NATIONAL TECHNICAL UNIVERSITY OF ATHENS
MASTERS IN DATA SCIENCE AND MACHINE LEARNING

Comparative Evaluation on Human Motion Generation and Control diffusion-based methods

DIPLOMA THESIS

of

GEORGIOS PAPOULIAS

Supervisor: Stefanos Kollias
Professor

Co-supervisor: Paraskevi Tzouveli
Dr

Athens, March 2024

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
MASTERS IN DATA SCIENCE AND MACHINE LEARNING

Comparative Evaluation on Human Motion Generation and Control diffusion-based methods

DIPLOMA THESIS

of

GEORGIOS PAPOULIAS

Supervisor: Stefanos Kollias
Professor

Co-supervisor: Paraskevi Tzouveli
Dr

Approved by the examination committee on 15th March 2024.

(Signature)	*(Signature)*	*(Signature)*
....	....	....
Stefanos Kollias	Georgios Stamou	Athanasios Voulodimos
Professor	Professor	Assistant Professor

Athens, March 2024

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
MASTERS IN DATA SCIENCE AND MACHINE LEARNING

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signature)

....
Georgios Papoulias

15th March 2024

Abstract

Human motion generation aims to generate natural human pose sequences and shows immense potential for computer animation. Substantial progress has been made recently in motion data collection technologies and motion generation methods, laying the foundation for increasing interest in human motion generation & control, especially for data-hungry deep architectures which were previously impossible to use in this scope, such as the diffusion-based models. However, despite the recorded progress, challenges remain due to the wide range of possible movements, human sensitivity to motion quality, and limited data availability leading to solutions either low-quality or limited in expressiveness. In this survey, we present a comparative assessment of newly-emerged diffusion-based architectures, which seem to be fully promising in the context of human motion synthesis among other generative approaches. We firstly provide an overview of diffusion models' operation and discuss techniques for representing human motion, along with commonly used motion capture datasets. Subsequently, we delve into the specifics of the architectures we interfere with, followed by the quantitative and qualitative comparison of the methods for two mainstream sub-tasks: text-conditioned human motion generation and human motion trajectory control. Finally, we offer insights and highlight unresolved issues, aiming to provide a comprehensive understanding of this evolving field and spark innovative solutions to its challenges.

Keywords

human motion synthesis, diffusion Models, text-to-motion generation, trajectory control, conditional motion generation

Περίληψη

Η παραγωγή ανθρώπινης κίνησης αποσκοπεί στη δημιουργία αληθοφανών ακολουθιών ανθρώπινων στάσεων και παρουσιάζει τεράστιες δυνατότητες για τον τομέα του ςομπυτερ ανιματιον. Σημαντική πρόοδος έχει σημειωθεί πρόσφατα στις τεχνολογίες συλλογής δεδομένων κίνησης και στις μεθόδους παραγωγής κίνησης, θέτοντας τις θεμέλια για το αυξανόμενο ενδιαφέρον για την παραγωγή ανθρώπινης κίνησης & καθώς τον έλεγχο της, ιδίως για τις βαθιές αρχιτεκτονικές που απαιτούν πολλά δεδομένα και που προηγουμένως ήταν αδύνατο να χρησιμοποιηθούν σε αυτό το πεδίο, όπως τα μοντέλα που βασίζονται στη διάχυση. Ωστόσο, παρά την δεδομένε αυτή πρόοδο, οι προκλήσεις παραμένουν λόγω του μεγάλου εύρους πιθανών κινήσεων του ανθρώπινου σώματος, της ευαισθησίας του ανθρώπινου ματιού στην ποιότητα της κίνησης και της περιορισμένης διαθεσιμότητας δεδομένων που οδηγούν σε λύσεις είτε χαμηλής ποιότητας είτε περιορισμένης εκφραστικότητας. Στην παρούσα έρευνα, παρουσιάζουμε μια συγκριτική αξιολόγηση των πρόσφατα αναδυόμενων αρχιτεκτονικών που βασίζονται στη διάχυση, οι οποίες φαίνεται να είναι πλήρως υποσχόμενες στο πλαίσιο της σύνθεσης ανθρώπινης κίνησης μεταξύ άλλων γενετικών προσεγγίσεων. Αρχικά, παρέχουμε μια επισκόπηση της λειτουργίας των μοντέλων διάχυσης και συζητάμε τεχνικές για την αναπαράσταση της ανθρώπινης κίνησης καθώς και τα πιο ευρύτερα διαδεδομένα σύνολα δεδομένων καταγεγραμμένης κίνησης. Στη συνέχεια, εμβαθύνουμε στις ιδιαιτερότητες των αρχιτεκτονικών με τις οποίες εμπλεκόμαστε, ενώ ακολουθεί η ποσοτική και ποιοτική σύγκριση των μεθόδων για δύο κύριες λειτουργικότητας: παραγωγή ανθρώπινης κίνησης με βάση το κείμενο και έλεγχος της τροχιάς της ανθρώπινης κίνησης. Τέλος, προσφέρουμε ιδέες και επισημαίνουμε άλυτα ζητήματα, με στόχο να παρέχουμε μια ολοκληρωμένη κατανόηση αυτού του εξελισσόμενου πεδίου και να δώσουμε το έναυσμα για καινοτόμες λύσεις όσον αφορά υπάρχοντα προβλήματα και μελλοντικές προκλήσεις.

