to \mathcal{Z}$, a manipulator function which modify a pose to satisfy a given user-provided target t, and operates on latent samples. For the remaining of this thesis the three functions are modeled as neural networks whose parameters will be trained to fit our need.

Within this system, the implicit skeleton constraints may be enforced in one of two ways. In the first, the latent space is shaped in a specific way, so that E maps *any* latent sample to a possible pose. In the second, its shape is not enforced at all. Instead, M learns to navigate $\mathcal{Z}$ to only generate samples that will map to possible poses.

This chapter discusses the matter of designing E and D, and thus constructing the latent space, to accommodate both those ways. The specific design of M is explored in further chapters.

3.3 The pose dataset

In order to train the different networks, we build a large dataset of poses, obtained by aggregating multiple existing motion capture databases. This section describes these sources, the pre-processing step used to uniformize all their animation clips, then discuss the format in which the poses are fed to the networks.

3.3.1 Pose representation for neural networks

The choice of data representation plays an important role in the success of neural networks methods, and different representations can hide or reveal important factors of variation in the data [BCV13]. Animation data can be represented in many ways and the question of which is best suited to train neural network is a standing issue for the research community. One part of the differences lies within the pose representation (parameterized by J and P in our notation), which we aim to model, and so for which we need to make a choice. We first give an overview of the options used in the literature, discussing the strong and weak points of each. We then discuss the reasons behind our pick and describe the input format used in the work that follows.

Literature

Pose data is most commonly represented as a hierarchy of joints angles, where each joint is parameterized by its orientation relative to its parent. Being the representation of choice in the traditional animation pipeline, it might seem a natural pick for neural networks tools, as it would make their integration in existing workflows straightforward. It is however not the case, mostly due to individual rotations raising issues when it comes to modeling, and multiple representations of rotations themselves have been studied.

The first problem is a well-known one: all rotation representations parameterized by 3 components (such as Euler or axis-angle) are subject to a singularity called the gimbal lock, "loosing" a degree of freedom if two of their rotation planes align. To avoid it, early work used exponential maps representations where the singularities only happen with larger angles [THR06]. This approach was fit for the modelling of smaller angles, and was used to produce motion by generating pose offsets, but it is no longer a viable strategy for longer-horizon modelling. In order to avoid the problem, unit quaternions have been used in place [PGA18].

It has however been pointed out that quaternions, and all the previously used rotations representations, are subject to a second problem: they are non-continuous, in the sense that two parameterizations can amount to the same angle (i.e. 0 is equivalent to 2π) [Gra98; SDN09]. This is a specifically an issue when it comes to neural network modelling, as the discontinuities generate erroneous comparisons between close values. A recent study has highlighted this shortcoming, and proposed to avoid it by introducing new 5 and 6 dimensions representations [Zho+19], but its usage is not yet standard in the field.

The hierarchical aspect of angular representations is also limiting, as multiple angular errors in the kinematic chain accumulate up to large positional errors down the chains (Fig. 3.2). This problem has been tackled by computing the positional errors after finding the joints position through an expensive (in terms of computation during training) forward kinematics pass, or by weighting the loss terms of the joints to give a greater penalty to errors close to the root of the chain [Gho+20].

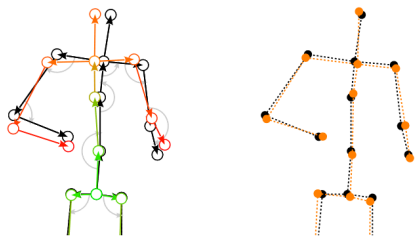


FIGURE 3.2: Illustration of the accumulation of errors with hierarchical angular representations: small angle errors in the kinematic chain can quickly accumulate to larger positional errors, especially noticeable in the end effectors (left). Positional representations (left) are much less sensitive to the problem. Figure from Mourot *et al.* [Mou+22].

The alternative option is to represent the pose using each joint's position, usually relative to the skeleton's root bone. Compared to angular representations, positions do not suffer from error accumulation, and the Cartesian coordinate system is continuous and free from singularities. However, positional representations do not ensure constant bone lengths the way angular ones do.

Since it is possible to convert from one parameterization to the other, and since both offer different flaws, none is strictly superior to the other. Rotations are closer to the format used in actual animation systems, while positions have historically been chosen for modelling, as the bone length difference can be alleviated at a low cost through post-processing. Some work has also been dedicated to hybrid or redundant representations [Abe+20; LLL18; Hol+20] and tend to indicate that these would be beneficial, but no definitive formulation has been agreed upon.

Input format for our neural networks

Considering these differences and limitations, and the fact that this thesis' goal is to explore new use cases for neural networks in animation rather than to improve their modeling power, we stick to positional representations, which allows us to build upon previous modelling work more easily. To accommodate for inevitable bone length errors, we suggest a simple work-around to the problem, see 3.3.3. However and fortunately, since our approach considers neural networks as generic tools for animation, it can easily be adapted to accommodate newer architectures and representations once they have stabilized and reached their full potential.

Our input skeleton, visible in Fig. 3.3, is composed of 21 joints. The pose tensor x representing its parameterization which is fed to the networks is obtained by concatenating the position of each joint in pose space, i.e. related to the pelvis' one. In other words, our choice of pose representations uses $J = 21$, $P = 3$, and $\mathcal{X} = \mathbb{R}^{63}$.

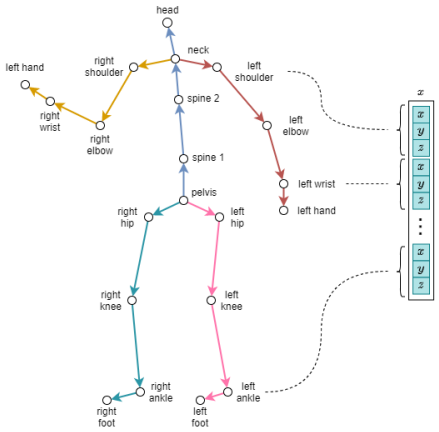


FIGURE 3.3: Illustration of the skeleton used as input. The kinematic chains are shown in different colors. The pose tensor p is obtained by concatenating the positions of each joint.

3.3.2 Source animation datasets

The success of data-oriented animation techniques is heavily reliant on the availability of large amounts of high-quality motion data. Fortunately, with the improvements brought to motion capture technology making it more and more accessible, such data is no longer as scarce as it used to be, and animation researchers can count on a large choice of motion datasets freely available online [Uni03; Ion+14; FP14; Vol+14; Mah+19; Har+20].

We build the pose dataset by processing a subset of these. While a larger dataset would undoubtably increase the performance of our models [Sun+17], the goal of this thesis is not to pursue a state-of-the-art value, but rather to explore different

application spaces for deep learning models. With this in mind, we restrain ourselves to a large-enough dataset of 1,5 million poses which we find sufficient to support our arguments, with the added benefits of lowering the training times of our models. Our source databases are: the CMU MoCap Database [Uni03], Emilya [FP14], and the clips from the Edinburgh University [HSK16].

The Carnegie Mellon University Graphics Lab Motion Capture Database (CMU) [Uni03] is perhaps the most commonly used motion dataset available. It contains 2605 motion recordings, acted by 144 subjects, over a large variety of actions, from simple locomotion samples to sports and multi-subjects interactions, which amount to roughly 10 hours of footage.

The Emilya database [FP14] is interested in how emotions are expressed in daily actions. It contains recorded motion clips during which 11 actors and actresses were asked to perform 7 "every day" actions while acting 8 emotions, for a total of about 25 hours of motion capture data. The affective states where induced by reading different scenarios to the performers.

The Edinburgh Locomotion dataset [HSK16] is focused on longer, uninterrupted locomotion clips, and was designed specifically for deep learning applications. Actors were asked to move around randomly on the capture set at different speed in order to record more variations of natural motion. It contains less data than the other two (about 2 hours worth of motion capture), but contains more random and continuous motion.

3.3.3 Preprocessing

The different source database present differences in the data they contain, and in order to create the dataset which we can then use to train neural networks, we pre-process them to bring them to a common standard. This procedure is inspired from the format conversion presented by Holden *et al.* [HSK16], although we adapt some of its step to fit our needs.

Each animation clip is retargeted to a standard target skeleton following the scheme proposed by Holden *et al.* [Hol+15]. First, the joint orientations of the source skeleton are copied to the target one for each matching joints. Then, the source skeleton is scaled to match the target's bone lengths. Finally, the joints of the target skeleton are moved to match the new position of the source's one with a full-body IK pass [YN03].

We then extract singular poses from the clips. For each frame, the global translation is removed, and each joint's position is calculated relative to the *pelvis* (root) joint. We find the forward direction of each pose by computing the cross product of the vertical axis with the average of vectors formed by the left-right shoulders and hips axis, then rotate them so that they all face the same direction. We also sub-sample each clip to 30 frames per second, to avoid filling the dataset with highly similar poses and speed up training times. The parameterizations of each joint are then normalized by subtracting the mean and dividing by the standard deviation of their value over the dataset, as is commonly done to prepare data before feeding it to neural networks, then concatenated to form the final input tensors.

It is also worth noting that although the dataset is composed of individual poses, it is not yet randomized, and neighboring poses from the same original clip are kept close together. This feature is leveraged later in this thesis when we require similar poses in terms of feasible motion from one to the other. For the same reason, very short

clips (below one seconds) are excluded, as are noisy ones and those with little motion. After this cleaning step, the final pose dataset is composed of about 1,5 million poses.

3.3.4 Post-processing

It is a common observation with neural networks working with joints position that the generated positions can be jittery, and the resulting poses can suffer from slight variations in bone lengths. Our models are no exception, and while the variation is not visually detectable most of the time, they can accumulate to be quite noticeable when a pose generated by the networks is fed back to them. We propose an optional post-processing step which can be applied to generated poses to ensure constant bone lengths. We use the backward step from the FABRIK IK solver [AL11], which iteratively pulls back each joint of the skeleton's kinematic chains with a very light computation cost. In our implementation using the kinematic chains illustrated in Fig. 3.3, this post-processing step takes only 1.5 milliseconds.

The algorithm for one of the skeleton's kinematic chain is outlined in 1. The joints are denoted p_n, with p_0 the root and p_n the end-effector. d_i represents the original length of the bone between p_i and p_{i+1}, initialized once with the reference skeleton.

Algorithm 1 The FABRIK backward pass [AL11]

for $i = 1, \dots, n-1$ **do**
  $r_i \leftarrow |p_{i+1} - p_i|$ {Find the new bone length between p_i and p_{i+1}}
  $\lambda_i \leftarrow d_i / r_i$
  $p_{i+1} \leftarrow (1 - \lambda_i)p_i + \lambda_i p_{i+1}$ {"Pull" p_{i+1} back to honor bone length}
end for

3.4 Architectures

We study two approaches for the design of the latent pose space. In the first the latent space is constructed without constraints by a baseline autoencoder, which serves the role of the E and D functions. In the second, we instead force certain properties onto the latent space, by using a scaffolding of autoencoder and GANs. For both cases, we describe the neural networks architectures and the training parameters used in further experiments.

3.4.1 Baseline autoencoder

The idea of building a latent space for complex data is not new, and so for a baseline experiment, upon which we can base further improvements later on, we turn to a staple in the literature: autoencoders. Autoencoders are made up of two neural networks tasked to learn an efficient encoding of some complex data. As illustrated in Fig. 3.4, the encoder E learns to map real data points to a learned, usually more compact, latent space; and the decoder D learns to map them back to the original data space. As this architecture closely matches our previously described system, autoencoders are a natural choice. In their original flavor, they also have the advantage of being conceptually simple and easy to train, which are interesting properties with regard to our goal of designing accessible tools. On another hand however, the latent space they produce is unknown and might not be suited for some applications. For example, it might not be convex and therefore interpolations within it would not be decoded

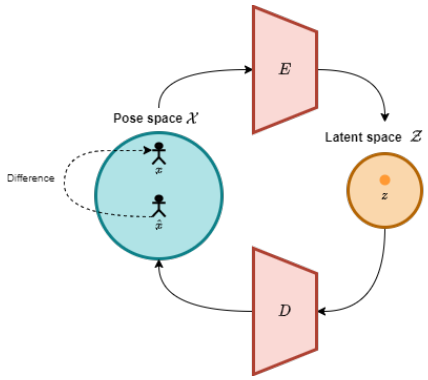


FIGURE 3.4: Overview of the autoencoder architecture. The encoder learns to map real samples to the latent space, and the decoder to map latent ones to the pose space. Both networks are trained to minimize the reconstruction error.

in plausible poses. For the same reason, they lack the generative power some other architectures have.

In our implementation, illustrated in Fig. 3.5 the encoder network is composed of two fully connected layers with 200 neurons and Rectified Linear Unit (ReLU) [NH10] activations, followed by an output layer with no activation. The output layer's size is based on the number of dimensions d in which the latent representations are encoded. We empirically find that $d = 64$ yields a good balance of representation accuracy and inference speed. The decoder is the exact reversed replica and uses the same set of weights.

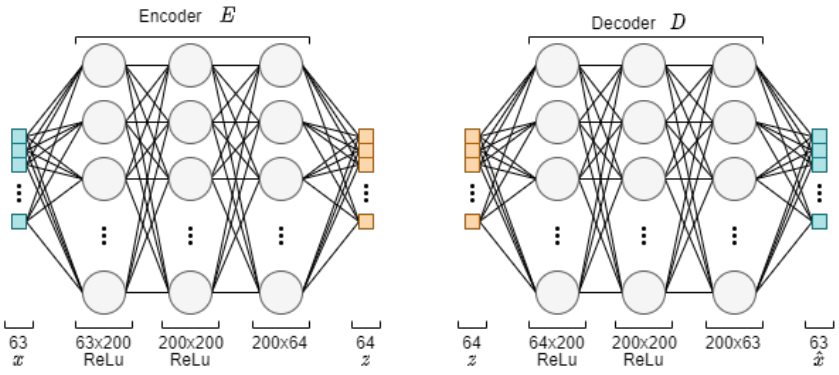


FIGURE 3.5: Architecture details of the encoder and decoder models.

The encoder and decoder's parameters, respectively θ and ϕ, are found by minimizing the squared error between poses sampled from the dataset x and their reconstructed equivalent, as shown in Eq. 3.1. Note that with our networks having a low number of units, we found no benefits to adding a sparcity penalty term to limit the number of neurons active at the same time.

$$\min_{\theta,\phi} \mathcal{L}_{ae}(\theta,\phi) = \mathop{\mathbb{E}}_{x\sim p_{data}} \left[\left\| x - D(E(x)) \right\|_2^2 \right] \tag{3.1}$$

The autoencoder is trained for 20 epochs with batches of 256 poses, using the Adam optimizer [KB15] with a learning rate of 0.0001.

3.4.2 Autoencoding Generative Adversarial Network (AEGAN)

Overview

In our second experiment, we turn to more complex architectures in order to get rid of some limitations our baseline latent space presented. More specifically, we seek to ensure that the built latent space is convex, so that any modification made to a latent pose maps back to another plausible one. Such a property would come with additional advantages in an editing setup, such as coherent interpolation and being able to generate plausible poses by drawing random samples from the known latent distribution.

Unfortunately, none of the major approaches to generative modelling using neural networks fit our specifications: autoencoders' latent spaces are unconstrained and GANs [Goo+14],only learn to generate new data similar to the training one, lacking the possibility to encode existing samples, which is required in an editing setup. VAEs [KW14] would be good candidates, however, they are known to struggle to generate highly precise samples [ZSE17].

We thus turn to a hybrid approach, similar to the one proposed by Lazarou [Laz20] to create a latent space of images in which they illustrate good interpolation properties. Their architecture, a scaffolding of autoencoders and GANs, is dubbed Autoencoding Generative Adversarial Network (AEGAN). We adapt their work to operate on pose data rather than images. The method uses four separate networks, illustrated in Fig. 3.6: an encoder E, a generator G, a pose discriminator DP and a latent vector discriminator DL. The encoder and decoder are trained to map from pose to latent space and back, and use feedback from the discriminators to shape the desired latent space.

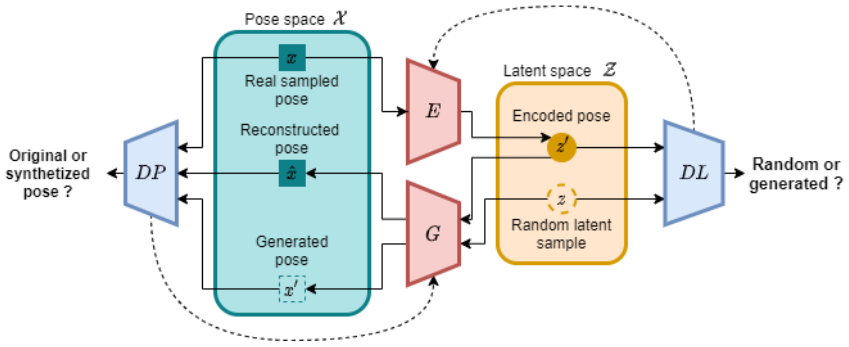


FIGURE 3.6: High level overview of the AEGAN architecture. E and G function as a traditional autoencoder, but take an additional feedback from both discriminators to enforce latent space convexity.

Networks architectures

The skeleton is a hierarchical structure made of several kinematic chains, and each joint's parameterization has an impact on other joints down the chain. In this more complex approach, we also attempt to explicitly model this spacial structure by incorporating the Structured Prediction Layers (SPL) proposed by Aksan *et al.* [AKH19] in place of fully-connected layers. SPL splits dense connections in multiple smaller layers, connected themselves following the kinematic chains of the skeleton, as illustrated in Fig. 3.7.

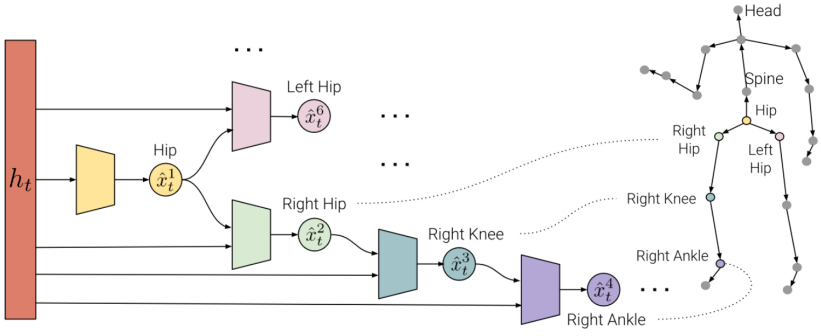


FIGURE 3.7: Overview of the Structured Prediction Layer from Aksan *et al.* [AKH19]. The embeddings of parent joints are added to the input of the embedding layer of a joint.