to Giannis and Mitsos, who left too early and too suddenly

Acknowledgements

I extend my heartfelt gratitude to Dr. Paraskevi Tzouveli, my supervisor, for their invaluable guidance, encouragement, and unwavering support throughout my research journey. Their mentorship, especially during challenging times, played a pivotal role in shaping this thesis. I am deeply thankful to my mother for her endless love, understanding, and patience throughout this endeavor, as well as to my cousin Stratis, whose belief in my capabilities never faded out, providing constant motivation. I am also grateful to my friends for their encouragement, understanding, and companionship, which have been a source of inspiration akin to a second family. Last but certainly not least, special thanks should go to my friend Mike for his continuous encouragement and technical guidance when I faced obstacles which seemed insurmountable back then.

Athens, March 2024

Georgios Papoulias

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 English

Humans plan and execute body motions based on their intention and the environmental stimulus[13]. As an essential goal of artificial intelligence, in general, and machine learning, in specific, reproducing coherent, natural and diverse human-like motion patterns has gained increasing interest from various research communities, such as computer animation[14], computer graphics[15], robotics[16],[17] and human-computer interaction[18], thus addressing a wide range of applications in domains ranging from film production, video games and AR/VR to biomechanics and sports analysis.

Human motion generation is a challenging field, due to several reasons, including the vast span of possible motions (multi-dimensional learning manifold) and the difficulty and cost of acquiring high quality motion data using motion capture techniques. With the rise of Deep Learning[19] in recent years, a rapid development of various generative methods has been witnessed such as Autoregressive Models[20], Variational Autoencoders(VAEs)[21], Normalizing Flows[22], Generative Adversarial Networks(GANs)[23] and Denoising Diffusion Probabilistic Models(DDPMs)[1]. These methods have demonstrated great success across different domains,including text[24][25], image processing[26],[27], video[28],[29] and 3D objects[30],[31]. On the other side, the remarkable progress in human modeling [5],[32],[33] facilitates the extraction [34],[35] of coherent, artifact-free human motion from animation videos, which actually comes down to the task of efficiently estimating human pose from videos as described in [34], [35] and [36], and the construction of large human motion datasets such as [37],[38],[39] and [40] analyzed in one of the following chapters. The rapid progress in the domains enumerated above entails the substantial increase of attention on data-driven human motion generation on behalf of the scientific community during the past decade - and especially the artificial intelligence community.

However, apart from the initial obstacles partially overcome over the past few years, human motion generation - and editing, as a natural aftereffect - is a trivial task impossible to be diminished to just applying deep generative architectures to human motion datasets. The inherent nature of human motion is highly non-linear as human motion patterns are subject to a multitude of physical and bio-mechanical constraints. Another issues is that it is easily perceivable by the human brain, and consequently the human eye, when unnatural kinematics are introduced in an artificially generated motion. As a result, demands in the context of smoothness, plausibility and coherence are high. Additionally, human motion generation needs to integrate context using a conditional signal (more often than not it is either a textual description or an audio message), yielding the extra restriction that the generated motion be in accordance with the conditional signal, raising the degree of difficulty of the task even more.

For the recently emerging text-to-motion or action-to-motion paradigms, where motion is generated conditioned on natural language, another inherent problem is data labeling. For instance, the label ”kick” or the text prompt "a man is performing a kick" could refer to either a soccer kick or a martial arts kick. Inversely, a given kick could be efficiently described by a plethora of textual descriptions, thus constituting a **many-to-many** problem. To this end, diffusion models are the most promising candidates deep network architectures found in literature for expressing a many-to-many distribution problem in the human motion generation and editing setting.

This work will begin to unfold by firstly delving into the fundamentals of diffusion models and justifying their superiority compared to other state-of-the-art deep architectures in the context of human motion generation in Chapter 2. In Chapter 3, the motion representation (e.g. translation, rotation, skeleton topology) used as input for this type of architectures, the datasets used for the training of the networks under evaluation as well as the pivotal role of the **SMPL**[41] model will be analyzed in depth. After having completed the part of displaying the theoretical background needed to fully comprehend the context of the present work, the architectures of the neural networks employed in the human motion generation and editing settings will be analytically elaborated on in Chapter 4. Finally, in Chapter 5 the experiments conducted aiming at performing a rounded comparative evaluation between different diffusion-oriented[42] methods will be displayed, entailing both qualitative and quantitative assessment. The overall conclusions will be summed up in Chapter 6 along with suggesting possible future research directions.

1.2 Ελληνικά

Οι άνθρωποι σχεδιάζουν και εκτελούν κινήσεις του σώματος με βάση την προσωπική πρόθεσή τους και τα περιβαλλοντικά ερεθίσματα[13]. Ως βασικός στόχος της τεχνητής νοημοσύνης, γενικά, και της μηχανικής μάθησης, ειδικότερα, η αναπαραγωγή συνεκτικών, φυσικών και ποικιλόμορφων ανθρώπινων μοτίβων κίνησης έχει αποκτήσει αυξανόμενο ενδιαφέρον από μια πληθώρα ερευνητικών πεδίων, όπως τα κινούμενα σχέδια αναπαραγόμενα από υπολογιστές[14], τα γραφικά υπολογιστών[15], η ρομποτική[16],[17] και αλληλεπίδραση ανθρώπου-υπολογιστή[18], αντιμετωπίζοντας έτσι ένα ευρύ φάσμα εφαρμογών σε τομείς που κυμαίνονται από την παραγωγή ταινιών, τα βιντεοπαιχνίδια και την εικονική και επαυξημένη πραγματικότητα έως τη βιομηχανική και την αθλητική ανάλυση.

Η παραγωγή ανθρώπινης κίνησης αποτελεί ένα απαιτητικό πεδίο, για διάφορους λόγους, όπως το τεράστιο εύρος των πιθανών κινήσεων (πολυδιάστατη κυρτό περίβλημα μάθησης) καθώς και η δυσκολία και το κόστος απόκτησης δεδομένων κίνησης υψηλής ποιότητας με τη χρήση τεχνικών καταγραφής κίνησης. Με την άνοδο της βαθιάς μάθησης τα τελευταία χρόνια, παρατηρήθηκε μια ταχεία ανάπτυξη διαφόρων γεννητικών μεθόδων, όπως τα αυτοπαλινδρομικά μοντέλα, οι μεταβλητοί αυτοκωδικοποιητές (ΈΑΕς)[21], οι κανονικοποιητικές ροές[22], τα γενετικά ανταγωνιστικά δίκτυα(ΓΑΝς)[23] και τα στοχαστικά μοντέλα αποθορυβοποίησης/διάχυσης(ΔΔΠΜς)[1]. Αυτές οι μέθοδοι έχουν επιδείξει μεγάλη επιτυχία σε διάφορους τομείς, όπως επεξεργασία κειμένου[24][25], επεξεργασία εικόνας[26],[27], βίντεο[28],[29] και τρισδιάστατων αντικειμένων[30],[31]. Από την άλλη πλευρά, η αξιοσημείωτη πρόοδος στην ανθρώπινη μοντελοποίηση [5],[32],[33] διευκολύνει την εξαγωγή [34],[35] συνεκτικής, απαλλαγμένης από ασυνέχειες ανθρώπινης κίνησης από βίντεο κινουμένων σχεδίων, η οποία στην πραγματικότητα καταλήγει στο έργο της αποτελεσματικής εκτίμησης της ανθρώπινης στάσης από βίντεο, όπως περιγράφεται στις [34], [35] και [36], και στην κατασκευή μεγάλων συνόλων δεδομένων ανθρώπινης κίνησης όπως [37],[38],[39] και [40] που αναλύονται σε ένα από τα επόμενα κεφάλαια. Η ραγδαία πρόοδος στους τομείς που απαριθμούνται παραπάνω συνεπάγεται τη σημαντική αύξηση της προσοχής της επιστημονικής κοινότητας στην παραγωγή ανθρώπινης κίνησης οδηγούμενη από δεδομένα κατά την τελευταία δεκαετία - και ιδίως της κοινότητας της τεχνητής νοημοσύνης.