The encoder network is composed of a single SPL layer of 252 neurons (12 neuron per joint embedding) and ReLU activations, followed by a fully connected layer with 32 neurons and a hyperbolic tangent (*tanh*) function. Instead of its usual role of inducing nonlinearity in the network, the output *tanh* function restricts the latent vectors' values to $[-1, 1]$, so that the bounds within which a random one can be drawn and decoded in a generative fashion is known. The decoder is the opposite with a fully-connected layer with 252 neurons, ReLU activations, and an SPL layer with 63 outputs units.

The pose discriminator is the same as the encoder except for the last activation which is a Sigmoid function. The latent discriminator is a multilayer perceptron with 4 fully connected layers of 256 neurons with ReLU activations. The last activation is once again a Sigmoid function, which is used to clamp the discriminators' outputs to $[0, 1]$ so that it can be interpreted as a *"real or generated"* label.

Loss functions

The pose discriminator DP outputs a single scalar representing the probability for a given pose to be "real" over being generated. During training, its weights ψ are optimized to correctly classify raw samples from the pose dataset as well as ones generated from randomly drawn latent vectors. This is achieved by maximizing the loss function shown in Eq. 3.2.

$$\max_{\psi} \mathcal{L}_{DP}(\psi) = \mathop{\mathbb{E}}_{x \sim p_{data}} \Big[\log \Big(DP(x) \Big) \Big] + \mathop{\mathbb{E}}_{z \sim p_z} \Big[\log \Big(1 - DP(G(z)) \Big) \Big] \tag{3.2}$$

In the same manner, DL discriminates between random latent samples and encoded poses. Its parameters γ are found by maximizing Eq. 3.3.

$$\max_{\gamma} \mathcal{L}_{DL}(\gamma) = \mathop{\mathbb{E}}_{x \sim p_{data}} \Big[\log \Big(1 - DL(E(x)) \Big) \Big] + \mathop{\mathbb{E}}_{z \sim p_z} \Big[\log \Big(DL(z) \Big) \Big] \tag{3.3}$$

The convexity of the latent space is dictated by the interaction between the discriminators and the autoencoder. As G tries to fool DP, it is encouraged to turn any point from the latent space in a valid pose. At the same time the latent discriminator

encourages the encoder to encode poses into latent vectors indistinguishable from random ones, and so to use the latent space to its full extent. We therefore build a convex latent space in which the real poses are evenly spread. E and G are thus trained conjointly to achieved two objectives: to reconstruct a pose with fidelity *and* to fool the discriminators. These objectives are translated in a three-terms loss function: Eq. 3.4.

$$\min_{\theta,\phi} \mathcal{L}_{ae}(\theta,\phi) = \mathcal{L}_{recon}(\theta,\phi) + \mathcal{L}_{adv}(\theta,\phi) \tag{3.4}$$

The first term is the reconstruction loss: Eq. 3.5.

$$\min_{\theta,\phi} \mathcal{L}_{recon}(\theta,\phi) = \mathop{\mathbb{E}}_{x\sim p_{data}} \Big[\big\| x - G(E(x)) \big\|_2^2 \Big] + \mathop{\mathbb{E}}_{z\sim p_z} \Big[\big\| z - E(G(z)) \big\|_2^2 \Big] \tag{3.5}$$

It is built similarly to the simple autoencoder's one, which minimizes the reconstruction error of an encoded-then-decoded pose. However, the network must also ensure that a randomly drawn latent vector can be decoded then re-encoded to an identical one which is enforced by its second term. Note that in this form, the loss is similar to the cycle consistency loss used in CycleGAN architectures [Zhu+17], with the difference that the consistency is enforced between the known pose domain and a latent one, rather than two known image domains.

The second term is the adversarial loss (Eq. 3.6), whose terms represent the objectives with regard to the discriminators: (1) and (2) fool the pose discriminator into classifying generated and encoded-decoded poses as real ones; and (3) and (4) fool the latent discriminator into classifying encoded poses and generated-encoded poses as random latent sample.

$$\begin{aligned} \min_{\theta,\phi} \mathcal{L}_{adv}(\theta,\phi) = & \mathop{\mathbb{E}}_{z\sim p_z} \Big[\log DP\big(G(z)\big) \Big] && (1) \\ + & \mathop{\mathbb{E}}_{x\sim p_{data}} \Big[\log DP\Big(G\big(E(x)\big)\Big) \Big] && (3) \\ + & \mathop{\mathbb{E}}_{x\sim p_{data}} \Big[\log DL\big(E(x)\big) \Big] && (2) \\ + & \mathop{\mathbb{E}}_{z\sim p_z} \Big[\log DL\Big(E\big(G(z)\big)\Big) \Big] && (4) \end{aligned} \tag{3.6}$$

Training parameters

The networks are trained for 200 epochs with a batch size of 256 poses. We use the Adam optimizer [KB15] with a learning rate of 0.001 for the pose discriminator and 0.0002 for the encoder, decoder and latent discriminator. In order to regularize the performance of each networks during the training process, the encoder-decoder and latent discriminator are trained with respectively two and three times as many samples as the pose discriminator.

3.5 Comparisons

The presented architectures share the common goal of constructing a latent pose space but differ in their approach to the problem. This section discusses the stronger and weaker aspect of both.

The first difference lies in the desired shape and properties of the latent space. To observe this difference in structure, we conduct an experiment in which we encode 5000 real poses from the dataset and visualize them on a 2D plane using t-Distributed Stochastic Neighbor Embedding (t-SNE [HR02; MH08]). To evaluate the convexity of the spaces, the same amount of random latent points are drawn from a uniform distribution with the same bounds as each space. In order to give a comparison point, the real pose samples are also visualized, along with random equivalents. The results are shown in Fig. 3.8.

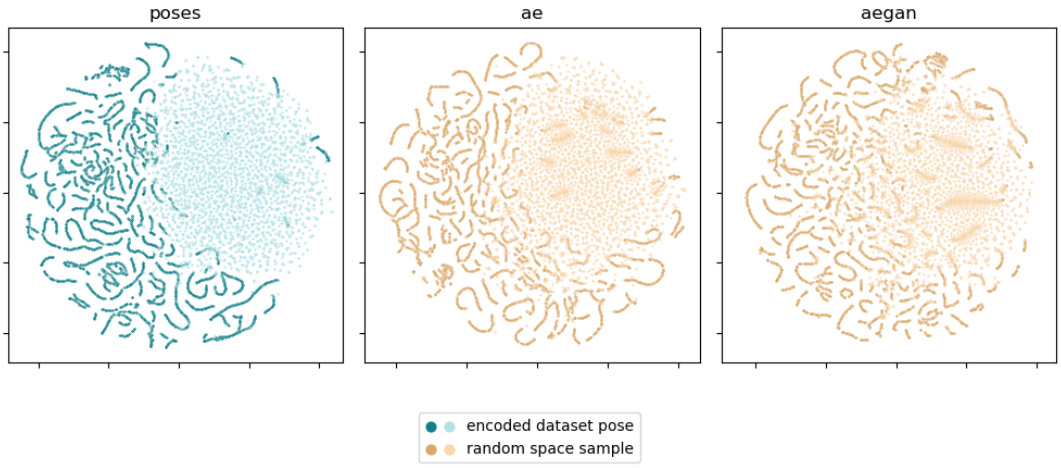


FIGURE 3.8: Visual comparison of pose spaces (original pose space, baseline autoencoder's latent space, and AEGAN's latent space). Real poses (darker) and randomly drawn data points (lighter) are embedded in 2D using t-SNE [HR02; MH08].

The experiment in the pose space illustrates the difference between the skeleton parameterization space $\mathcal{S}$ and its constrained subspace of believable poses $\mathcal{X}$. Random points are drawn from $\mathcal{S}$ and therefore do not necessarily satisfy the constraints that make them par of $\mathcal{X}$. As a result, in the visualization, they are clearly separated from encoded poses, which are part of $\mathcal{X}$.

In the autoencoder's space, the same separation is visible, although less clear. This is expected, since the model does not enforce any structure on its latent space. We can however observe that sequences of poses, forming animation clips, are still clustered together.

Finally, in AEGAN's space, the separation is blurred. While some existing poses are still encoded on the edges of the space, random and encoded poses are for the most part undifferentiated.

While considering the structural properties of the latent space only, AEGAN has a clear advantage, and it could seem like a straightforward choice for any application. However, this structures comes at a cost: the presence of multiple discriminators make training AEGAN significantly more complicated than training a baseline autoencoder.

It is necessary to balance the performance of the generator as well as those of the discriminators, so that none of them become *too* efficient at "beating" the other. Regularizing the training is a known difficulty of adversarial-based approaches, and one that has prevented their general adoption. For the same reason, the training times for AEGAN that for the autoencoder (About 4 hours for AEGAN against 15 minutes for the AE, during our experiment on the same hardware).

3.6 Conclusion

This chapter gives an overview of the general method explored in this thesis. Neural networks are used to train a latent pose space, which can then be used for pose editing. By learning functions that map existing pose samples to and from the space, the networks learn the subtleties of the constraints that a skeleton configuration must respect in order to be able to be considered an actual character pose. Two architectures are described, with different expectations regarding the latent space properties: a simple autoencoder, AE and a more complex one, AEGAN, which incorporate two discriminators to enforce the constructed latent space's structure.

Both approaches are capable of reaching their goal. They however display different advantages and drawbacks: AEGAN's latent space is smooth and convex, which means that it can be used to interpolate between samples without the risk of considering an improper pose. In contrast, AE's latent space does not have such safeguards, and exploring it requires caution. Obtaining these interesting properties come at the cost of a more complex and lengthy training phase. As a result, selecting one or the other in further applications mostly depends on the benefits of a more structured latent space in the considered use-case. In cases where such benefits are not obvious, then the simplicity of the AE might be a better solution.

CHAPTER 4

Full-body Inverse Kinematics in the latent space

4.1 Introduction

Once the latent space has been constructed in one of the two ways depicted in the previous chapter, we turn to exploring its capabilities. The first considered application is full-body Inverse Kinematics solvers.

Full-body IK is an extension of the original IK problem in which a character's skeleton in its entirety is considered, instead of a single kinematic chain. The goal is thus to find a parameterization of the skeleton where one or more joints are closest to

user-provided target positions or orientations, while respecting the pose constraints. Generic IK solvers focus on enforcing the bone lengths and (in some cases) joint orientation limits. Pose constraints are more subtle in nature, and enforcing them is the most difficult part of the task: in a real-world scenario, the constraints are hand-crafted by the animators for each new character.

This is where the latent framework proves useful: since we made sure that the decoding function produces poses similar to the training dataset, solving the problem in the latent space ensures that the skeleton constraints are respected, without having to explicitly enforce them.

This chapter discusses two methods leveraging the latent space for full-body Inverse Kinematics. The first makes uses of the constrained, convex latent space to solve user-provided constraints through optimization. The second trains a neural network to navigate the unconstrained latent space for the same purpose.

4.2 Optimization in the latent space

4.2.1 Method

In our first approach, we propose an optimization-based solution in a known latent space to solve the IK problem. The general idea, illustrated in Fig. 4.1, is to gradually modify the latent vector representing the original pose to satisfy the user-provided target. Since the latent representation lies on a convex latent space (built with the AEGAN architecture), any and all solutions considered during the optimization process maps back to a valid pose.

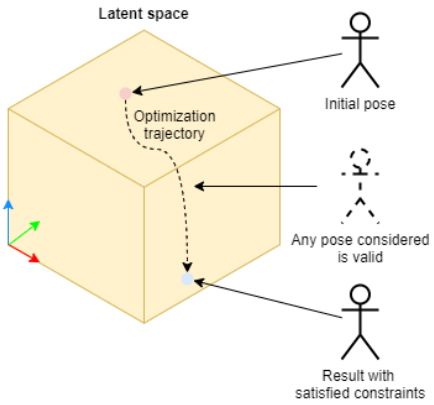


FIGURE 4.1: Optimizing in the convex latent space guaranties that any pose considered during the process will be coherent.

The full IK solver is outlined by Alg. 2. The initial configuration consists of a starting pose x and the user-provided positions for n joints targets $t_{i..n}$. The pose is projected onto the latent space, then modified by an optimizer O until the effectors $x_{i..n}$ are close enough (in terms of euclidean distance) to their respective target $t_{0..n}$, as shown in the objective function Eq. 4.1.

$$\mathcal{L}_{ik}(x,t) = \sum_{i=0}^{n} \sqrt{(x_i - j_i)^2} \tag{4.1}$$

Algorithm 2 Latent space optimization

$z \leftarrow E(x)$ % Encode
$i \leftarrow 0$
while $\lambda \geq \epsilon$ **and** $i \leq max\_iter$ **do**
  $i \leftarrow i + 1$
  $z \leftarrow O(z)$ % Optimize
  $x' \leftarrow D(z)$ % Decode
  $\lambda \leftarrow \mathcal{L}_{ik}(x', t)$
end while
return x'

To ensure constant maximum time, the process can be stopped when a maximum number of iteration (max_iter) is reached. In this setting, our contribution is more largely the construction of a smooth latent space, in which many existing optimization methods can be used. In the later experiments O uses stochastic gradient descent (SGD), but could easily be swapped for another.

4.2.2 Results

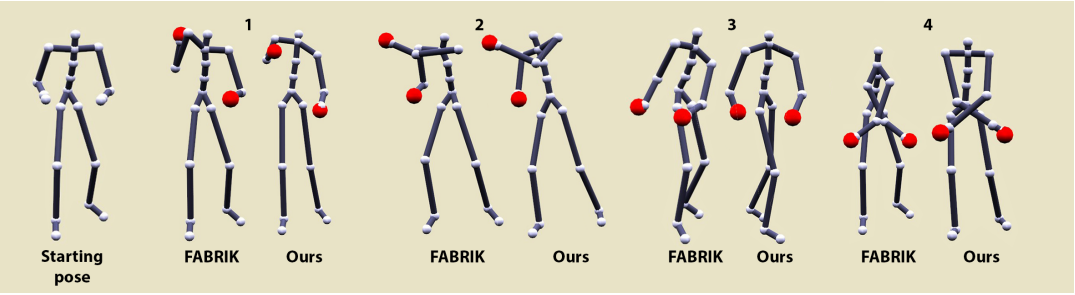


FIGURE 4.2: Examples of pose edition using latent optimization and FABRIK [AL11]. Targets are shown in red.

In order to evaluate the results of our method we integrate our solver in an example posing software and compare its outcome with a comparable, non-neural method: FABRIK [AL11]. We pick FABRIK for the traits that make it a popular IK solver: its simplicity and fast convergence times. We implement a full-body human skeleton solver as described in [ACL16] but stay as close as possible to our method setup process by not manually implementing any joint orient constraints. Figure 4.2 shows a comparison of the poses obtained by latent optimization and FABRIK with the same targets set.

FABRIK working on kinematics chains with no prior on the human skeleton, it may end up with unrealistic poses, whereas our optimization process exploring the latent space results in poses satisfying the constraints without breaking the implicit skeleton rules. Just like with the other IK methods, the distance between limbs is constant but additionally self-occlusion is avoided, and the poses appear natural. In (1) and (2) the skeleton leans on one side in order to reach a target above its shoulder, giving way to its arm. The legs also move slightly to appear in balance. In (3), our method makes the skeleton twists it upper body to face the two targets on its side. (4) illustrates the limits of FABRIK without specifying many constraints: when trying to reach targets on opposite sides, the shoulders move forward unnaturally. With our methods, the torso structure is implicitly learned, and the algorithm finds a more suitable solution.

While the final output of the latent optimization-based IK solver are good, it does not completely fill our expectations. The main limiting factor is the execution time: even when operating on a smooth space, the optimization is an iterative process which is not guaranteed to find a satisfying solution within a bound time. In practice and using our baseline implementations, solving a single pose takes up to half a second, which is not enough to allow a user to interactively manipulate the skeleton.

4.3 Learning full-body inverse kinematics solvers

4.3.1 Method

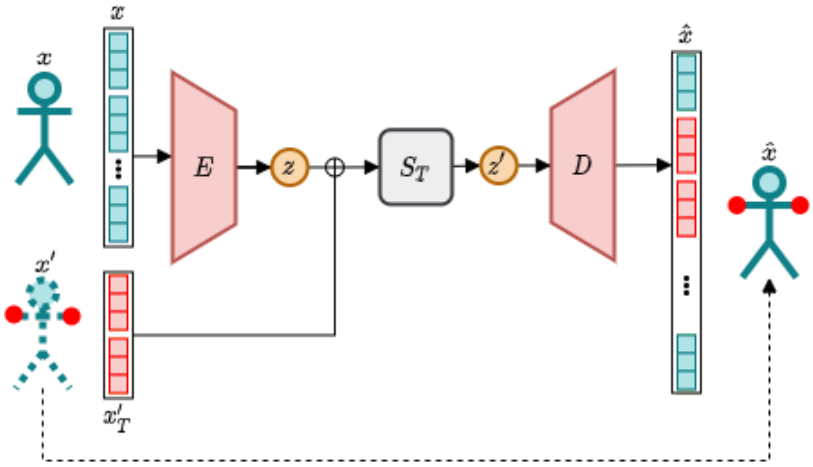


FIGURE 4.3: High level overview of the generation setup. The target joint's positions (red) are matched as closely as possible, while the other joints (green) should be as close as possible to the starting pose. The loss is computed as the difference between the generated pose and the target pose.

In our second approach, we propose to train another neural network to solve the IK problem in the latent space, replacing the optimization process. Since neural networks are capable of efficiently modeling complex data distributions, enforcing the smoothness of the latent space is not as beneficial as with the optimization-based approach. This second method therefore uses the latent space built by a simple autoencoder. This choice is additionally motivated by our goal for usability, as the training of an autoencoder is way simpler than that of AEGAN.

The high-level idea of the method is illustrated in Fig. 4.3: the autoencoder builds a latent pose space, and the solver model operates on this space to solve the full-body IK problem. In this section we first present the architecture and training of the solver network, then describe a methodology using multiple instances of the solver model at once to work with a varying amount of targets.

4.3.2 Pose solver network

An instance of the solver model S_T (along with its parameters set β_T) is specialized to solve the IK problem for a set of specific targets $T = \{t_0, \ldots, t_n\}$. It is trained to generate a new pose from an input pose and the desired targets locations. As

it operates on the latent space built by the autoencoder, it more precisely accepts and outputs a latent pose vector, i.e. with x'_T the concatenated target positions, $\hat{z} = S_T(z \oplus x'_T)$. While most solvers will only be concerned with a single joint, some situations might benefit from having multiple targets linked together (for example, if attempting to move both hands in contact of an object).