Ωστόσο, πέρα από τα αρχικά εμπόδια που ξεπεράστηκαν εν μέρει τα τελευταία χρόνια, η παραγωγή ανθρώπινης κίνησης - και η επεξεργασία της, ως φυσικό επακόλουθο - δεν είναι μια αποστολή που μπορεί να περιοριστεί στην απλή εφαρμογή βαθιών γενεσιουργών αρχιτεκτονικών σε σύνολα δεδομένων ανθρώπινης κίνησης. Η εγγενής φύση της ανθρώπινης κίνησης είναι εξαιρετικά μη γραμμική, καθώς τα μοτίβα που παρατηρούνται στην ανθρώπινη κίνηση υπόκεινται σε πλήθος φυσικών και βιοκινητικών περιορισμών. Ένα άλλο ζήτημα είναι ότι γίνεται εύκολα αντιληπτό από τον ανθρώπινο εγκέφαλο, και κατά συνέπεια από το ανθρώπινο μάτι, όταν εισάγονται αφύσικες κινηματικές ιδιότητες σε μια τεχνητά παραγόμενη κίνηση. Ως αποτέλεσμα, οι απαιτήσεις όσον αφορά τη συνέχειας, της αληθοφάνειας και της συνοχής της παραγόμενης κίνησης είναι υψηλές. Επιπλέον, η παραγωγή ανθρώπινης κίνησης πρέπει να ενσωματώσει το περιεχόμενο χρησιμοποιώντας ένα υπό συνθήκη σήμα (τις περισσότερες φορές πρόκειται είτε για μια περιγραφή μέσω κειμένου είτε για ένα ηχητικό μήνυμα), γεγονός που συνεπάγεται τον επιπλέον περιορισμό ότι η παραγόμενη κίνηση πρέπει να είναι σύμφωνη με το υπό συνθήκη σήμα, αυξάνοντας ακόμη περισσότερο τον βαθμό δυσκολίας του έργου.

Για τα πρόσφατα ανερχόμενα αντικείμενα επιστημονικής ενασχόλησης όπως η μετατροπή κειμένου σε κίνηση ή δράσης σε κίνηση, όπου η κίνηση παράγεται με βάση τη φυσική γλώσσα, ένα άλλο εγγενές πρόβλημα είναι η σήμανση των δεδομένων. Για παράδειγμα, η δράση ¨κλωτσιά¨ ή το κείμενο-οδηγία ¨ένας άνδρας εκτελεί μια κλωτσιά¨ θα μπορούσε να αναφέρεται είτε σε μια κλωτσιά ποδοσφαίρου είτε σε μια κλωτσιά πολεμικών τεχνών. Αντίστροφα, ένα δεδομένο λάκτισμα θα μπορούσε να περιγραφεί αποτελεσματικά από μια πληθώρα κειμενικών περιγραφών, αποτελώντας έτσι ένα πρόβλημα **πολλά-με-πολλά**. Για το σκοπό αυτό, τα μοντέλα διάχυσης είναι οι πιο υποσχόμενες υποψήφιες αρχιτεκτονικές βαθιών δικτύων που έχουν βρεθεί στη βιβλιογραφία για την έκφραση ενός προβλήματος κατανομής πολλών προς πολλούς στο πλαίσιο δημιουργίας και επεξεργασίας ανθρώπινης κίνησης.