The network is composed of three fully connected layers with 126 neurons and ReLU activations, and an output layer whose dimensions match the latent space's.

During training, we randomly sample an input pose x from the dataset and feed it to the network. We also sample a second pose x' to use as a target. Since the network will be used in an interactive fashion by the user, with at least one operation per frame, it is expected that the differences between the original positions of the effectors and their target to be relatively small. To reflect this, the target x' is sampled from the same source clip as x, among one of its 20 frames of the input. This association thus serves two purposes: first, ensuring that the target pose is reachable from the starting one, and second, restricting the difference between any two poses seen during training from being too wide.

The network's weights β_T are optimized to minimize the loss function in Eq.4.2 designed to represent its high level objective: reaching the targets with the associated joints while retaining a realistic pose. We guide the network toward this objective by using a slightly modified mean squared error function, separating each pose x in two sets of joints: x_T the joints associated to the targets, and x_r the others. We introduce a constant k to give more relative importance to the target term of the function, so that the non-targets joints of x' are only used to nudge the final result toward a plausible pose. In our experiments k is set to 0.01. The idea behind this trick is that the target pose sampled from the dataset is not an absolute truth to be reached at all cost, but should rather be considered as a guide, the main goal being reaching the targets.

$$\begin{aligned} \hat{x} &= D\Big(S\big(E(x) \oplus x'_T\big)\Big) \\ \min_{\beta_T} \mathcal{L}_{s_T}(\beta_T) &= \mathop{\mathbb{E}}_{(x;x')\sim p_{data}} \Big[\|\hat{x}_T - x'_T\|_2^2 + k \cdot \|\hat{x}_r - x'_r\|_2^2\Big] \end{aligned} \tag{4.2}$$

In our experiments, an instance of the solver model is trained for 5 epochs with the Adam optimizer [KB15] using a learning rate of 0.0001 and a batch size of 256.

4.3.3 Solving other targets configurations

The solvers as presented here are designed to edit a pose while respecting only a few targets at once (one to two in our examples). In a real-world scenario however, a user might want to manipulate a character through different effectors. In cases where the effectors are used one after the other, it is possible to keep the weights set of different solvers at hand, and to switch to the corresponding one when required. Scenarios in which multiple effector targets change position during the same frame are rare in an interactive format, but can be envisaged in cases where the targets are moved in one step, and resolved in another. Multiple solvers might also be required to "pin down" a joint while moving another. In these cases, another strategy is necessary.

We propose to handle these cases by running multiple solver instances in sequence, each one using the output of its predecessor as input, i.e. for three different solvers S_{T1}, S_{T2}, S_{T3}, each specialized for a target group T_i, the resulting modified latent pose is given by $\hat{z} = (S_{T3} \circ S_{T2} \circ S_{T1})(z)$. Since the embeddings functions are separate from the solvers, this also represents an efficiency gain, as the encoding and decoding only needs to be run once per frame.

4.3.4 Results

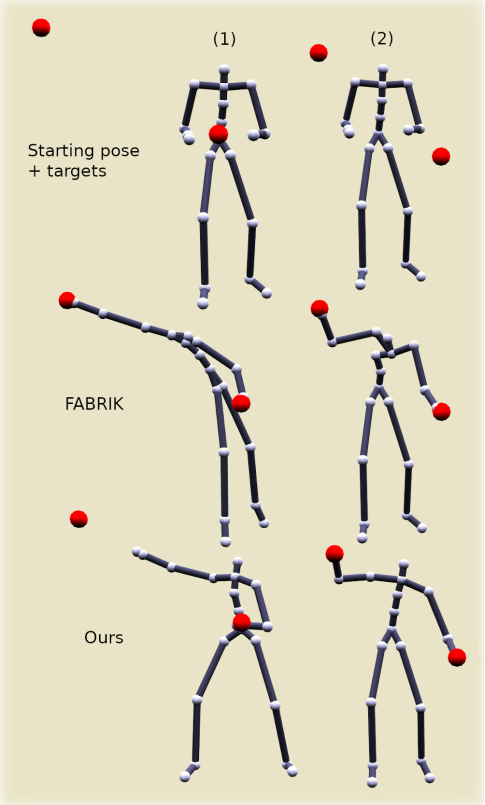


FIGURE 4.4: Starting from a pose and targets for two joints, an IK solver like FABRIK (middle) generates less realistic poses than our neural solver (bottom).

Fig. 4.4 showcases an example of how our method can be used to edit a pose by moving the targets around. In this case, a single solver with the targets set to both skeleton's hands is used. We once again compare against FABRIK in the same setup as the previous section.

Our solver yields poses satisfying the constraints without breaking the implicit skeleton rules: the distance between limbs is constant, self-occlusion is avoided, and the poses appear natural. The side-by-side comparisons with FABRIK's results highlight the limits of working on kinematics chains with no prior on the human skeleton.

Example (1) illustrates how the targets are used as guides rather than fixed, unbreakable rules. While FABRIK extends the full body, our solver generates a new pose where the torso is slightly twisted towards the right-hand target while the legs are spread to mimic maintaining balance. Even though our method is aimed toward beginner animators, experienced ones could also find it useful. It could for example be used as a fast prototyping tool to flesh out the pose, while switching to more accurate and manipulation-heavy tools to focus on the details later on.

Some additional results of the neural IK solver, used in an interactive context, can be seen in the accompanying video of one of our papers: https://www.youtube.com/watch?v=h8nzECEtw3c.

Combined solvers

Figure 4.5 demonstrates an example with the multi-solver setup described in 4.3.3. In this example three solvers are used at once: one for the two hands, one for the two ankles and one for the head. Compared to the FABRIK result, our method yields a plausible pose: the skeleton is bent down to meet the head target, but the general orientation of the pose is kept intact. The limbs also retain some sort of curvature rather than fully extending unnaturally. Here again some targets are not strictly reached, as the pose generated by earlier solvers in the chain are modified by the others further down, but the guidance provided by the targets is respected. This setup also incurs slightly slower runtimes (see Table 4.1) but is still faster than FABRIK.

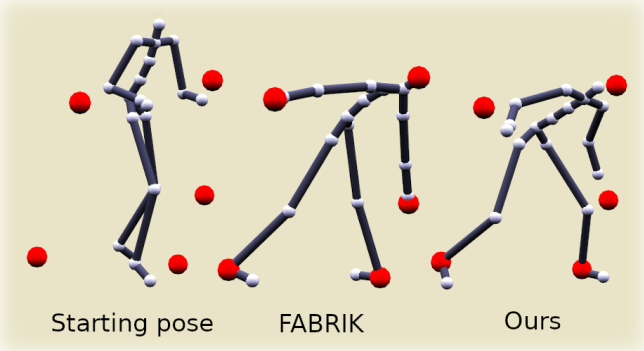


FIGURE 4.5: Sample results solving multiple targets with a sequence of neural solvers (S_{hands}, S_{feet}, and S_{head}). Targets are shown in red.

Run times

At run-time, the complexity of the solver is fixed and regardless of the targets' positions, a single pass through the networks, which can be seen as just a few matrix multiplications, is enough to produce a result pose. This property coupled with the relatively small size of the networks allow for a fast solving process, as highlighted in table 4.1.

Method	Memory footprint (kB)	Runtime (ms)**
FABRIK (2)	-	6.56
Ours (2)	442	1.47 (3.03*)
FABRIK (5)	-	6.74
Ours (5)	826	3.36 (4.58*)

*With post-processing
**Average over 1000 random iterations

TABLE 4.1: Comparative numeric results of the neural and FABRIK solvers with two and five end-effectors (using the combined solver method). All experiments are run on a single CPU thread.

Compared to other data-driven pose methods, the computing-heavy part of our process is done once at training time. Even so, the training itself is kept short thanks to the

modest size of the networks: around an hour for the auto-encoder and 15 minutes for the solvers, on a single GPU.

Memory footprint

An advantage of neural networks is the low memory footprint they hold. While other data-driven pose design methods require a reference to the pose database (or a compressed version of it) to be kept in memory, neural networks only require their trained weights. These can be quite heavy as well in the case of large models, but as ours are quite small, so are their weights. As a comparison point, the Gaussian Processes-based method NAT-IK [WTR11] discloses a 30 MB memory footprint while our full-body solver only takes up 826 kB (including the weights for the autoencoder and each solver).

Comparison with other pose edition approaches

Huang et al. [Hua+17] proposed a general comparison chart for full-body IK methods, ranking common approaches by speed and subjective quality. Adding our solution to the chart (Fig. 4.6) highlights the useful spot it fills by striking a good balance between speed and accuracy. To the best of our knowledge, this work presents the first method leveraging neural networks for pose edition. It stands apart from previous learning-based approaches as the first one to combine real-time edition speed with fully learned skeleton constraints. In comparison, NAT-IK [Hua+17] uses soft learned constraints but still requires explicit, manual ones to be set. Style IK [WTR11] does not, but the poses are not generated in real time.

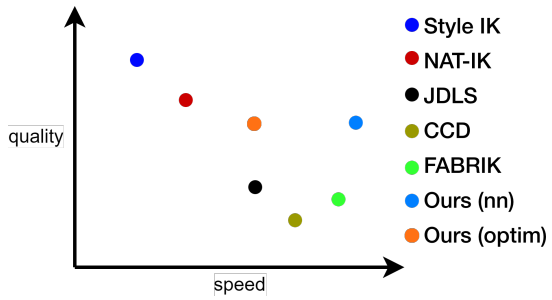


FIGURE 4.6: General comparison of various full-body IK methods in terms of speed and quality. Style IK [WTR11], NAT-IK [Hua+17], JDLS [BK05], CCD [WC91] FABRIK [AL11]

4.4 Conclusion

This chapter presented two different approaches sharing a common goal: to leverage the latent pose space in the context of a full-body IK solver. In the first, latent poses representation are optimized, taking advantage of the enforced properties of AEGAN's latent space to restrict the search space. In the second, another neural networks learns to navigate the unrestricted latent space built by an autoencoder.

Both methods show similarly satisfactory results, and enable the user to easily manipulate a character skeleton. Learning from a large dataset of ground truth poses allows us to avoid manually specifying the complex constraints of the human skeleton, and to only generate plausible poses.

The methods however differ in efficiency. The optimization-based solution suffers from unbound solve times, and as a result is not suited for interactive editing. The other runs in real time thanks to the efficient inference of neural networks, at the cost of an additional training step.

The work presented in this paper was published in two conference communications:

- Léon Victor, Alexandre Meyer. "Character Pose Design in Latent Space For Animation Edition". Journées Françaises de l'Informatique Graphique 2020, Nov 2020, Nancy, France. hal.archives-ouvertes.fr/hal-03338910
- Léon Victor, Alexandre Meyer, Saïda Bouakaz. "Learning-based pose edition for efficient and interactive design". Computer Animation and Virtual Worlds. 2021; 32:e2013. doi: 10.1002/cav.2013

Following the communication of our results, multiple similar approaches have been published (see Voletis *et al.* [Vol+22] and Bensadoun *et al.* [Ben+22] for example), supporting our intuition that using neural networks to power user-centric creative interfaces is an interesting venue for further research.

CHAPTER 5

Learned pose manipulators for animation style editing

5.1 Introduction

Approaching animation creation under the style metaphor is an interesting idea, with the potential to facilitate the production or modification of animations through the use of higher-level parameters than raw skeleton parameterization. However and as our review of the literature highlights, most of the previous efforts dedicated to this kind of interface focus on applying the style of an animation to another. In cases where style parameters are exposed to the user, the characteristics linked to the available style categories are extracted from hand-labeled data, which takes control away from the user who might have a different expectation for a given style.

This chapter presents a method which allow users to edit a character pose using "style" constraints, by leveraging the latent pose space presented in an earlier chapter. We first discuss the limits of the commonly used approaches to motion style editing, then describe the objective pose parameters we use in place of the usual high-level style labels, and a method to learn to enforce them. Finally, we propose a method to propagate style changes made to a pose to neighboring frames to facilitate the editing of whole animation clips.

5.2 Limits of existing style editing techniques

If the definition of *style* as a separation from an animation's *content* is commonly accepted in the computer animation literature, the limits of what the term itself entails are not clearly defined themselves. Some associate an animation's style to emotions experienced by the character, such as "joy", "anger" or "sadness" [FP14]. Some link it to existing motion analysis systems, such as Laban's [LU71; Dur+16], or to psychological concepts such as the arousal/valence scale [EF67; Ran+19]. Some others use arbitrary categories, such as "gorilla", "childish", "zombie", or "neutral" [Xia+15].

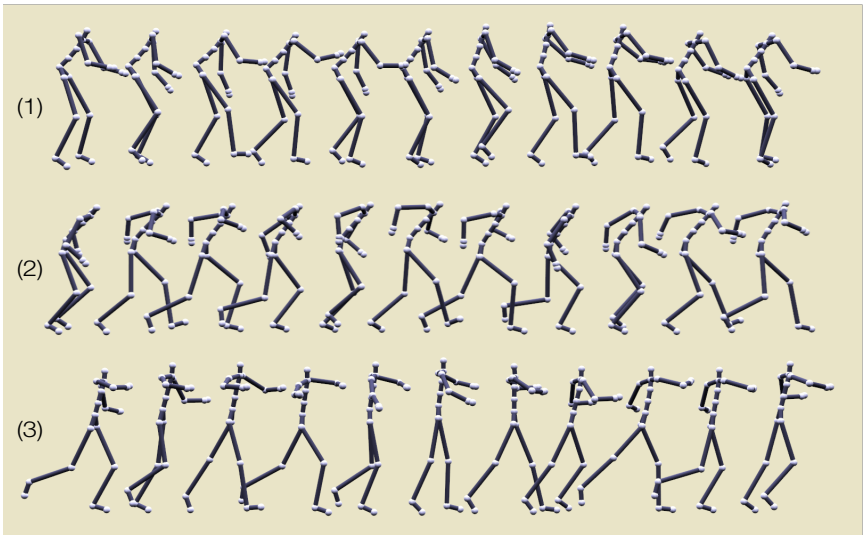


FIGURE 5.1: Examples of walk animations in exaggerated styles from existing motion databases [Hol+15; Xia+15]. (1) "zombie"; (2) "old man"; (3) "childish". Distinguishing each style with no context is not straight-forward.

This categorization might however be too subjective to be considered as an expressive parameter to provide to users. Indeed, various factors can introduce bias in style categories, leading editing tools based on them to fail to meet the user's expectation.

A first source of bias lies in how the data used to calibrate the style categories is recorded. It is usually not captured in the wild, but rather played by actors [Xia+15]. The sequences are thus often overplayed, exaggerated representations of the actor and director's visions of the categories (see Fig. 5.1). In the most grounded acquisition procedures, emotions states are elicited in the subjects by exposing them to pre-determined scenario, designed to provoke a specific reaction [FP14]. Eliciting emotions is however not a fail-proof process: A subject's personality and individual experience might influence their reaction when being exposed to a prepared material. Even in

these settings where precautions are taken, the bias is not totally avoided [MWE14; HHN10].

A second source of bias appears when considering differences of expectation regarding a style's characteristics between a future user and the authors of style categories. It is most evident for "fantasy" styles representing a shared cultural reference. For example, the "zombie" style from Xia *et al.*'s database (Fig. 5.1) depicts a slow, disjointed walk. It is evidently a common depiction of the zombie figure in popular culture, however, it is not the only one: users editing animation using this style category might be surprised if they expected a fast and sleek zombie. The problem also appears when considering categories representing a social trait or behavior ("proud", "strutting", "childish"), built from stereotypical representations which are not universally shared. Notably, the universality of emotions is still actively discussed in other fields, with evidence pointing to the fact that the way we perceive and express emotions is influenced by our cultural specificities [EA02; Bar+19]. In the computer animation field, early research on motion style acknowledged this limitation [ABC96], but following work lost sight of the nuance and often consider categories pertaining to emotional status as universal. We should remain vigilant when designing tools based on these potentially biased assessments.

One other drawback of existing style editing approaches is that it is not possible to create a new style from the ground up, as the style must be identified and present in an existing clip. In the best ones, the style can be extracted from a single existing stylized clip and applied to others, but it nevertheless needs to exist in the first place. In a real-world scenario this might not always possible, as animators might want to design a new style on the fly, and to choose themselves the aspects of the motion that define its characteristics.

5.3 Pose style parameters

5.3.1 Definition

Even with the limits of style-based approaches in mind, being able to edit an animation sequence by manipulating higher-level parameters is a desirable goal. We thus propose an alternative solution which focuses on leaving the task of defining the characteristics of a style to the user. The general idea is to introduce higher-level pose parameters that can be directly computed from the pose data. Since these are objective, we do not make assumptions on the expectations of users with regard to a style. Instead, they are able to combine multiple of these parameters to fit their own view of a style.

For the sake of an example, imagine an artist editing an animation sequence of a character walking, expressing "cold rage". In his mind, the character should be taking short, fast steps, his arms tense, stretched toward the floor. With previous style edition methods, they might be able to hit an "angry" button, or maybe use a slider to determine how angry the result should be. Unfortunately, it might be that the system was calibrated with another definition of angry in mind: the character now walks faster, the arms bent and swinging, with the fists close to the torso. Using the pose parameters would allow the user to circumvent this problem. By combining a few ones such as the angle formed by the elbows, the distance between the feet, or between the hands and the floor, and editing the animation through them, we can avoid relying on assumed style characteristics.

5.3.2 Experimental parameters

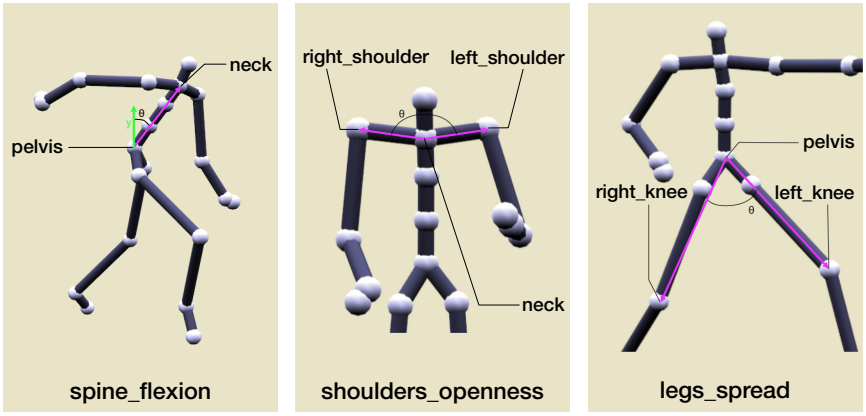


FIGURE 5.2: Visual depiction of the three pose parameters used in this paper's examples.

In order to demonstrate the application of our system, the following experiments use three sample metrics applicable to a human skeleton: *spine_flexion*, *shoulders_openness* and *legs_spread*. All three are illustrated in Fig. 5.2. In a real-world scenario, users could introduce other parameters as they see fit, as long as they can provide a function to calculate them from pose data.

All three represent the angle formed at the intersection of two vectors $\vec{u}$ and $\vec{v}$ obtained from the pose. Further equations use the $\theta(\vec{u}, \vec{v})$ function in Eq. 5.1 to compute the angle at their intersection.

$$\theta(\vec{u}, \vec{v}) = \arccos \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \cdot \|\vec{v}\|} \tag{5.1}$$

The *spine_flexion* (SF) parameter represents the angle between the world "up" vector $\vec{y}$ and the skeleton's spine axis. It is computed as shown in Eq. 5.2. It could be used to bend the character forward when in a hurry, or backward when relaxed.

$$SF(x) = \theta(x_{neck} - x_{pelvis},\ \vec{y}) \tag{5.2}$$

Next the metric *shoulders_openness* (SO, see Eq. 5.3) describes how open the character's torso/shoulders area is. Example usages might be to raise the character shoulders to denote anxiety or shyness, and to open up to denote a more relaxed or proud behavior.

$$SO(x) = \theta(x_{spine1} - x_{rshoulder}, x_{lshoulder} - x_{spine1}) \tag{5.3}$$

Finally, *legs_spread* (LS, see Eq. 5.4) denotes how close the character's knees are to each other. Possible usages include restricting walking strides, or tweaking whether the character appears in balance.

$$LS(x) = \theta(x_{pelvis} - x_{rknee}, x_{lknee} - x_{pelvis}) \tag{5.4}$$

5.4 Editing a pose through style parameters

With pose style parameters defined and some example ones laid out, we can turn to actually using them to edit a pose. The idea is similar to the Inverse Kinematics solver discussed in the previous chapter: train a neural network to generate a new latent pose from a starting one z and a target value p for P a specific pose parameter : $\hat{z} = M_P(z \oplus p)$. An instance of the network is specialized for a single pose parameter, and it operates in the latent pose space to implicitly handle skeletal constraints.

To reflect this objective during training, we sample two poses from the same original animation sequence with an offset of n frames. Sampling from a neighboring frame prevents the target pose from being too different from the starting one. This ensures that the network learns to *edit* a pose rather than to "dream up" an unrelated one matching the target pose parameter. In our experiments, n is sampled randomly for each iteration in $n \in [-10, 10]$. As we target an interactive edition context, aiming for smaller changes in metric is reasonable. Gathered together, the parameter network's weight set β_P are found by minimizing Eq. 5.5.

$$\min_{\beta_P} \mathcal{L}_{M_P}(\beta_P) = \mathop{\mathbb{E}}_{(x;x')\sim x_{data}} \left[\left\| x' - D\Big(M\big(E(x) \oplus P(x')\big)\Big) \right\|_2^2 \right] \tag{5.5}$$

Keeping with the idea of smaller networks we opt for a two-layers perceptron with 126 hidden units per layer and ReLU activation. The input layer's number of units is equal to the size of the auto-encoder's latent dimension plus one (to account for the parameter value). The output is the same size as the latent dimension. We empirically find that this architecture suits our needs, larger/deeper networks yielding no perceptible value.

Training parameters. Each pose parameter network is trained using the Adam optimizer [KB15] with a learning rate of 0.0001, and batches of 1024 pose pairs until convergence, which takes around 15 minutes on an NVIDIA Quadro T1000.

5.5 Extending to edit animation clips

The editing pipeline presented in this thesis up until this point only operates on a single pose at once. In a real-world scenario however animators typically edit a specific key frame, but expect their changes to be gradually applied to the neighboring ones. To meet these expectations we propose a method to edit an entire animation sequence through a single set of targets, composed of the user's choice of pose parameters and/or IK targets.

The proposed approach (Fig. 5.3) resembles the traditional animation curve paradigm, using a user-provided weight curve W to control how much each key pose will be impacted by a modification.

To edit an animation A, the user selects a key frame A_n and provides their desired target value for each parameter: $P_{0..i}$ as well as a weight curve W. For each frame A_t in the sequence, a latent representation z_t is encoded. Each of them is then processed by the pose metric module, resulting in an intermediary latent pose $\overline{z}_t$. The final modified latent pose $\hat{z}_t$ is obtained by blending z_t and $\overline{z}_t$ using the frame's weight factor $W(t)$ (Eq. 5.6).

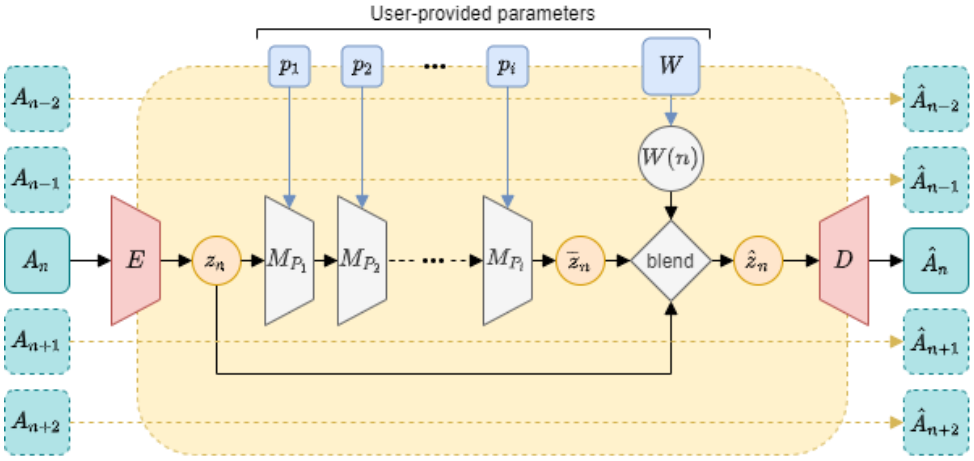


FIGURE 5.3: Visual overview of the proposed method to echo modifications made to a key pose A_n. The pose is encoded then sequentially modified by the pose parameter networks $M_{P_{0..i}}$ to accommodate the parameter targets $p_{0..i}$ provided by the user. The resulting latent tensor is then blended with the original one by a per-frame factor $W(n)$, then finally decoded. By repeating this operation for each neighboring pose according to the weight curve, the full clip is modified with regard to the provided targets.

$$
\begin{aligned}
z_t &= E(A_t) \\
\overline{z}_t &= M_P(z_t) \\
\hat{z}_t &= (1 - W(t)) \cdot z_t + W(t) \cdot \overline{z}_t \\
\hat{A}_t &= D(\hat{z}_t)
\end{aligned}
\tag{5.6}
$$

In our experiments W is a hat function starting at 0 at the clip's extremities and linearly increasing upon reaching 1 at the selected frame. This weight function is however an external parameter: experiments with a sinusoidal yield comparable results, and the curve could be edited by animators to fit their expectations, much like existing animation curves. We also note that each one of the clip's poses is processed independently of the others. The edition can thus happen in parallel, maintaining real time interactivity.

5.6 Results

5.6.1 Pose edition

In Fig. 5.4, a single pose is modified by a user by providing gradually increasing target *spine_flexion* parameter values. The results respect the desired changes as shown by the measured values below each pose. The resulting poses also illustrate the interesting properties of the latent pose space. At every iteration, the skeleton's arms

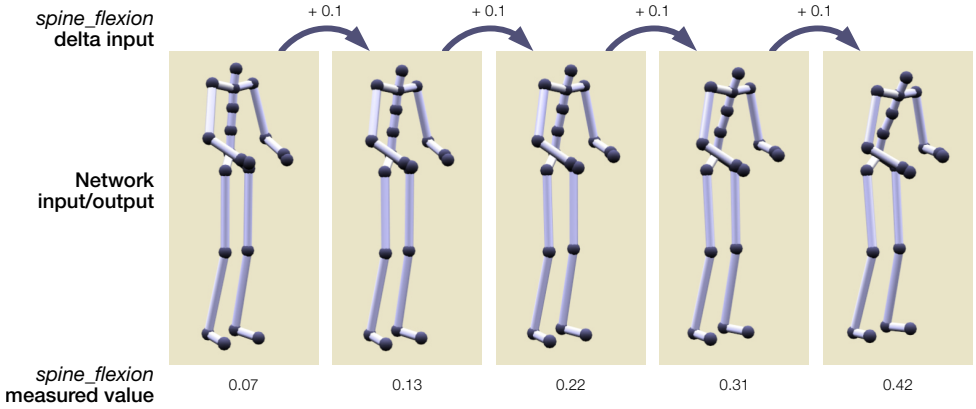


FIGURE 5.4: Incremental modification of a single pose by providing a target *spine_flexion* input. On each step, the user's input is a 0.1 increase over the computed parameter value.

are slightly raised, and its knees bent, to compensate the loss of balance induced by the flexion of the spine.

5.6.2 Using multiple modules

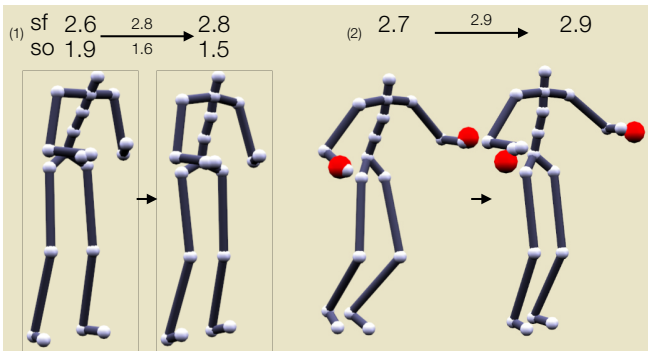


FIGURE 5.5: Sample results of the pipeline using multiple modules. In (1), *shoulders_openness* (so) and *spine_flexion* (sf) are used simultaneously. In (2) the IK solver from the previous chapter is used along the *spine_flexion* parameter manipulator.

Fig. 5.5 presents some results obtained by running multiple modules in sequence. The first example shows the result of modifying the *spine_flexion* and *shoulders_openness* metrics at the same time. Both modifications can be seen on the skeleton: its spine is straightened, and its shoulders lowered. The rest of the pose is also slightly edited to accommodate for those change: the hands are lowered, and knees bent a little further.

The second one shows the result of using the *spine_flexion* parameter module along with the IK solver for the two hands presented in the previous chapter. The method allows us to find a compromise between each provided constraint. Here, the skeleton's back is straightened, but the hands are lowered to make up for the difference, still reaching for the targets.

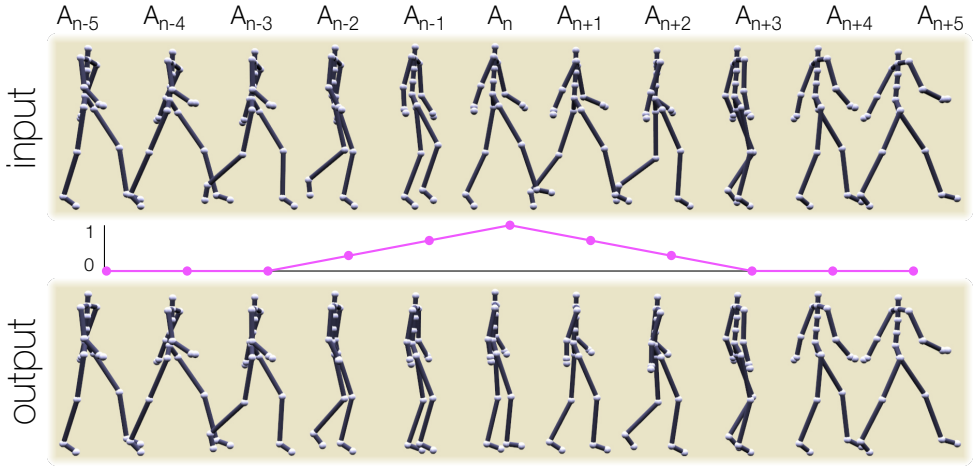


FIGURE 5.6: Editing an animation clip according to the user input. The output is produced by the pipeline given a selected frame A_n, a target value for the *legs_spread* metric, and a weight curve. At the selected frame, the character's legs are closer to each other. The rest of the clip is less impacted, and the overall walking motion is preserved.

5.6.3 Animation edition

Fig. 5.6 showcases the result of an animation being edited using the animation pipeline. The user selects a frame A_n, provides a target *leg_spread* parameter value and an interpolation factor curve, a hat function peaking at one at A_n and reaching down to 0 on both sides at A_{n-3} and A_{n+3}. The remaining frames use a factor of zero.

Each frame pose is processed separately by the network, and the result is given by interpolating the latent representations of the input and output poses, using the frame-specific factor.

As a result, the output animation $\hat{A}$ is modified, with the impact being most important on the selected frame and gradually lowering on each side. At frame n the output pose is modified the most, and the reduced legs angle is very visible. At frames $n-2$ and $n+2$ this impact is barely visible, while further frames are not modified at all.

Interpolating between original and modified latent poses allow the whole sequence to keep its temporal coherence even though the pipeline processes one frame at a time, with no knowledge of the past and future frame context. We also take advantage of the properties of latent spaces: any latent interpolation result is mapped back to the real pose space, thus avoiding artifacts such as skeleton interpenetration.

5.7 Conclusion

The different approaches to animation style editing in the literature often rely on style categories, which suffer from under-considered biases and as such, might be too subjective to be used as-is in editing tools. Expert animators often want to tweak animation clips in a fine and precise manner to fit their vision of a character's motion. We believe that editing tools should seek to support them with controllable tools. In this chapter an alternative approach was described, making use of objective pose parameters computed on a single pose. This approach leaves the control to users, who can define parameters set which fit their *own* perception of styles. The proposed

method once again leverages the latent pose space used throughout the thesis to automatically enforce implicit and subtle constraints. We also describe a pipeline in which multiple neural networks act as real-time pose manipulators.

The main shortcoming of this approach is that it is limited to a single standardized humanoid skeleton, with which the networks are trained. This is a common problem with data-driven methods: in order to adapt to a new skeleton, the network would need to be retrained. This could be difficult if access to existing animation data for the new skeleton is limited, but recent progress in animation retargeting [Abe+20] could help by allowing more accessible data augmentation. The modularity of the parameters helps here as well, as different skeleton will require a new set of parameters.

CHAPTER 6

Conclusion

6.1 Contributions summary

Virtual characters play important roles in most computer-generated media, and synthesizing then editing motion data to animate them is a crucial task in production processes. The work presented in this thesis follows a wide array of research efforts which aim to make computer animation always more detailed and expressive, all the while managing the complexity of the creative pipelines.

In recent years, deep learning methods have started making waves in the field, leveraging the impressive data modelling capabilities of neural networks and the growing availability of Motion Capture data to produce new, unseen motion. Applications considered by previous work have however remained quite high-level, generating full motion clips in one go and leaving little room for artistic control. This thesis therefore proposes to explore this blind spot, by applying neural networks to a more restricted task, pose editing, and embedding the resulting models in interactive editing tools that leave control to the artists.

Our method relies on the construction of a latent pose space, on which existing poses can be projected then edited. A large part of the problems encountered while editing a character pose can be attributed to the subtle constraints a character's skeleton's parameterization must respect in order to be considered as a plausible pose. With this in mind, the space is structured to ensure any of its sample represent a valid pose. This condition is handled by neural networks trained to map existing poses to and from the latent space.

In chapter 3, we describe the construction of this space, by detailing two different approaches. The first relies on a simple autoencoder, and results in a space of unspecified shape. To the contrary, in the second, an adversarial method is employed to shape the resulting space and enforce its convexity, providing it with smooth

interpolation properties. Both spaces form the base upon which the pose editing methods described later build.

Chapter 4 is dedicated to our first pose editing problem: full-body Inverse Kinematics. In a first section we propose an optimization-based solution, taking advantage of the smooth nature of the latent space to speed up the process by only considering valid latent poses. This approach yields appealing results, however, its convergence time is not bound, and as such it is not ideally suited for interactive applications. In a second section we develop another solution to the same problem, this time training another neural network to explore an unconstrained latent space. Once trained, the method runs in real-time and yields high-quality results, allowing it to be used as an interactive tool to manipulate a character's skeleton as a puppet.

Finally, in chapter 5, we study the existing methods that propose to edit character poses by manipulating their *style*, and argue that the concept, as currently applied in computer animation, might build upon biased point of views. We argue for an alternative approach which allow users to compose their own definition of a style by combining objective pose parameters. We propose a system in which the pose parameters are edited in the latent space.

Overall, the work presented in this thesis makes a case for user-centered applications of neural networks in animation. Deep learning approach have already proven to capable of modelling large amounts of data, and generating impressive and complete character motion. However, we believe that their learning power and fast inference speeds make them good candidates to be used in real-time interaction with a user. We hope this document can inspire future work in this direction, and that future animators can rely on data-powered tools to put their idea in motion.

The work presented in this thesis has also been published as part of several papers:

- Léon Victor, Alexandre Meyer. "Character Pose Design in Latent Space For Animation Edition". Journées Françaises de l'Informatique Graphique 2020, Nov 2020, Nancy, France. hal.archives-ouvertes.fr/hal-03338910
- Léon Victor, Alexandre Meyer, Saïda Bouakaz. "Learning-based pose edition for efficient and interactive design". Computer Animation and Virtual Worlds. 2021; 32:e2013. doi: 10.1002/cav.2013
- Léon Victor, Alexandre Meyer, Saïda Bouakaz. "Pose Metrics: a New Paradigm for Character Motion Edition". Preprint. https://arxiv.org/abs/2301.06514

6.2 Limitations & perspectives

This thesis' contributions are a first step towards user-controllable, data-driven animation editing tools, focusing on pose editing. The application is however still new and multiple problems still stand out.

Mainly, in this thesis, the pose editing task is considered as a stand-alone process. However, in real scenarios, single poses are always part of a sequence, and form an animation clip. A lot of information, related to velocity, pace, or motion amplitude for example, is lost when separating pose data from its context. We propose an approach to account for the dynamic nature of motion in 5, but the training is still done on pose data alone. Providing neural network models with more of this context might be beneficial, even for tasks related to pose edition. This is especially the case for our

style parameters based method, as many of the existing motion descriptors from the literature are not applicable to single poses. An interesting perspective will then be to adapt the neural networks models so that they can extract information from motion sequences, and apply them in contexts similar to ours.