Η εργασία αυτή θα αρχίσει να ξεδιπλώνεται αρχικά με την εμβάθυνση στα θεμελιώδη των μοντέλων διάχυσης και την αιτιολόγηση της υπεροχής τους σε σύγκριση με άλλες σύγχρονες βαθιές αρχιτεκτονικές στο πλαίσιο της παραγωγής ανθρώπινης κίνησης στο κεφάλαιο 2. Στο Κεφάλαιο 3, θα αναλυθούν σε βάθος η αναπαράσταση της κίνησης (π.χ. μεταφορά, περιστροφή, τοπολογία σκελετού) που χρησιμοποιείται ως είσοδος για αυτού του είδους τις αρχιτεκτονικές, τα σύνολα δεδομένων που χρησιμοποιούνται για την εκπαίδευση των υπό αξιολόγηση δικτύων καθώς και ο κομβικός ρόλος του μοντέλου **ΣΜΠΛ**[41]. Αφού ολοκληρωθεί το κομμάτι της παρουσίασης του θεωρητικού υπόβαθρου που απαιτείται για την πλήρη κατανόηση του πλαισίου της παρούσας εργασίας, θα αναπτυχθούν αναλυτικά οι αρχιτεκτονικές των νευρωνικών δικτύων που χρησιμοποιήθηκαν σε περιβάλλοντα παραγωγής και επεξεργασίας ανθρώπινης κίνησης στο κεφάλαιο 4. Τέλος, στο Κεφάλαιο 5 θα παρουσιαστούν τα πειράματα που διεξήχθησαν με στόχο την εκτέλεση μιας στρογγυλής συγκριτικής αξιολόγησης μεταξύ διαφορετικών μεθόδων προσανατολισμένων στη διάχυση [42], διεξάγοντας τόσο ποιοτική όσο και ποσοτική αξιολόγηση. Τα συνολικά συμπεράσματα θα συνοψιστούν στο κεφάλαιο 6 μαζί με την πρόταση πιθανών μελλοντικών ερευνητικών κατευθύνσεων.

Chapter 2

Diffusion Models - Theoretical background

2.1 Intuition behind Diffusion Models

The foundation of diffusion models and their introduction in deep learning as described in [43] is based on the idea that diffusion processes can be strategically employed to decompose the inherent structural components present in the data distribution we aim to model. To circumscribe this theoretical framework more concretely, consider a hypothetical scenario where a container filled with water has a small amount of colored dye introduced. If we consider the concentration of dye molecules as a representation of a probability distribution, the principal goal of a generative model is to gain a thorough understanding of this probabilistic structure. At first glance, it is crucial that we recognize that accomplishing this venture could prove to be quite challenging.

However, even in cases where we are unable to directly create a model to clarify the inherent structure of our data distribution, there is a feasible alternative approach. This approach entails the transition from our complicated and, more often than not, high-dimensional data distribution into a significantly simplified distribution that can be easily modeled and basically encloses the main intrinsic features of the initial distribution.

In the context of this physical analogy, if the diffusion process is allowed to unfold over a sufficiently extended period, the dye molecules will eventually disperse uniformly throughout the container, resulting in a state of equilibrium that can be described by a distribution as simple as a uniform distribution. Initially, the utility of this transformation may not be immediately apparent. However, what can actually seem useful is the consideration of the contingency of reversing this diffusion process over time. Thus, initiating from a uniform distribution and executing the exact reverse process would provide us with the possibility to recreate the original data distribution. Reversing the process would, of course, be deemed as impossible by physicists as it would entail the violation of the **second thermodynamic law** in real world. However, the outburst observed in the deep learning domain yielded a potential solution to address these challenges.

Diffusion models in the context of deep learning were first used for image generation[1] and refer to a specific class of generative models that aim to model the process of generating realistic images by simulating the process analyzed above. These models, such as the Denoising Diffusion Probabilistic Models (DDPM) or Noise-Contrastive Estimation[44] (NCE) models, leverage the diffusion process as conceptualized in nature to generate diverse and realistic images. The fundamental idea behind diffusion models is to generate images by iteratively adding noise to a starting image. Instead of directly specifying pixel values, these models start with a simple distribution (e.g., Gaussian) and iteratively transform it to generate a realistic image. This process is analogous to the diffusion of heat or particles in a container over time. Diffusion models perform a series of transformations on the initial noise distribution to gradually refine it into an image. At each step, a fraction of the noise is added, leading to a diffusion process that converges towards a complex, high-dimensional data distribution representing realistic images.

In direct analogy to the image generation domain, the motion generation task operates in the same philosophy. The motion representation features - which usually differ from method to method but mostly include joint positions, joint rotations or joint velocities and combinations of these - are gradually noised more and more. The reverse process, expectedly, samples noise from a distribution, performs a set of transformations from space to space finally yielding motion representation feature values that correspond to a coherent human motion.

To expound upon the noising process, a pivotal catalyst in the functionality of diffusion models, an exploration of the temperature parameter becomes imperative. Termed as such due to its role in modulating the intensity of the noising process, akin to how ambient air temperature dictates the pace of a pill's diffusion within a liquid receptacle, the temperature parameter governs the amplitude of noise infusion at each procedural step. Higher temperatures correspond to more significant noise, and as the process progresses, the temperature decreases. This temperature scheduling allows for a balance between exploration and exploitation during the generation process. In deep learning diffusion models, the process of diffusion is learned from data. The model learns the dynamics of the diffusion process, capturing the intricate relationships and dependencies between pixels in image generation or between joint rotation features in human motion generation. This learned diffusion process enables the generation of diverse and realistic *images*.

Diffusion models are effective in handling complex data distributions, including those found in natural human motions. The learned diffusion process captures the intricate dependencies between joint features describing human motions, allowing the model to generate high-quality animations with realistic poses. Diffusion models stand as the state-of-the-art method in both in human motion synthesis and editing. They enable the generation of diverse samples from a learned distribution, making them suitable for tasks such as **motion inpainting in both time and space** and style transfer.

In summary, diffusion models in deep learning for image generation leverage the principles of noise-driven diffusion processes to iteratively refine a noise distribution into realistic images. The learned dynamics, temperature scheduling, and training objectives make diffusion models a powerful approach for capturing and generating complex image data distributions.

2.2 The Diffusion Process

As mentioned above, a Diffusion Model consists of a **forward process (or diffusion process)**, in which a datum is progressively noised, and a **reverse process (or reverse diffusion process)**, in which noise is transformed back into a sample from the target distribution. In other words, diffusion models work by destroying training data through the successive addition of Gaussian noise, and then learning to recover the data by reversing this noising process. After training, we can use the Diffusion Model to generate data by simply passing randomly sampled noise through the learned denoising process. Hence, a meticulous elucidation of both processes is imperative to later delve into the intricacies of architectural variations and optimization techniques when it comes to diffusion models. Commencing our examination with the formerly alluded process, specifically the forward process, is our starting point.

2.2.1 Forward Process

Diffusion models can be seen as latent variable models similarly to e.g. variational autoencoders[45]. Latent means that we are referring to a hidden continuous feature space. In In practice, they are formulated using a Markov chain of T steps.The forward diffusion process is the Markov chain of diffusion steps in which we slowly and randomly add noise to the original data. A Markov chain[46] means that each step only depends on the previous one, which is a mild assumption as the reality is that causality is being propagated across multiple states in the Markov Chain.

Given a data point x_0 sampled from a real data distribution $q(x)(x_0 \leftarrow q(x_0))$, in the forward diffusion process, noise is gradually added to a clean datum, until its structure is destroyed and the datum is transformed into pure noise. In other words, the forward process is characterised by a Markov chain, which generates a sequence of random variables $\mathbf{x}_t$ with distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1})$. The noise added to the original datum at each Markov Chain step is sampled from a *Gaussian* distribution formulated as follows:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \mu_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \Sigma_t = \beta_t\mathbf{I}) \tag{2.1}$$

Since we are in the multi-dimensional scenario, $\mathbf{I}$ is the identity matrix, indicating that each dimension has the same standard deviation β_t. As long as $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the normal distribution described above and we have adopted the Markovian assumption for our chain, the posterior joint distribution on the latent variables can be written in closed form as can be seen in 2.2:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{i=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{2.2}$$

However, this form entails a heavy burden in terms of computational load if the number of timesteps in the Markov Chain is large. Thus, to simplify the expression, a **reparameterization trick** is employed aiming at turning the posterior distribution into a tractable expression as elaborated in [47]. More specifically, the idea behind this trick could be verbally summarized as *transforming a sample from a fixed, known distribution*

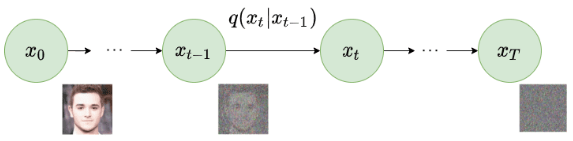


Figure 2.1. *Forward Diffusion Process modified by [1]*

to a sample from $q(z)$. If we consider the Gaussian distribution, we can express z with respect to a fixed ϵ, where $\epsilon \sim \mathcal{N}(0, 1)$, as follows:

$$z = \mu + \sigma\epsilon \tag{2.3}$$

The epsilon term introduces the necessary stochastic part in equation 2.3. Let $a_t = 1 - \beta_t$ and $\bar{a}_t = \prod_{s=0}^{T} a_s$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the reparameterization trick yields, in a recursive manner:

$$\begin{aligned}\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon_{t-1} \\ &= \sqrt{a_t}\mathbf{x}_{t-2} + \sqrt{1-\beta_t}\epsilon_{t-2} \\ &= \ldots \\ &= \sqrt{\bar{a}_t}\mathbf{x}_0 + \sqrt{1-\bar{a}_t}\epsilon_0\end{aligned} \tag{2.4}$$