In the same vein, the networks used in this thesis are deliberately kept simple. As shown in the literature review, model architectures specifically dedicated to animation data are currently a heavily studied topic, so applying these modern approaches to our tasks could significantly improve the quality of the results.

Next, our approach is limited to a standardized skeleton. In actual production, each character often has its own, and existing data might not even be available. Future work should focus on alleviating this limit by reusing trained networks on different skeleton topologies. Improvement in motion retargeting might also help to adapt existing motion data to a new skeleton.

This thesis' contributions are still in an early shape, and have not yet been confronted to actual animators. An interesting perspective is thus to lead user-studies to gather feedback on the usability and efficiency of the methods. End-users consideration would help identify more limitations as well as more venue for improvements.

Data-oriented animation is already widely used in the industry, thanks to the increased availability of MoCap technologies. Deep learning methods however, while receiving a lot of attention from the research community, have yet to break in production setups. Leveraging their power in the context of user-centric tools, in which the software facilitates the animators work while leaving them in control, might be the key to a widespread usage. We hope our work inspires more research towards similar goals, empowering animators in their creative process and helping to populate ever more extraordinary virtual worlds.

Acronyms

AE Autoencoder network.

AEGAN Autoencoding Generative Adversarial Network.

CCD Cyclic Coordinate Descent.

CMU The Carnegie Mellon University Graphics Lab Motion Capture Database.

CNN Convolutional Neural Networks.

CRBM Conditioned Restricted Boltzmann Machine.

DoF Degree of Freedom.

ERD Encoder-Recurrent-Decoder.

FABRIK Forward and Backward Reaching Inverse Kinematics.

FK Forward Kinematics.

GAN Generative Adversarial Network.

GCN Graph Convolutional Network.

GPLVM Gaussian Process Latent Variable Model.

ICA Independent Component Analysis.

IK Inverse Kinematics.

LMA Laban Movement Analysis (LMA).

LoA Line of Action.

LTI Linear Time Invariant.

MoCap Motion Capture.

MSE Mean Squared Error.

NeRF Neural Radiance Field.

NN Neural Network.

PCA Principal Components Analysis.

RBF Radial Basis Function.

RBM Restricted Boltzmann Machine.

ReLU Rectified Linear Unit.

RNN Recurrent Neural Network.

SPL Structured Prediction Layer.

t-SNE t-Distributed Stochastic Neighbor Embedding.

VAE Variational Autoencoder network.

List of Figures

Bibliography

[ABC96] Kenji Amaya, Armin Bruderlin, and Tom W. Calvert. "Emotion from Motion". In: *Proceedings of Graphics Interface '96*. GI '96. Toronto, Ontario, Canada: Canadian Human-Computer Communications Society, May 1996. ISBN: 0-9695338-5-3. DOI: 10.20380/gi1996.26.

[Abe+20] Kfir Aberman et al. "Skeleton-Aware Networks for Deep Motion Retargeting". In: *ACM Trans. Graph.* 39.4 (July 2020), 62:62:1–62:62:14. ISSN: 0730-0301. DOI: 10.1145/3386569.3392462.

[ACC15] Andreas Aristidou, Panayiotis Charalambous, and Yiorgos Chrysanthou. "Emotion Analysis and Classification: Understanding the Performers' Emotions Using the LMA Entities". In: *Computer Graphics Forum* 34.6 (2015), pp. 262–276. ISSN: 1467-8659. DOI: 10.1111/cgf.12598.

[ACL16] Andreas Aristidou, Yiorgos Chrysanthou, and Joan Lasenby. "Extending FABRIK with Model Constraints". In: *Computer Animation and Virtual Worlds* 27.1 (2016), pp. 35–57. ISSN: 1546-427X. DOI: 10.1002/cav.1630.

[AF02] Okan Arikan and David A. Forsyth. "Interactive Motion Generation from Examples". In: *ACM Trans. Graph.* 21.3 (July 2002), pp. 483–490. ISSN: 0730-0301. DOI: 10.1145/566654.566606.

[AKH19] Emre Aksan, Manuel Kaufmann, and Otmar Hilliges. "Structured Prediction Helps 3D Human Motion Modelling". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. ICCV '19. 2019, pp. 7144–7153. DOI: 10.1109/ICCV.2019.00724.

[Aks+21] Emre Aksan et al. "A Spatio-temporal Transformer for 3D Human Motion Prediction". In: *2021 International Conference on 3D Vision (3DV)*. Dec. 2021, pp. 565–574. DOI: 10.1109/3DV53792.2021.00066.

[AL11] Andreas Aristidou and Joan Lasenby. "FABRIK: A Fast, Iterative Solver for the Inverse Kinematics Problem". In: *Graphical Models* 73.5 (Sept. 2011), pp. 243–260. ISSN: 1524-0703. DOI: 10.1016/j.gmod.2011.05.003.

[Ala+12] Sarah Fdili Alaoui et al. "Movement Qualities as Interaction Modality". In: *Proceedings of the Designing Interactive Systems Conference*. DIS '12.

New York, NY, USA: Association for Computing Machinery, June 2012, pp. 761–769. ISBN: 978-1-4503-1210-3. DOI: 10.1145/2317956.2318071.

[Ali+20] 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)*. CVPR '20. June 2020, pp. 5222–5231. DOI: 10.1109/CVPR42600.2020.00527.

[AM00] Marc Alexa and Wolfgang Müller. "Representing Animations by Principal Components". In: *Computer Graphics Forum* 19.3 (2000), pp. 411–418. ISSN: 1467-8659. DOI: 10.1111/1467-8659.00433.

[AP19] Omid Alemi and Philippe Pasquier. "Machine Learning for Data-Driven Movement Generation: A Review of the State of the Art". In: *arXiv:1903.08356 [cs, stat]* (Mar. 2019). DOI: 10.48550/arxiv.1903.08356. arXiv: 1903.08356 [cs, stat].

[Ari+17] Andreas Aristidou et al. "Emotion Control of Unstructured Dance Movements". In: *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA '17. New York, NY, USA: Association for Computing Machinery, July 2017, pp. 1–10. ISBN: 978-1-4503-5091-4. DOI: 10.1145/3099564.3099566.

[Ari+18] Andreas Aristidou et al. "Inverse Kinematics Techniques in Computer Graphics: A Survey: Inverse Kinematics Techniques in Computer Graphics". In: *Computer Graphics Forum* 37.6 (Sept. 2018), pp. 35–58. ISSN: 01677055. DOI: 10.1111/cgf.13310.

[Av16] Shailen Agrawal and Michiel van de Panne. "Task-Based Locomotion". In: *ACM Trans. Graph.* 35.4 (2016), pp. 1–11. DOI: 10.1145/2897824.2925893.

[AYB17] Michel Abdul-Massih, Innfarn Yoo, and Bedrich Benes. "Motion Style Retargeting to Characters With Different Morphologies". In: *Computer Graphics Forum* 36.6 (2017), pp. 86–99. ISSN: 1467-8659. DOI: 10.1111/cgf.12860.

[Bad+94] Norman I. Badler et al. "Posture Interpolation with Collision Avoidance". In: *Proceedings of Computer Animation '94*. May 1994, pp. 13–20. DOI: 10.1109/CA.1994.324011.

[Bar+13] Avi Barliya et al. "Expression of Emotion in the Kinematics of Locomotion". In: *Experimental brain research* 225.2 (2013), pp. 159–176. DOI: 10.1007/s00221-012-3357-4.

[Bar+19] Lisa Feldman Barrett et al. "Emotional Expressions Reconsidered: Challenges to Inferring Emotion From Human Facial Movements". In: *Psychol Sci Public Interest* 20.1 (July 2019), pp. 1–68. ISSN: 1529-1006. DOI: 10.1177/1529100619832930.

[BC15] Michael Büttner and Simon Clavet. *Motion Fields - The Road to Next Gen Animation*. Nucl.ai, 2015. URL: https://www.youtube.com/watch?v=z_wpgHFSWss.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (Aug. 2013), pp. 1798–1828. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2013.50.

[Ben+22] Raphael Bensadoun et al. *Neural Inverse Kinematics*. May 2022. DOI: 10.48550/arXiv.2205.10837. arXiv: 2205.10837 [cs].

[Ber+15] Mathias Berglund et al. "Bidirectional Recurrent Neural Networks as Generative Models". In: *Advances in Neural Information Processing Systems*. Vol. 28. NEURIPS '15. Curran Associates, Inc., 2015. DOI: 10.48550/arxiv.1504.01575.

[BH00] Matthew Brand and Aaron Hertzmann. "Style Machines". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 183–192. ISBN: 978-1-58113-208-3. DOI: 10.1145/344779.344865.

[BH89] Richard H. Bartels and Ines Hardtke. "Speed Adjustment for Key-Frame Interpolation". In: *Proceedings of Graphics Interface '89*. Vol. 89. GI '89. London, Ontario, Canada, 1989, p. 6. DOI: 10.20380/gi1989.03.

[Bir55] Ray L. Birdwhistell. "Background to Kinesics". In: *ETC: A Review of General Semantics* 13.1 (1955), pp. 10–18. ISSN: 0014-164X. URL: https://www.jstor.org/stable/42581570.

[Bis92] Leslie Bishko. "Relationships between Laban Movement Analysis and Computer Animation". In: *Dance and Technology* 1 (1992).

[BK05] Samuel R. Buss and Jin-Su Kim. "Selectively Damped Least Squares for Inverse Kinematics". In: *Journal of Graphics Tools* 10.3 (Jan. 2005), pp. 37–49. ISSN: 1086-7651. DOI: 10.1080/2151237X.2005.10129202.

[BKL18] 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)*. June 2018, pp. 1499–149909. DOI: 10.1109/CVPRW.2018.00191.

[Bla48] Preston Blair. *Animation*. Walter Foster Publishing, 1948.

[Blo95] Jack Block. "A Contrarian View of the Five-Factor Approach to Personality Description". In: *Psychological Bulletin* 117 (1995), pp. 187–215. ISSN: 1939-1455. DOI: 10.1037/0033-2909.117.2.187.

[Bow00] Richard Bowden. "Learning Statistical Models of Human Motion". In: *Proceedings of CVPR 2000 - IEEE Workshop on Human Modeling, Analysis and Synthesis*. Vol. 2000. Hilton Head, South Carolina, U.S.A., 2000.

[BS07] Randolph Blake and Maggie Shiffrar. "Perception of Human Motion". In: *Annual review of psychology* 58 (2007), p. 47. DOI: 10.1146/annurev.psych.57.102904.190152.

[BS79] Norman I. Badler and Stephen W. Smoliar. "Digital Representations of Human Movement". In: *ACM Comput. Surv.* 11.1 (Mar. 1979), pp. 19–38. ISSN: 0360-0300. DOI: 10.1145/356757.356760.

[Bus09] Samuel R Buss. *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods*. Tech. rep. Sept. 2009.

[Büt+17] Judith Bütepage et al. "Deep Representation Learning for Human Motion Prediction and Classification". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. CVPR '17. July 2017, pp. 1591–1599. DOI: 10.1109/CVPR.2017.173.

[Büt19] Michael Büttner. *Machine Learning for Motion Synthesis and Character Control*. I3D'19, May 2019. URL: https://www.youtube.com/watch?v=zuvmQxcCOM4&t=3587s.

[BW71] Nestor Burtnyk and Marceli Wein. "Computer-Generated Key-Frame Animation". In: *Journal of the SMPTE* 80.3 (Mar. 1971), pp. 149–153. ISSN: 0361-4573. DOI: 10.5594/J07698.

[BW76] Nestor Burtnyk and Marceli Wein. "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation". In: *Commun. ACM* 19.10 (Oct. 1976), pp. 564–569. ISSN: 0001-0782. DOI: 10.1145/360349.360357.

[BW95] Armin Bruderlin and Lance Williams. "Motion Signal Processing". In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '95*. SIGGRAPH '95. ACM Press, 1995, pp. 97–104. ISBN: 978-0-89791-701-8. DOI: 10.1145/218380.218421.

[Cai+20] Yujun Cai et al. "Learning Progressive Joint Propagation for Human Motion Prediction". In: *Computer Vision – ECCV 2020*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 226–242. ISBN: 978-3-030-58571-6. DOI: 10.1007/978-3-030-58571-6_14.

[Cao+20] Zhe Cao et al. "Long-Term Human Motion Prediction with Scene Context". In: *Computer Vision – ECCV 2020*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 387–404. ISBN: 978-3-030-58452-8. DOI: 10.1007/978-3-030-58452-8_23.

[CH05] Jinxiang Chai and Jessica K. Hodgins. "Performance Animation from Low-Dimensional Control Signals". In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. New York, NY, USA: Association for Computing Machinery, July 2005, pp. 686–696. ISBN: 978-1-4503-7825-3. DOI: 10.1145/1186822.1073248.

[CH07] Jinxiang Chai and Jessica K. Hodgins. "Constraint-Based Motion Optimization Using a Statistical Dynamic Model". In: *ACM SIGGRAPH 2007 Papers*. SIGGRAPH '07. New York, NY, USA: Association for Computing Machinery, July 2007, 8–es. ISBN: 978-1-4503-7836-9. DOI: 10.1145/1275808.1276387.

[Chi+00] Diane Chi et al. "The EMOTE Model for Effort and Shape". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '00*. SIGGRAPH '00. ACM Press, 2000, pp. 173–182. ISBN: 978-1-58113-208-3. DOI: 10.1145/344779.352172.

[CHP89] John. E. Chadwick, David. R. Haumann, and Richard. E. Parent. "Layered Construction for Deformable Animated Characters". In: *SIGGRAPH Comput. Graph.* 23.3 (July 1989), pp. 243–252. ISSN: 0097-8930. DOI: 10.1145/74334.74358.

[CL06] Yu-Ren Chien and Jing-Sin Liu. "Learning the Stylistic Similarity Between Human Motions". In: *Advances in Visual Computing*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 170–179. ISBN: 978-3-540-48631-2. DOI: 10.1007/11919476_18.

[Cla16] Simon Clavet. *Motion Matching and The Road to Next-Gen Animation*. San Francisco, California, USA, 2016. URL: https://www.gdcvault.com/play/1023280/Motion-Matching-and-The-Road.

[CMV04] Antonio Camurri, Barbara Mazzarino, and Gualtiero Volpe. "Analysis of Expressive Gesture: The EyesWeb Expressive Gesture Processing Library". In: *Gesture-Based Communication in Human-Computer Interaction*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 460–467. ISBN: 978-3-540-24598-8. DOI: 10.1007/978-3-540-24598-8_42.

[CON05] Bing-Yu Chen, Yutaka Ono, and Tomoyuki Nishita. "Character Animation Creation Using Hand-Drawn Sketches". In: *Visual Comput* 21.8 (Sept. 2005), pp. 551–558. ISSN: 1432-2315. DOI: 10.1007/s00371-005-0333-z.

[CÖS19] Loïc Ciccone, Cengiz Öztireli, and Robert W. Sumner. "Tangent-Space Optimization for Interactive Animation Control". In: *ACM Trans. Graph.* 38.4 (July 2019), pp. 1–10. ISSN: 07300301. DOI: 10.1145/3306346.3322938.

[Cre+17] Arthur Crenn et al. “Toward an Efficient Body Expression Recognition Based on the Synthesis of a Neutral Movement”. In: *19th ACM International Conference on Multimodal Interaction*. Vol. 17. ICMI 2017 Proceedings of the 19th ACM International Conference on Multimodal Interaction. Glasgow, United Kingdom, Nov. 2017. DOI: 10.1145/3136755.3136763.

[Cre+20] Arthur Crenn et al. “Generic Body Expression Recognition Based on Synthesis of Realistic Neutral Motion”. In: *IEEE Access* 8 (2020), pp. 207758–207767. DOI: 10.1109/ACCESS.2020.3038473.

[CSY20] 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)*. CVPR ’20. June 2020, pp. 6518–6526. DOI: 10.1109/CVPR42600.2020.00655.

[Dav+06] James Davis et al. “A Sketching Interface for Articulated Figure Animation”. In: *ACM SIGGRAPH 2006 Courses*. SIGGRAPH ’06. New York, NY, USA: Association for Computing Machinery, July 2006, 15–es. ISBN: 978-1-59593-364-5. DOI: 10.1145/1185657.1185776.

[Del+21] Florian Delconte et al. “Tree Defect Segmentation Using Geometric Features and CNN”. In: *Reproducible Research in Pattern Recognition*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 80–100. ISBN: 978-3-030-76423-4. DOI: 10/grqz6c.

[Des+21] Yann Desmarais et al. “A Review of 3D Human Pose Estimation Algorithms for Markerless Motion Capture”. In: *Computer Vision and Image Understanding* 212.C (Nov. 2021). ISSN: 1077-3142. DOI: 10.1016/j.cviu.2021.103275.

[Du+19] Han Du et al. “Stylistic Locomotion Modeling and Synthesis Using Variational Generative Models”. In: *Motion, Interaction and Games*. MIG ’19. New York, NY, USA: ACM, 2019, 32:1–32:10. ISBN: 978-1-4503-6994-7. DOI: 10.1145/3359566.3360083.

[Dur+16] Funda Durupinar et al. “PERFORM: Perceptual Approach for Adding OCEAN Personality to Human Motion Using Laban Movement Analysis”. In: *ACM Trans. Graph.* 36.1 (Oct. 2016). ISSN: 0730-0301. DOI: 10.1145/2983620.

[EA02] Hillary Anger Elfenbein and Nalini Ambady. “On the Universality and Cultural Specificity of Emotion Recognition: A Meta-Analysis”. In: *Psychological Bulletin* 128.2 (2002), pp. 203–235. ISSN: 1939-1455. DOI: 10.1037/0033-2909.128.2.203.

[EA10] Seyed Ali Etemad and Ali Arya. "Modeling and Transformation of 3D Human Motion". In: *Proceedings of the International Conference on Computer Graphics Theory and Applications*. Angers, France, 2010, pp. 307–315. ISBN: 978-989-674-026-9. DOI: 10.5220/0002755003070315.

[EA14] Seyed Ali Etemad and Ali Arya. "Classification and Translation of Style and Affect in Human Motion Using RBF Neural Networks". In: *Neurocomputing* 129 (Apr. 2014), pp. 585–595. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2013.09.001.