Consequently, employing this property provides us with the capability to produce one sample $\mathbf{x}_t$ using the following distribution:

$$\mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{a}_t}\mathbf{x}_0, (1-\bar{a}_t)\mathbf{I}) \tag{2.5}$$

As can be easily deduced, the aforementioned process allows for tractable closed-form sampling at any timestep accelerating the sampling process analyzed in section 2.3 by orders of magnitude. This owed to the fact that, as long as β_t is a hyperparameter, we can precompute (namely, before the sampling process begins) the quantities a_t and $\bar{a}_t$ for a number of timesteps equal to the number the sampling process is about to be executed, which, subsequently means that we sample noise at any timestep **t** and obtain $\mathbf{x}_t$ in one shot. Hence, we can sample our latent variable $\mathbf{x}_t$ at any arbitrary timestep we might desire.

2.2.2 Backward Process

It is the process of training a neural network to recover the original data by reversing the noising process applied in the forward pass described in 2.2.1. As extensively analyzed in [43], theory validates that if consider that the forward diffusion process is being evolved for a very big number of timesteps ($T \to \infty$), which is equivalent to assigning β_t a value small enough, the latent variable $\mathbf{x}_T$ could be nearly considered as an **isotropic** Gaussian distribution, namely a Gaussian distribution for which $\Sigma = \sigma^2\mathbf{I}$. Therefore, if we were somehow capable of learning the reverse distribution $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$, we would be able to

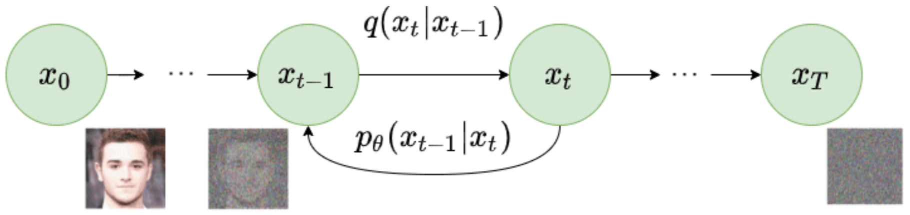


Figure 2.2. *Reverse Diffusion Process modified by [1]*

sample $\mathbf{x}_T$ from $\mathcal{N}(0, \mathbf{I})$, simulate the reverse process and acquire a sample from $q(x_0)$ generating a totally novel data point 'belonging' to the original data distribution.

In practice, though, the calculation of the $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ distribution is infeasible as it would require the whole training dataset to obtain reliable statistical estimates for the data distribution. What can be practically done to approximate $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is that we use a parameterized model p_∂, e.g. a neural network, which can be assumed as Gaussion as long as β_t is small enough permitting the hypothesis that $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is a Gaussian distribution. Thus, the model p_∂ can be implemented by just modelling the **mean and variance parameters** as follows:

$$p_\partial(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\partial(\mathbf{x}_t, t), \Sigma_\partial(\mathbf{x}_t, t)) \tag{2.6}$$

Hence, we end up training the network with our actual objective reduced to predicting the mean and variance for each timestep, namely the quantities denoted as $\mu_\partial(\mathbf{x}_t, t)$ and $\Sigma_\partial(\mathbf{x}_t, t)$). This task is accomplished by applying the reverse formula for all timesteps. In further detail, we can transit from $\mathbf{x}_T$ to the data distribution as per:

$$p_\partial(\mathbf{x}_{0:T}) = p_\partial(\mathbf{x}_T) \prod_{i=1}^{T} p_\partial(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{2.7}$$

By additionally conditioning the model on timestep t (positional encoding), the model is capable of learning the Gaussian parameters, namely the mean and covariance matrices, for each individual timestep.

2.3 Sampling

2.3.1 DDPM Sampling

As discussed in the previous chapter, the DDPM is a generative model that belongs to the family of diffusion models. The basic idea is to model the data generation process as a diffusion process, where the true data distribution is transformed through a sequence of noise levels. Denoising autoencoders are used to model the conditional distributions at each step of the diffusion process. DDPM was practically the parent model of all the other architectures which normally followed later, comprising the foundation for an entire family of generational networks.