[EA16] Seyed Ali Etemad and Ali Arya. "Expert-Driven Perceptual Features for Modeling Style and Affect in Human Motion". In: *IEEE Transactions on Human-Machine Systems* 46.4 (2016), pp. 534–545. DOI: 10.1109/THMS.2016.2537760.

[EF67] P. Ekman and W. V. Friesen. "Head and Body Cues in the Judgment of Emotion: A Reformulation". In: *Percept Mot Skills* 24.3 (June 1967), pp. 711–724. ISSN: 1558-688X. DOI: 10.2466/pms.1967.24.3.711.

[EF69] Paul Ekman and Wallace V. Friesen. "The Repertoire of Nonverbal Behavior: Categories, Origins, Usage, and Coding". In: *semiotica* 1.1 (1969), pp. 49–98. DOI: 10.1515/semi.1969.1.1.49.

[Fek+95] Jean-Daniel Fekete et al. "TicTacToon: A Paperless System for Professional 2D Animation". In: *SIGGRAPH '95 22nd International ACM Conference on Computer Graphics and Interactive Techniques*. Los-Angeles, United States: ACM, Aug. 1995, pp. 79–90. DOI: 10.1145/218380.218417.

[Fen+12] Andrew Feng et al. "An Analysis of Motion Blending Techniques". In: *Motion in Games*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 232–243. ISBN: 978-3-642-34710-8. DOI: 10.1007/978-3-642-34710-8_22.

[FF05] Kate Forbes and Eugene Fiume. "An Efficient Search Algorithm for Motion Data Using Weighted PCA". In: *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '05. 2005, pp. 67–76. DOI: 10.1145/1073368.1073377.

[FP14] Nesrine Fourati and Catherine Pelachaud. "Emilya: Emotional Body Expression in Daily Actions Database". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. Reykjavik, Iceland: European Languages Resources Association (ELRA), May 2014, pp. 3486–3493. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/334_Paper.pdf.

[Fra+15] Katerina Fragkiadaki et al. “Recurrent Network Models for Human Dynamics”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. ICCV ’15. 2015, pp. 4346–4354. DOI: 10.1109/ICCV.2015.494.

[Gal92] Peggy E. Gallaher. “Individual Differences in Nonverbal Behavior: Dimensions of Style”. In: *Journal of Personality and Social Psychology* 63 (1992), pp. 133–145. ISSN: 1939-1315. DOI: 10.1037/0022-3514.63.1.133.

[GC19] 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* 33.01 (July 2019), pp. 2580–2587. ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33012580.

[GCR13] Martin Guay, Marie-Paule Cani, and Rémi Ronfard. “The Line of Action: An Intuitive Interface for Expressive Character Posing”. In: *ACM Trans. Graph.* 32.6 (Nov. 2013), pp. 1–8. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/2508363.2508397.

[GEB16] Leon Gatys, Alexander Ecker, and Matthias Bethge. “A Neural Algorithm of Artistic Style”. In: *Journal of Vision* 16.12 (Sept. 2016), p. 326. ISSN: 1534-7362. DOI: 10.1167/16.12.326.

[Gho+17] Partha Ghosh et al. “Learning Human Motion Models for Long-Term Predictions”. In: *2017 International Conference on 3D Vision (3DV)*. Oct. 2017, pp. 458–466. DOI: 10.1109/3DV.2017.00059.

[Gho+20] Saeed Ghorbani et al. “Probabilistic Character Motion Synthesis Using a Hierarchical Deep Latent Variable Model”. In: *Computer Graphics Forum* 39.8 (2020), pp. 225–239. ISSN: 1467-8659. DOI: 10.1111/cgf.14116.

[GJH01] Aphrodite Galata, Neil Johnson, and David Hogg. “Learning Variable-Length Markov Models of Behavior”. In: *Computer Vision and Image Understanding* 81.3 (2001), pp. 398–413. DOI: 10.1006/cviu.2000.0894.

[Gle+08] Michael Gleicher et al. “Snap-Together Motion: Assembling Run-Time Animations”. In: *ACM SIGGRAPH 2008 Classes*. SIGGRAPH ’08. New York, NY, USA: Association for Computing Machinery, Aug. 2008, pp. 1–9. ISBN: 978-1-4503-7845-1. DOI: 10.1145/1401132.1401203.

[Glo+11] Donald Glowinski et al. “Toward a Minimal Representation of Affective Gestures”. In: *IEEE Transactions on Affective Computing* 2.2 (2011), pp. 106–118. DOI: 10.1109/T-AFFC.2011.7.

[Gol90] Lewis R. Goldberg. “An Alternative "Description of Personality": The Big-Five Factor Structure”. In: *Journal of Personality and Social Psychology* 59 (1990), pp. 1216–1229. ISSN: 1939-1315. DOI: 10.1037/0022-3514.59.6.1216.

[Goo+14] Ian J. Goodfellow et al. "Generative Adversarial Networks". In: *arXiv:1406.2661 [cs, stat]* (June 2014). arXiv: 1406.2661 [cs, stat]. URL: http://arxiv.org/abs/1406.2661.

[Gop+19] Anand Gopalakrishnan et al. "A Neural Temporal Model for Human Motion Prediction". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. CVPR '19. June 2019, pp. 12108–12117. DOI: 10.1109/CVPR.2019.01239.

[Gra98] F. Sebastian Grassia. "Practical Parameterization of Rotations Using the Exponential Map". In: *Journal of Graphics Tools* 3.3 (Jan. 1998), pp. 29–48. ISSN: 1086-7651. DOI: 10.1080/10867651.1998.10487493.

[GRC19] Maxime Garcia, Remi Ronfard, and Marie-Paule Cani. "Spatial Motion Doodles: Sketching Animation in VR Using Hand Gestures and Laban Motion Analysis". In: *Motion, Interaction and Games*. MIG '19. New York, NY, USA: Association for Computing Machinery, Oct. 2019, pp. 1–10. ISBN: 978-1-4503-6994-7. DOI: 10.1145/3359566.3360061.

[GRG96] Shang Guo, James Roberge, and Thom Grace. "Controlling the Movement of an Articulated Figure Using Parametric Frame-Space Interpolation". In: *Fourth International Conference on Computer-Aided Design and Computer Graphics*. Vol. 2644. SPIE, 1996, pp. 766–772. DOI: 10.1117/12.235491.

[Gro+04] Keith Grochow et al. "Style-Based Inverse Kinematics". In: *ACM SIGGRAPH 2004 Papers*. SIGGRAPH '04. New York, NY, USA: ACM, 2004, p. 10. DOI: 10.1145/1186562.1015755.

[Gro95] Ed Groff. "Laban Movement Analysis: Charting the Ineffable Domain of Human Movement". In: *Journal of Physical Education, Recreation & Dance* 66.2 (Feb. 1995), pp. 27–30. ISSN: 0730-3084. DOI: 10.1080/07303084.1995.10607038.

[Gue05] Ann Hutchinson Guest. *Labanotation: The System of Analyzing and Recording Movement*. Fourth. New York: Routledge, Jan. 2005. ISBN: 978-0-203-62612-2. DOI: 10.4324/9780203626122.

[Gvv13] Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. "Flexible Muscle-Based Locomotion for Bipedal Creatures". In: *ACM Trans. Graph.* 32.6 (Nov. 2013), pp. 1–11. ISSN: 07300301. DOI: 10.1145/2508363.2508399.

[Hab+17] Ikhsanul Habibie et al. "A Recurrent Variational Autoencoder for Human Motion Synthesis". In: *Procedings of the British Machine Vision Conference 2017*. London, UK: British Machine Vision Association, 2017, p. 119. ISBN: 978-1-901725-60-5. DOI: 10.5244/C.31.119.

[HAB20] Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. "MoGlow: Probabilistic and Controllable Motion Synthesis Using Normalising Flows". In: *ACM Trans. Graph.* 39.6 (Nov. 2020), 236:1–236:14. ISSN: 0730-0301. DOI: 10.1145/3414685.3417836.

[Hah+15] Fabian Hahn et al. "Sketch Abstractions for Character Posing". In: *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA '15. New York, NY, USA: Association for Computing Machinery, Aug. 2015, pp. 185–191. ISBN: 978-1-4503-3496-9. DOI: 10.1145/2786784.2786785.

[Har+20] Félix G. Harvey et al. "Robust Motion In-Betweening". In: *ACM Trans. Graph.* 39.4 (July 2020), 60:60:1–60:60:12. ISSN: 0730-0301. DOI: 10.1145/3386569.3392480.

[Har18] Geoff Harrower. *Real Player Motion Tech in 'EA Sports UFC 3'*. Game Developer Conference, 2018. URL: https://www.gdcvault.com/play/1025228/Real-Player-Motion-Tech-in.

[He+22] Chengan He et al. *NeMF: Neural Motion Fields for Kinematic Animation*. June 2022. DOI: 10.48550/arXiv.2206.03287. arXiv: 2206.03287 [cs].

[HG07] Rachel Heck and Michael Gleicher. "Parametric Motion Graphs". In: *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*. I3D '07. New York, NY, USA: Association for Computing Machinery, Apr. 2007, pp. 129–136. ISBN: 978-1-59593-628-8. DOI: 10.1145/1230100.1230123.

[HGM19] Alejandro Hernandez, Jürgen Gall, and Francesc Moreno. "Human Motion Prediction via Spatio-Temporal Inpainting". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. ICCV '19. Oct. 2019, pp. 7133–7142. DOI: 10.1109/ICCV.2019.00723.

[HHN10] Joseph Henrich, Steven J. Heine, and Ara Norenzayan. "The Weirdest People in the World?" In: *Behavioral and Brain Sciences* 33.2-3 (June 2010), pp. 61–83. ISSN: 1469-1825, 0140-525X. DOI: 10.1017/S0140525X0999152X.

[HK10] Yazhou Huang and Marcelo Kallmann. "Motion Parameterization with Inverse Blending". In: *International Conference on Motion in Games*. Springer, 2010, pp. 242–253. DOI: 10.1007/978-3-642-16958-8_23.

[HKS17] Daniel Holden, Taku Komura, and Jun Saito. "Phase-Functioned Neural Networks for Character Control". In: *ACM Trans. Graph.* 36.4 (July 2017), pp. 1–13. ISSN: 07300301. DOI: 10.1145/3072959.3073663.

[Hol+15] Daniel Holden et al. "Learning Motion Manifolds with Convolutional Autoencoders". In: *SIGGRAPH ASIA 2015 Technical Briefs*. SA '15.

Kobe, Japan: ACM Press, 2015, pp. 1–4. ISBN: 978-1-4503-3930-8. DOI: 10.1145/2820903.2820918.

[Hol+17] Daniel Holden et al. "Fast Neural Style Transfer for Motion Data". In: *IEEE Comput. Grap. Appl.* 37.4 (2017), pp. 42–49. ISSN: 0272-1716. DOI: 10.1109/MCG.2017.3271464.

[Hol+20] Daniel Holden et al. "Learned Motion Matching". In: *ACM Trans. Graph.* 39.4 (July 2020), 53:53:1–53:53:12. ISSN: 0730-0301. DOI: 10.1145/3386569.3392440.

[HP18] Félix G. Harvey and Christopher Pal. "Recurrent Transition Networks for Character Locomotion". In: *SIGGRAPH Asia 2018 Technical Briefs*. SA '18. New York, NY, USA: Association for Computing Machinery, Dec. 2018, pp. 1–4. ISBN: 978-1-4503-6062-3. DOI: 10.1145/3283254.3283277.

[HP22] Jewoong Hwang and Kyoungju Park. "Audio-Driven Facial Animation: A Survey". In: *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. Oct. 2022, pp. 614–617. DOI: 10/grqphd.

[HPP05] Eugene Hsu, Kari Pulli, and Jovan Popović. "Style Translation for Human Motion". In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. New York, NY, USA: Association for Computing Machinery, July 2005, pp. 1082–1089. ISBN: 978-1-4503-7825-3. DOI: 10.1145/1186822.1073315.

[HR02] Geoffrey Hinton and Sam Roweis. "Stochastic Neighbor Embedding". In: *Proceedings of the 15th International Conference on Neural Information Processing Systems*. NEURIPS '02. Cambridge, MA, USA: MIT Press, Jan. 2002, pp. 857–864.

[HSK16] 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), pp. 1–11. ISSN: 07300301. DOI: 10.1145/2897824.2925975.

[HTY05] K. Hachimura, K. Takashina, and M. Yoshimura. "Analysis and Evaluation of Dancing Movement Based on LMA". In: *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.* Aug. 2005, pp. 294–299. DOI: 10.1109/ROMAN.2005.1513794.

[Hua+17] Jing Huang et al. "Multi-Variate Gaussian-Based Inverse Kinematics". In: *Computer Graphics Forum* 36.8 (2017), pp. 418–428. ISSN: 1467-8659. DOI: 10.1111/cgf.13089.

[Ion+14] Catalin Ionescu et al. "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments". In: *IEE TPMAMI* 36.7 (July 2014), pp. 1325–1339. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2013.248.

[Jac+14] Alec Jacobson et al. “Tangible and Modular Input Device for Character Articulation”. In: *ACM Trans. Graph.* 33.4 (July 2014), 82:1–82:12. ISSN: 0730-0301. DOI: 10.1145/2601097.2601112.

[Jai+16] Ashesh Jain et al. “Structural-RNN: Deep Learning on Spatio-Temporal Graphs”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR ’16. 2016, pp. 5308–5317. URL: https://openaccess.thecvf.com/content_cvpr_2016/html/Jain_Structural-RNN_Deep_Learning_CVPR_2016_paper.html.

[Kal08] Marcelo Kallmann. “Analytical Inverse Kinematics with Body Posture Control”. In: *Computer animation and virtual worlds* 19.2 (2008), pp. 79–91. DOI: 10.1002/cav.176.

[Kap+13] Mubbasir Kapadia et al. “Efficient Motion Retrieval in Large Motion Databases”. In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D ’13. New York, NY, USA: Association for Computing Machinery, Mar. 2013, pp. 19–28. ISBN: 978-1-4503-1956-0. DOI: 10.1145/2448196.2448199.

[KB15] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv:1412.6980 [Cs]*. San Diego, California, US, May 2015. DOI: 10.48550/ARXIV.1412.6980. arXiv: 1412.6980 [cs].

[KB84] Doris H. U. Kochanek and Richard H. Bartels. “Interpolating Splines with Local Tension, Continuity, and Bias Control”. In: *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’84. New York, NY, USA: Association for Computing Machinery, 1984, pp. 33–41. ISBN: 978-0-89791-138-2. DOI: 10.1145/800031.808575.

[KBB82] Doris H. U. Kochanek, Richard H. Bartels, and K. S. Booth. *A Computer System for Smooth Keyframe Animation*. Tech. rep. University of Waterloo, 1982, p. 90.

[Ken80] Adam Kendon. “Gesticulation and Speech: Two Aspects of the Process of Utterance”. In: *The relationship of verbal and nonverbal communication* 25.1980 (1980), pp. 207–227. DOI: 10.1515/9783110813098.207.

[KG03] Lucas Kovar and Michael Gleicher. “Flexible Automatic Motion Blending with Registration Curves”. In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA ’03. Goslar, DEU: Eurographics Association, July 2003, pp. 214–224. ISBN: 978-1-58113-659-3. DOI: 10.2312/SCA03/214-224.

[KG04] Lucas Kovar and Michael Gleicher. “Automated Extraction and Parameterization of Motions in Large Data Sets”. In: *ACM Trans. Graph.* 23.3 (Aug. 2004), pp. 559–568. ISSN: 0730-0301. DOI: 10.1145/1015706.1015760.

[KG18] Yuki Koyama and Masataka Goto. "OptiMo: Optimization-Guided Motion Editing for Keyframe Character Animation". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. Montreal QC, Canada: ACM Press, 2018, pp. 1–12. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3173735.

[KGB19] Jogendra Nath Kundu, Maharshi Gor, and R. Venkatesh Babu. "BiHMP-GAN: Bidirectional 3D Human Motion Prediction GAN". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 8553–8560. ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33018553.

[KGP02] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. "Motion Graphs". In: *ACM Trans. Graph.* 21.3 (July 2002), pp. 473–482. ISSN: 0730-0301. DOI: 10.1145/566654.566605.

[Kim+20] SangBin Kim et al. "Motion Retargetting Based on Dilated Convolutions and Skeleton-specific Loss Functions". In: *Computer Graphics Forum* 39.2 (2020), pp. 497–507. ISSN: 1467-8659. DOI: 10.1111/cgf.13947.

[KK81] Paul R. Kleinginna and Anne M. Kleinginna. "A Categorized List of Motivation Definitions, with a Suggestion for a Consensual Definition". In: *Motivation and Emotion* 5.3 (Sept. 1981), pp. 263–291. ISSN: 1573-6644. DOI: 10.1007/BF00993889.

[KL19] Hye Ji Kim and Sung-Hee Lee. "Perceptual Characteristics by Motion Style Category". In: *Eurographics 2019 - Short Papers*. The Eurographics Association, 2019. DOI: 10.2312/egs.20191000.

[KM05] Richard Kulpa and Franck Multon. "Fast Inverse Kinematics and Kinetics Solver for Human-like Figures". In: *5th IEEE-RAS International Conference on Humanoid Robots, 2005.* IEEE, 2005, pp. 38–43. DOI: 10.1109/ICHR.2005.1573542.

[KMA05] Richard Kulpa, Franck Multon, and Bruno Arnaldi. "Morphology-Independent Representation of Motions for Interactive Human-like Animation". In: *Eurographics*. 2005. DOI: 10.1111/j.1467-8659.2005.00859.x.

[Kne+95] Brian Knep et al. "Dinosaur Input Device". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1995, pp. 304–309. DOI: 10.1145/223904.223943.

[Krü+10] Björn Krüger et al. "Fast Local and Global Similarity Searches in Large Motion Capture Databases." In: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '10. 2010, pp. 1–10. ISBN: 978-3-905674-27-9. DOI: 10.2312/SCA/SCA10/001-010.

[KW14] Diederik P. Kingma and Max Welling. *Auto-Encoding Variational Bayes*. May 2014. DOI: 10.48550/arXiv.1312.6114. arXiv: 1312.6114 [cs, stat].

[LA18] Xiao Lin and Mohamed R. Amer. "Human Motion Modeling Using DV-GANs". In: *arXiv:1804.10652 [cs]* (Apr. 2018). DOI: 10.48550/arxiv.1804.10652. arXiv: 1804.10652 [cs].

[Lai+12] Ranch Y. Q. Lai et al. *Interactive Character Posing by Sparse Coding*. Jan. 2012. DOI: 10.48550/arXiv.1201.1409. arXiv: 1201.1409 [cs].

[Las01] John Lasseter. "Tricks to Animating Characters with a Computer". In: *SIGGRAPH Comput. Graph.* 35.2 (May 2001), pp. 45–47. ISSN: 0097-8930. DOI: 10.1145/563693.563706.

[LAT21] Suhas Lohit, Rushil Anirudh, and Pavan Turaga. "Recovering Trajectories of Unmarked Joints in 3D Human Actions Using Latent Space Optimization". In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2021, pp. 2341–2350. DOI: 10.1109/WACV48630.2021.00239.

[Laz20] Conor Lazarou. *Autoencoding Generative Adversarial Networks*. Apr. 2020. DOI: 10.48550/arxiv.2004.05472. arXiv: 2004.05472.

[LB84] Stan Lee and John Buscema. *How To Draw Comics The Marvel Way*. Simon and Schuster, Sept. 1984. ISBN: 978-0-671-53077-8.

[LC95] Zicheng Liu and Michael F. Cohen. "Keyframe Motion Optimization By Relaxing Speed and Timing". In: *Computer Animation and Simulation '95*. Vienna: Springer Vienna, 1995, pp. 144–153. DOI: 10.1007/978-3-7091-9435-5_11.

[LCM20] Fabrizio Lamberti, Alberto Cannavo, and Paolo Montuschi. "Is Immersive Virtual Reality the Ultimate Interface for 3D Animators?" In: *Computer* 53.4 (Apr. 2020), pp. 36–45. ISSN: 1558-0814. DOI: 10.1109/MC.2019.2908871.

[Lee+02] Jehee Lee et al. "Interactive Control of Avatars Animated with Human Motion Data". In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '02. New York, NY, USA: Association for Computing Machinery, July 2002, pp. 491–500. ISBN: 978-1-58113-521-3. DOI: 10.1145/566570.566607.

[Lee+10] Yongjoon Lee et al. "Motion Fields for Interactive Character Locomotion". In: *ACM SIGGRAPH Asia 2010 Papers*. SA '10. New York, NY, USA: Association for Computing Machinery, Dec. 2010, pp. 1–8. ISBN: 978-1-4503-0439-9. DOI: 10.1145/1866158.1866160.

[Lev+12] Sergey Levine et al. “Continuous Character Control with Low-dimensional Embeddings”. In: *ACM Transaction on Graphics* 31.4 (July 2012), 28:1–28:10. ISSN: 0730-0301. DOI: 10.1145/2185520.2185524.

[LG15] Caroline Larboulette and Sylvie Gibet. “A Review of Computable Expressive Descriptors of Human Motion”. In: *Proceedings of the 2nd International Workshop on Movement and Computing*. MOCO ’15. New York, NY, USA: Association for Computing Machinery, Aug. 2015, pp. 21–28. ISBN: 978-1-4503-3457-0. DOI: 10.1145/2790994.2790998.

[LHP05] C. Karen Liu, Aaron Hertzmann, and Zoran Popović. “Learning Physics-Based Motion Style with Nonlinear Inverse Optimization”. In: *ACM Trans. Graph.* 24.3 (July 2005), pp. 1071–1081. ISSN: 0730-0301. DOI: 10.1145/1073204.1073314.

[Li+18] Zimo Li et al. *Auto-Conditioned Recurrent Networks for Extended Complex Human Motion Synthesis*. July 2018. DOI: 10.48550/arXiv.1707.05363. arXiv: 1707.05363 [cs].

[Li+19] Shujie Li et al. “Bidirectional Recurrent Autoencoder for 3D Skeleton Motion Data Refinement”. In: *Computers & Graphics* 81 (June 2019), pp. 92–103. ISSN: 0097-8493. DOI: 10.1016/j.cag.2019.03.010.

[Li+20] Maosen Li et al. “Dynamic Multiscale Graph Neural Networks for 3D Skeleton Based Human Motion Prediction”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. CVPR ’20. June 2020, pp. 211–220. DOI: 10.1109/CVPR42600.2020.00029.

[Lim19] Jongin Lim. “PMnet: Learning of Disentangled Pose and Movement for Unsupervised Motion Retargeting”. In: *Proceedings of the 30th British Machine Vision Conference (BMVC 2019)*. Cardiff, United Kingdom: British Machine Vision Association, BMVA, Sept. 2019, p. 14. URL: https://bmvc2019.org/wp-content/uploads/papers/0997-paper.pdf.

[LL04] Jehee Lee and Kang Hoon Lee. “Precomputing Avatar Behavior from Human Motion Data”. In: *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA ’04. Goslar, DEU: Eurographics Association, Aug. 2004, pp. 79–87. ISBN: 978-3-905673-14-2. DOI: 10.1145/1028523.1028535.

[LLL18] Kyungho Lee, Seyoung Lee, and Jehee Lee. “Interactive Character Animation by Learning Multi-Objective Control”. In: *ACM Trans. Graph.* 37.6 (Dec. 2018), pp. 1–10. ISSN: 07300301. DOI: 10.1145/3272127.3275071.

[LS01] Jehee Lee and Sung Yong Shin. “A Coordinate-Invariant Approach to Multiresolution Motion Analysis”. In: *Graphical Models* 63.2 (Mar. 2001), pp. 87–105. ISSN: 1524-0703. DOI: 10.1006/gmod.2001.0548.

[LS99] Jehee Lee and Sung Yong Shin. "A Hierarchical Approach to Interactive Motion Editing for Human-like Figures". In: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '99. USA: ACM Press/Addison-Wesley Publishing Co., July 1999, pp. 39–48. ISBN: 978-0-201-48560-8. DOI: 10.1145/311535.311539.

[LT02] Ik Soo Lim and Daniel Thalmann. "Construction of Animation Models out of Captured Data". In: *Proceedings. IEEE International Conference on Multimedia and Expo*. Vol. 1. IEEE, 2002, pp. 829–832. DOI: 10.1109/ICME.2002.1035910.

[LU71] Rudolf von Laban and Lisa Ulmann. *The Mastery of Movement*. 1971.

[LY84] David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*. Vol. 2. Springer, 1984.

[Ma+10] Wanli Ma et al. "Modeling Style and Variation in Human Motion". In: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '10. Goslar, DEU: Eurographics Association, July 2010, pp. 21–30. DOI: 10.2312/SCA/SCA10/021-030.

[Mah+19] Naureen Mahmood et al. "AMASS: Archive of Motion Capture as Surface Shapes". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. ICCV '19. Oct. 2019, p. 10. DOI: 10.1109/ICCV.2019.00554.

[Mao+19] Wei Mao et al. "Learning Trajectory Dependencies for Human Motion Prediction". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. ICCV '19. Oct. 2019, pp. 9488–9496. DOI: 10.1109/ICCV.2019.00958.

[MBC01] Mark Mizuguchi, John Buchanan, and Tom W. Calvert. "Data Driven Motion Transitions for Interactive Games". In: (2001). ISSN: 1017-4656. DOI: 10.2312/egs.20011039.

[MBR17] 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)*. CVPR '17. July 2017, pp. 4674–4683. DOI: 10.1109/CVPR.2017.497.

[MC12] Jianyuan Min and Jinxiang Chai. "Motion Graphs++: A Compact Generative Model for Semantic Motion Analysis and Synthesis". In: *ACM Trans. Graph.* 31.6 (Nov. 2012), p. 1. ISSN: 07300301. DOI: 10.1145/2366145.2366172.

[MC13] Sébastien Moya and Floren Colloud. *A Fast Geometrically-Drive Prioritized Inverse Kinematics Solver*. Poster. Brazil, 2013.

[MC90] Claudia L. Morawetz and Tom W. Calvert. "Goal-Directed Human Animation of Multiple Movements". In: *Proceedings on Graphics Interface '90*.

Halifax, Nova Scotia, Canada: Canadian Information Processing Society, June 1990, pp. 60–67. DOI: 10.20380/gi1990.07.

[MCC09] Jianyuan Min, Yen-Lin Chen, and Jinxiang Chai. "Interactive Generation of Human Animation with Deformable Motion Models". In: *ACM Trans. Graph.* 29.1 (2009), pp. 1–12. DOI: 10.1145/1640443.1640452.

[McN92] David McNeill. *Hand and Mind: What Gestures Reveal about Thought*. University of Chicago Press, 1992.

[MH00] Luis Molina-Tanco and Adrian Hilton. "Realistic Synthesis of Novel Human Movements from a Database of Motion Capture Examples". In: *Proceedings Workshop on Human Motion*. Dec. 2000, pp. 137–142. DOI: 10.1109/HUMO.2000.897383.

[MH08] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data Using T-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. ISSN: 1533-7928. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[MK05] Tomohiko Mukai and Shigeru Kuriyama. "Geostatistical Motion Interpolation". In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. New York, NY, USA: Association for Computing Machinery, July 2005, pp. 1062–1070. ISBN: 978-1-4503-7825-3. DOI: 10.1145/1186822.1073313.

[MLC10] Jianyuan Min, Huajun Liu, and Jinxiang Chai. "Synthesis and Editing of Personalized Stylistic Human Motion". In: *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '10. New York, NY, USA: Association for Computing Machinery, Feb. 2010, pp. 39–46. ISBN: 978-1-60558-939-8. DOI: 10.1145/1730804.1730811.

[MLS20] Wei Mao, Miaomiao Liu, and Mathieu Salzmann. "History Repeats Itself: Human Motion Prediction via Motion Attention". In: *Computer Vision – ECCV 2020*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 474–489. ISBN: 978-3-030-58568-6. DOI: 10.1007/978-3-030-58568-6_28.

[MMK12] Masahiro Mori, Karl F. MacDorman, and Norri Kageki. "The Uncanny Valley [From the Field]". In: *IEEE Robotics & Automation Magazine* 19.2 (June 2012), pp. 98–100. ISSN: 1558-223X. DOI: 10.1109/MRA.2012.2192811.

[Mou+22] Lucas Mourot et al. "A Survey on Deep Learning for Skeleton-Based Human Animation". In: *Computer Graphics Forum* 41.1 (2022), pp. 122–157. ISSN: 1467-8659. DOI: 10.1111/cgf.14426.

[MP07] James McCann and Nancy Pollard. "Responsive Characters from Motion Fragments". In: *ACM SIGGRAPH 2007 Papers*. SIGGRAPH '07. New

York, NY, USA: Association for Computing Machinery, July 2007, 6–es. ISBN: 978-1-4503-7836-9. DOI: 10.1145/1275808.1276385.

[MRC05] Meinard Müller, Tido Röder, and Michael Clausen. "Efficient Content-Based Retrieval of Motion Capture Data". In: *ACM Trans. Graph.* 24.3 (July 2005), pp. 677–685. ISSN: 0730-0301. DOI: 10.1145/1073204.1073247.

[MWE14] Jim McCambridge, John Witton, and Diana R. Elbourne. "Systematic Review of the Hawthorne Effect: New Concepts Are Needed to Study Research Participation Effects". In: *J Clin Epidemiol* 67.3 (Mar. 2014), pp. 267–277. ISSN: 1878-5921. DOI: 10.1016/j.jclinepi.2013.08.015.

[Nak+18] Masaki Nakada et al. "Deep Learning of Biomimetic Sensorimotor Control for Biomechanical Human Animation". In: *ACM Trans. Graph.* 37.4 (July 2018), 56:1–56:15. ISSN: 0730-0301. DOI: 10.1145/3197517.3201305.

[Neb99] Jean-Christophe Nebel. "Keyframe Interpolation with Self-Collision Avoidance". In: *Computer Animation and Simulation '99*. Eurographics. Vienna: Springer, 1999, pp. 77–86. ISBN: 978-3-7091-6423-5. DOI: 10.1007/978-3-7091-6423-5_8.

[NH10] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Madison, WI, USA: Omnipress, June 2010, pp. 807–814. ISBN: 978-1-60558-907-7.

[NW99] Jorge Nocedal and Stephen J. Wright. "Sequential Quadratic Programming". In: *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. New York, NY: Springer, 1999, pp. 526–573. ISBN: 978-0-387-22742-9. DOI: 10.1007/0-387-22742-3_18.

[OH06] Eng-Jon Ong and Adrian Hilton. "Learnt Inverse Kinematics for Animation Synthesis". In: *Graphical Models*. Special Issue on the Vision, Video and Graphics Conference 2005 68.5 (Sept. 2006), pp. 472–483. ISSN: 1524-0703. DOI: 10.1016/j.gmod.2006.07.004.

[Özt+13] Cengiz Öztireli et al. "Differential Blending for Expressive Sketch-Based Posing". In: *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '13. New York, NY, USA: Association for Computing Machinery, July 2013, pp. 155–164. ISBN: 978-1-4503-2132-7. DOI: 10.1145/2485895.2485916.

[Pav+20] Dario Pavllo et al. "Modeling Human Motion with Quaternion-Based Neural Networks". In: *Int J Comput Vis* 128.4 (Apr. 2020), pp. 855–872. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01245-6.

[Pel09] Catherine Pelachaud. "Studies on Gesture Expressivity for a Virtual Agent". In: *Speech Communication* 51.7 (2009), pp. 630–639. DOI: 10.1016/j.specom.2008.04.009.

[Pen+17] Xue Bin Peng et al. "DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning". In: *ACM Trans. Graph.* 36.4 (July 2017), pp. 1–13. ISSN: 07300301. DOI: 10.1145/3072959.3073602.

[Pen+18] Xue Bin Peng et al. "DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills". In: *ACM Trans. Graph.* 37.4 (July 2018), pp. 1–14. ISSN: 07300301. DOI: 10.1145/3197517.3201311.

[Per95] Ken Perlin. "Real Time Responsive Animation with Personality". In: *IEEE Transactions on Visualization and Computer Graphics* 1.1 (1995), pp. 5–15. DOI: 10.1109/2945.468392.

[PGA18] Dario Pavllo, David Grangier, and Michael Auli. "QuaterNet: A Quaternion-based Recurrent Model for Human Motion". In: *arXiv:1805.06485 [cs]* (May 2018). arXiv: 1805.06485 [cs]. URL: http://arxiv.org/abs/1805.06485.

[PSS02] Sang Il Park, Hyun Joon Shin, and Sung Yong Shin. "On-Line Locomotion Generation Based on Motion Blending". In: *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '02. 2002, pp. 105–111. DOI: 10.1145/545261.545279.

[Qin+20] Mengjiao Qin et al. "Remote Sensing Single-Image Resolution Improvement Using A Deep Gradient-Aware Network with Image-Specific Enhancement". In: *Remote Sensing* 12.5 (Jan. 2020), p. 758. ISSN: 2072-4292. DOI: 10/grqz6h.

[Ran+19] Tanmay Randhavane et al. *Identifying Emotions from Walking Using Affective and Deep Features*. June 2019. DOI: 10.48550/arXiv.1906.11884.

[RB09] Daniel Raunhardt and Ronan Boulic. "Motion Constraint". In: *The Visual Computer* 25.5 (2009), pp. 509–518. DOI: 10.1007/s00371-009-0336-2.

[RCB98] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. "Verbs and Adverbs: Multidimensional Motion Interpolation". In: *IEEE Computer Graphics and Applications* 18.5 (Sept. 1998), pp. 32–40. ISSN: 1558-1756. DOI: 10.1109/38.708559.

[RP07] Paul S. A. Reitsma and Nancy S. Pollard. "Evaluating Motion Graphs for Character Animation". In: *ACM Trans. Graph.* 26.4 (Oct. 2007), 18–es. ISSN: 0730-0301. DOI: 10.1145/1289603.1289609.

[RR93] M. Raghavan and B. Roth. "Inverse Kinematics of the General 6R Manipulator and Related Linkages". In: *Journal of Mechanical Design* 115.3 (Sept. 1993), pp. 502–508. ISSN: 1050-0472. DOI: 10.1115/1.2919218.

[RSC01] Charles F. Rose III, Peter-Pike J. Sloan, and Michael F. Cohen. "Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation". In: *Computer Graphics Forum* 20.3 (2001), pp. 239–250. ISSN: 1467-8659. DOI: 10.1111/1467-8659.00516.

[Rus80] James A. Russell. "A Circumplex Model of Affect". In: *Journal of Personality and Social Psychology* 39 (1980), pp. 1161–1178. ISSN: 1939-1315. DOI: 10.1037/h0077714.

[RWV21] Sarah Ribet, Hazem Wannous, and Jean-Philippe Vandeborre. "Survey on Style in 3D Human Body Motion: Taxonomy, Data, Recognition and Its Applications". In: *IEEE Transactions on Affective Computing* 12.4 (Oct. 2021), pp. 928–948. ISSN: 1949-3045. DOI: 10.1109/TAFFC.2019.2906167.

[RZS10] Cheng Ren, Liming Zhao, and Alla Safonova. "Human Motion Synthesis with Optimization-based Graphs". In: *Computer Graphics Forum* 29.2 (2010), pp. 545–554. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2009.01624.x.

[SB85] Scott N. Steketee and Norman I. Badler. "Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control". In: *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '85. New York, NY, USA: Association for Computing Machinery, July 1985, pp. 255–262. ISBN: 978-0-89791-166-5. DOI: 10.1145/325334.325243.

[SCF06] Ari Shapiro, Yong Cao, and Petros Faloutsos. "Style Components". In: *Proceedings of Graphics Interface 2006*. GI '06. CAN: Canadian Information Processing Society, June 2006, pp. 33–39. ISBN: 978-1-56881-308-0. URL: https://dl.acm.org/doi/10.5555/1143079.1143086.

[SDN09] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. "Learning 3-D Object Orientation from Images". In: *2009 IEEE International Conference on Robotics and Automation*. May 2009, pp. 794–800. DOI: 10.1109/ROBOT.2009.5152855.

[SH07] Alla Safonova and Jessica K. Hodgins. "Construction and Optimal Search of Interpolated Motion Graphs". In: *ACM SIGGRAPH 2007 Papers*. SIGGRAPH '07. New York, NY, USA: Association for Computing Machinery, July 2007, 106–es. ISBN: 978-1-4503-7836-9. DOI: 10.1145/1275808.1276510.

[Shi+01] Hyun Joon Shin et al. "Computer Puppetry: An Importance-Based Approach". In: *ACM Trans. Graph.* 20.2 (2001), pp. 67–94. DOI: 10.1145/502122.502123.

[Sho85] Ken Shoemake. "Animating Rotation with Quaternion Curves". In: *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '85. New York, NY, USA: Association for Computing Machinery, July 1985, pp. 245–254. ISBN: 978-0-89791-166-5. DOI: 10.1145/325334.325242.

[SHP04] Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. "Synthesizing Physically Realistic Human Motion in Low-Dimensional, Behavior-Specific Spaces". In: *ACM Trans. Graph.* 23.3 (Aug. 2004), pp. 514–521. ISSN: 0730-0301. DOI: 10.1145/1015706.1015754.

[Smi+19] Harrison Jesse Smith et al. "Efficient Neural Networks for Real-time Motion Style Transfer". In: *PACMCGIT* 2.2 (July 2019), 13:1–13:17. DOI: 10.1145/3340254.

[SMM05] Madhusudhanan Srinivasan, Ronald A. Metoyer, and Eric N. Mortensen. "Controllable Real-Time Locomotion Using Mobility Maps". In: *Proceedings of Graphics Interface 2005*. GI '05. Waterloo, CAN: Canadian Human-Computer Communications Society, May 2005, pp. 51–59. ISBN: 978-1-56881-265-6. URL: https://dl.acm.org/doi/10.5555/1089508.1089518.

[SN88] Lynne Shapiro Brotman and Arun N. Netravali. "Motion Interpolation by Optimal Control". In: *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '88. New York, NY, USA: Association for Computing Machinery, June 1988, pp. 309–315. ISBN: 978-0-89791-275-4. DOI: 10.1145/54852.378531.

[SO06] Hyun Joon Shin and Hyun Seok Oh. "Fat Graphs: Constructing an Interactive Character with Continuous Controls". In: *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '06. Goslar, DEU: Eurographics Association, Sept. 2006, pp. 291–298. ISBN: 978-3-905673-34-0. DOI: 10.2312/SCA/SCA06/291-298.

[SS18] Hanan Salam and Renaud Séguier. "A Survey on Face Modeling: Building a Bridge between Face Analysis and Synthesis". In: *Vis Comput* 34.2 (Feb. 2018), pp. 289–319. ISSN: 1432-2315. DOI: 10/gcxsqf.

[Sta+19] Sebastian Starke et al. "Neural State Machine for Character-Scene Interactions". In: *ACM Trans. Graph.* 38.6 (2019), p. 14. DOI: 10.1145/3355089.3356505.

[Sta+20] Sebastian Starke et al. "Local Motion Phases for Learning Multi-Contact Character Movements". In: *ACM Trans. Graph.* 39.4 (July 2020), 54:54:1–54:54:13. ISSN: 0730-0301. DOI: 10.1145/3386569.3392450.

[Sta+21] Sebastian Starke et al. “Neural Animation Layering for Synthesizing Martial Arts Movements”. In: *ACM Trans. Graph.* 40.4 (July 2021), 92:1–92:16. ISSN: 0730-0301. DOI: 10.1145/3450626.3459881.

[Sum+11] Evan A. Suma et al. “FAAST: The Flexible Action and Articulated Skeleton Toolkit”. In: *2011 IEEE Virtual Reality Conference*. Mar. 2011, pp. 247–248. DOI: 10.1109/VR.2011.5759491.

[Sun+17] Chen Sun et al. “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. ICCV ’17. IEEE Computer Society, Oct. 2017, pp. 843–852. ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.97.

[TBv04] Matthew Thorne, David Burke, and Michiel van de Panne. “Motion Doodles: An Interface for Sketching Character Motion”. In: *ACM Trans. Graph.* 23.3 (Aug. 2004), pp. 424–431. ISSN: 0730-0301. DOI: 10.1145/1015706.1015740.

[TF00] Joshua B. Tenenbaum and William T. Freeman. “Separating Style and Content with Bilinear Models”. In: *Neural Computation* 12.6 (June 2000), pp. 1247–1283. ISSN: 0899-7667. DOI: 10.1162/089976600300015349.

[TF96] Joshua Tenenbaum and William Freeman. “Separating Style and Content”. In: *Advances in Neural Information Processing Systems*. Vol. 9. NEURIPS ’96. MIT Press, 1996. URL: https://proceedings.neurips.cc/paper/1996/hash/70222949cc0db89ab32c9969754d4758-Abstract.html.

[TH09] Graham W. Taylor and Geoffrey E. Hinton. “Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML ’09. New York, NY, USA: ACM, 2009, pp. 1025–1032. ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553505.

[THB06] Lorenzo Torresani, Peggy Hackney, and Christoph Bregler. “Learning Motion Style Synthesis from Perceptual Observations”. In: *Advances in Neural Information Processing Systems*. Vol. 19. NEURIPS ’06. MIT Press, 2006. URL: https://dl.acm.org/doi/10.5555/2976456.2976631.

[THR06] Graham W. Taylor, Geoffrey E. Hinton, and Sam Roweis. “Modeling Human Motion Using Binary Latent Variables”. In: *Advances in Neural Information Processing Systems*. Vol. 19. NEURIPS ’06. MIT Press, 2006. DOI: 10.7551/mitpress/7503.003.0173.

[TLP07] Adrien Treuille, Yongjoon Lee, and Zoran Popović. “Near-Optimal Character Animation with Continuous Control”. In: *ACM SIGGRAPH 2007 Papers*. SIGGRAPH ’07. New York, NY, USA: Association for Computing Machinery, July 2007, 7–es. ISBN: 978-1-4503-7836-9. DOI: 10.1145/1275808.1276386.

[TMD12] Joëlle Tilmanne, Alexis Moinet, and Thierry Dutoit. "Stylistic Gait Synthesis Based on Hidden Markov Models". In: *EURASIP Journal on advances in signal processing* 2012.1 (2012), pp. 1–14. DOI: 10.1186/1687-6180-2012-72.

[TY17] Songqiao Tao and Yumeng Yang. "Collision-Free Motion Planning of a Virtual Arm Based on the FABRIK Algorithm". In: *Robotica* 35.6 (2017), pp. 1431–1450. DOI: 10.1017/S0263574716000205.

[UAT95] Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. "Fourier Principles for Emotion-Based Human Figure Animation". In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '95. New York, NY, USA: Association for Computing Machinery, Sept. 1995, pp. 91–96. ISBN: 978-0-89791-701-8. DOI: 10.1145/218380.218419.

[Uni03] Carnegie Mellon University. *CMU Graphics Lab Motion Capture Database*. 2003. URL: http://mocap.cs.cmu.edu/.

[Urt+04] Raquel Urtasun et al. "Style-Based Motion Synthesis". In: *Computer Graphics Forum* 23.4 (2004), pp. 799–812. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2004.00809.x.

[Vil+18] Ruben Villegas et al. "Neural Kinematic Networks for Unsupervised Motion Retargetting". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. CVPR '18. June 2018, pp. 8639–8648. DOI: 10.1109/CVPR.2018.00901.

[Vol+14] Ekaterina Volkova et al. "The MPI Emotional Body Expressions Database for Narrative Scenarios". In: *PLOS ONE* 9.12 (Dec. 2014), e113647. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0113647.

[Vol+22] Vikram Voleti et al. "SMPL-IK: Learned Morphology-Aware Inverse Kinematics for AI Driven Artistic Workflows". In: *SIGGRAPH Asia 2022 Technical Communications*. SA '22. New York, NY, USA: Association for Computing Machinery, Nov. 2022, pp. 1–7. ISBN: 978-1-4503-9465-9. DOI: 10.1145/3550340.3564227.

[Wan+18] Qi Wang et al. "Transferring Style in Motion Capture Sequences with Adversarial Learning". In: *26th European Symposium on Artifical Neural Networks*. ESANN '18. Bruges, Belgium, Apr. 2018. URL: https://hal.archives-ouvertes.fr/hal-02100672.

[Wan+20] Qi Wang et al. "Adversarial Learning for Modeling Human Motion". In: *Vis Comput* 36.1 (Jan. 2020), pp. 141–160. ISSN: 1432-2315. DOI: 10.1007/s00371-018-1594-7.

[Wan+21] He Wang et al. “Spatio-Temporal Manifold Learning for Human Motions via Long-Horizon Modeling”. In: *ACM Trans. Graph.* 27.1 (Jan. 2021), pp. 216–227. ISSN: 1941-0506. DOI: 10.1109/TVCG.2019.2936810.

[WC11] Xiaolin Wei and Jinxiang Chai. “Intuitive Interactive Human-Character Posing with Millions of Example Poses”. In: *IEEE Computer Graphics and Applications* 31.4 (July 2011), pp. 78–88. ISSN: 1558-1756. DOI: 10.1109/MCG.2009.132.

[WC91] Li-Chun Tommy Wang and Chih Cheng Chen. “A Combined Optimization Method for Solving the Inverse Kinematics Problems of Mechanical Manipulators”. In: *IEEE Trans. Robot. Automat.* 7.4 (1991), pp. 489–499. ISSN: 1042296X. DOI: 10.1109/70.86079.

[WCW14] Xin Wang, Qiudi Chen, and Wanliang Wang. “3D Human Motion Editing and Synthesis: A Survey”. In: *Computational and Mathematical Methods in Medicine* 2014 (2014), p. 11. DOI: 10.1155/2014/104535.

[WCX21] Zhiyong Wang, Jinxiang Chai, and Shihong Xia. “Combining Recurrent Neural Networks and Adversarial Training for Human Motion Synthesis and Control”. In: *ACM Trans. Graph.* 27.1 (Jan. 2021), pp. 14–28. ISSN: 1941-0506. DOI: 10.1109/TVCG.2019.2938520.

[Wel93] Chris Welman. “Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation”. PhD thesis. Simon Fraser University, 1993.

[Wen+21] Yu-Hui Wen et al. “Autoregressive Stylized Motion Synthesis With Generative Flow”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. CVPR ’21. 2021, pp. 13612–13621. DOI: 10.1109/CVPR46437.2021.01340.

[WFH07] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. “Multifactor Gaussian Process Models for Style-Content Separation”. In: *Proceedings of the 24th International Conference on Machine Learning*. ICML ’07. New York, NY, USA: Association for Computing Machinery, June 2007, pp. 975–982. ISBN: 978-1-59593-793-3. DOI: 10.1145/1273496.1273619.

[WFH08] 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 (Feb. 2008), pp. 283–298. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2007.1167.

[WH97] Douglas J. Wiley and James K. Hahn. “Interpolation Synthesis of Articulated Figure Motion”. In: *IEEE Computer Graphics and Applications* 17.6 (Nov. 1997), pp. 39–45. ISSN: 1558-1756. DOI: 10.1109/38.626968.

[WMC11] Xiaolin Wei, Jianyuan Min, and Jinxiang Chai. “Physically Valid Statistical Models for Human Motion Generation”. In: *ACM Trans. Graph.*

30.3 (May 2011), 19:1–19:10. ISSN: 0730-0301. DOI: 10.1145/1966394.1966398.

[WP95] Andrew Witkin and Zoran Popovic. "Motion Warping". In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '95. New York, NY, USA: Association for Computing Machinery, Sept. 1995, pp. 105–108. ISBN: 978-0-89791-701-8. DOI: 10.1145/218380.218422.

[WTR11] Xiaomao Wu, Maxime Tournier, and Lionel Reveret. "Natural Character Posing from a Large Motion Database". In: *IEEE Computer Graphics and Applications* 31.3 (May 2011), pp. 69–77. DOI: 10.1109/MCG.2009.111.

[Xia+15] Shihong Xia et al. "Realtime Style Transfer for Unlabeled Heterogeneous Human Motion". In: *ACM Trans. Graph.* 34.4 (July 2015), 119:1–119:10. ISSN: 07300301. DOI: 10.1145/2766999.

[XLM19] Yi Tian Xu, Yaqiao Li, and David Meger. "Human Motion Prediction Via Pattern Completion in Latent Representation Space". In: *2019 16th Conference on Computer and Robot Vision (CRV)*. CRV '19. May 2019, pp. 57–64. DOI: 10.1109/CRV.2019.00016.

[Yan+18] Xinchen Yan et al. "MT-VAE: Learning Motion Transformations to Generate Multimodal Human Dynamics". In: *Computer Vision – ECCV 2018*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 276–293. ISBN: 978-3-030-01228-1. DOI: 10.1007/978-3-030-01228-1_17.

[YL10] Yuting Ye and C. Karen Liu. "Synthesis of Responsive Motion Using a Dynamic Model". In: *Computer Graphics Forum* 29.2 (2010), pp. 555–562. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2009.01625.x.

[YM16] M. Ersin Yumer and Niloy J. Mitra. "Spectral Style Transfer for Human Motion between Independent Actions". In: *ACM Trans. Graph.* 35.4 (July 2016), pp. 1–8. ISSN: 07300301. DOI: 10.1145/2897824.2925955.

[YN03] Katsu Yamane and Yoshihiko Nakamura. "Natural Motion Animation through Constraining and Deconstraining at Will". In: *IEEE Transactions on Visualization and Computer Graphics* 9.3 (July 2003), pp. 352–360. ISSN: 1941-0506. DOI: 10.1109/TVCG.2003.1207443.

[Yos+11] Wataru Yoshizaki et al. "An Actuated Physical Puppet as an Input Device for Controlling a Digital Manikin". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. New York, NY, USA: Association for Computing Machinery, May 2011, pp. 637–646. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979034.

[Zad16] Kristjan Zadziuk. *Motion Matching, The Future of Games Animation... Today*. 2016. URL: https://www.youtube.com/watch?v=KSTn3ePDt50.

[ZCB00] Liwei Zhao, M. Costa, and N.L. Badler. "Interpreting Movement Manner". In: *Proceedings of the Computer Animation*. CA '00. May 2000, pp. 98–103. DOI: 10.1109/CA.2000.889052.

[Zha+09] Liming Zhao et al. "Automatic Construction of a Minimum Size Motion Graph". In: *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '09. New York, NY, USA: Association for Computing Machinery, Aug. 2009, pp. 27–35. ISBN: 978-1-60558-610-6. DOI: 10.1145/1599470.1599474.

[Zha+18] He Zhang et al. "Mode-Adaptive Neural Networks for Quadruped Motion Control". In: *ACM Trans. Graph.* 37.4 (July 2018), pp. 1–11. ISSN: 07300301. DOI: 10.1145/3197517.3201366.

[Zha+22] Mingyuan Zhang et al. *MotionDiffuse: Text-Driven Human Motion Generation with Diffusion Model*. Aug. 2022. DOI: 10.48550/arXiv.2208.15001. arXiv: 2208.15001 [cs].

[Zho+19] Yi Zhou et al. "On the Continuity of Rotation Representations in Neural Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. CVPR '19. 2019, pp. 5745–5753. DOI: 10.1109/CVPR.2019.00589.

[Zhu+17] Jun-Yan Zhu et al. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. ICCV '17. Oct. 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244.

[Zin19] Fabio Zinno. *From Motion Matching to Motion Synthesis, and All the Hurdles In Between*. 2019. URL: https://www.gdcvault.com/play/1026472/ML-Tutorial-Day-From-Motion.

[ZL16] Xudong Zhu and Kin Fun Li. "Real-Time Motion Capture: An Overview". In: *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*. CISIS '16. July 2016, pp. 522–525. DOI: 10.1109/CISIS.2016.134.

[ZPK20] Chuanqi Zang, Mingtao Pei, and Yu Kong. "Few-Shot Human Motion Prediction via Learning Novel Motion Dynamics". In: *29th International Joint Conference on Artificial Intelligence*. Vol. 1. July 2020, pp. 846–852. DOI: 10.24963/ijcai.2020/118.

[ZS09] Liming Zhao and Alla Safonova. "Achieving Good Connectivity in Motion Graphs". In: *Graphical Models* 71.4 (2009), pp. 139–152. DOI: 10.1016/j.gmod.2009.04.001.

[ZSE17] Shengjia Zhao, Jiaming Song, and Stefano Ermon. *Towards Deeper Understanding of Variational Autoencoding Models*. Feb. 2017. DOI: 10.48550/arXiv.1702.08658. arXiv: 1702.08658 [cs, stat].

[Zv18] Xinyi Zhang and Michiel van de Panne. "Data-Driven Autocompletion for Keyframe Animation". In: *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games - MIG '18*. Limassol, Cyprus: ACM Press, 2018, pp. 1–11. ISBN: 978-1-4503-6015-9. DOI: 10.1145/3274247.3274502.

FOLIO ADMINISTRATIF

THÈSE DE L'UNIVERSITÉ DE LYON OPÉRÉE AU SEIN DE L'INSA LYON

NOM : Victor **DATE DE SOUTENANCE :** 07/04/2023

PRÉNOM : Léon

TITRE : Learning-based Interactive Character Animation Edition

NATURE : Doctorat **NUMERO D'ORDRE :** 2023ISAL0028

ÉCOLE DOCTORALE : Mathématiques et Informatique de Lyon (EDA 512)

SPÉCIALITÉ : Informatique

RÉSUMÉ :

La principale méthode utilisée pour animer un personnage virtuel consiste à éditer le mouvement d'un squelette, sur lequel un modèle sous forme de maillage sera appliqué et déformé au besoin. Pour convaincre le spectateur, le mouvement d'un personnage doit respecter de nombreuses règles implicites, comme le modèle physique qui régit son monde ou les limites de sa morphologie. Ce faisant, il doit aussi rendre explicite l'action réalisée, et laisser transparaître l'état émotionnel du personnage. La subtilité de ces contraintes rend la création d'animation difficile et la production d'une animation réaliste et/ou plaisante repose fortement sur les connaissances, l'expérience et minutie de l'animateur. La démocratisation des techniques de capture de mouvement (Motion Capture, ou MoCap) permet de produire de plus en plus de données de mouvements tirés du monde réel, qui portent intrinsèquement des informations sur ces contraintes. Les travaux présentés dans cette thèse proposent d'exploiter ces données à l'aide de méthodes récentes d'apprentissage automatique, utilisant les réseaux de neurones pour fournir de nouveaux outils aux artistes animateurs.

MOTS-CLEFS : Computer Animation, Character Animation, Animation Editing, Neural Networks

LABORATOIRE DE RECHERCHE : Laboratoire d'Informatique en Image et Systèmes d'Information(LIRIS)

DIRECTRICE DE THÈSE : Véronique Eglin

PRÉSIDENT.E DE JURY : XXX

COMPOSITION DU JURY : M. Jean-Philippe Vandeborre, M. Renaud Séguier, MMe Sylvie Gibet, M. Sébastien Mavromatis, M. Bertrand Kerautret, MMe Véronique Eglin, MMe Saïda Bouakaz, M. Alexandre Meyer,