The goal is to teach a model to reverse the noising process so that we can generate images -or motions in our own paradigm- given the noise randomly sampled from a

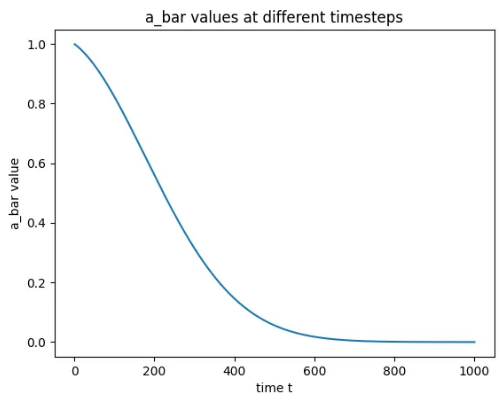


Figure 2.3. *Noise scheduling in DDPMs. Source [2]*

Gaussian Distribution. To delve into further detail, we should initiate our analysis from the equations 2.8 and 2.9:

$$a_t = 1 - \beta_t \tag{2.8}$$

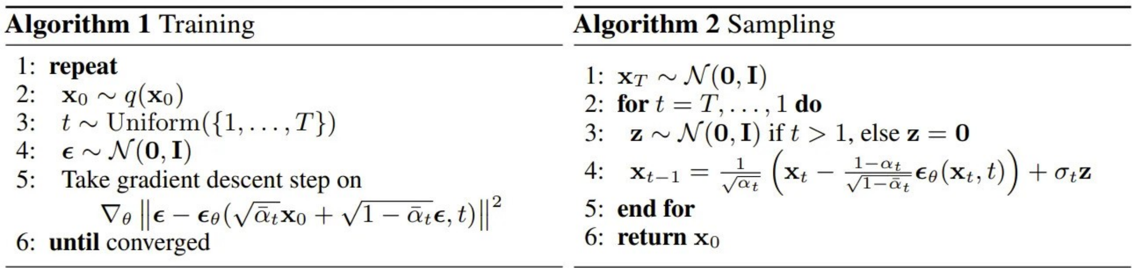
$$\bar{a}_t = \prod_{s=0}^{T} a_s \tag{2.9}$$

and explain what they stand for in practical terms. β_t is called a noise scheduler. The authors of the DDPM paper use a scheduler altering the values of β_t in linear fashion starting from the value of 10^{-4} at $t = 0$ and ending up at 0.02 at $t = \mathbf{T}$. These values function somewhat akin to percentages, dictating the relative magnitude of noise incorporated at a given moment in time (in moment t relative to $t - 1$). As apparent from equations 2.9, the amount of noise at each timestep increases exponentially while the percent of the original image decreases exponentially. Figure 2.3 shows the values of $\bar{a}_t$ over evolving timesteps of the diffusion process with the value of T set equal to 1000.

The founders of the DDPMs found out that **learning variances during the backward diffusion** process may lead to unstable training and poorer sample quality compared to fixed variances. As a result of this, they preferred to maintain the value of the covariance matrix Σ_ϑ constant and equal to β_t, since β_t is the noise variance at each timestep t, to avoid this undesirable effect. As long as variance is kept constant, we basically need to only predict the mean of the distribution. Equivalently, the estimation can be reduced down to predicting the noise ϵ, that was sampled from the normal distribution and added to the sample through the reparameterization trick 2.3. The authors found that predicting the noise yielded more stable training. Thus, when it comes to reparemeterizing $\mu_\vartheta(\mathbf{x}_t, t)$ of the reverse process distribution, the network can predict the **normal** noise sample ϵ (from a unit-variance distribution), which has been added to the sample x_0:

$$x_0 = \frac{1}{\sqrt{\bar{a}_t}}(x_t - \sqrt{1 - \bar{a}_t}\epsilon) \tag{2.10}$$

Hence, the final equation for $\mu_\vartheta(\mathbf{x}_t, t)$ is derived as follows and after combination with



Algorithm 1 Training

```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \text{Uniform}(\{1, \ldots, T\})$
4:   $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:   Take gradient descent step on
         $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: until converged
```

Algorithm 2 Sampling

```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \ldots, 1$ do
3:   $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4:   $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: end for
6: return $\mathbf{x}_0$
```

Figure 2.4. *Training and Sampling algorithms in DDPM models. Source:[1]*

equation 2.10:

$$\begin{aligned}\tilde{\mu}_\partial &= \frac{\sqrt{\bar{a}_{t-1}}\beta_t}{1-\bar{a}_t}x_0 + \frac{\sqrt{a_t}(1-\bar{a}_{t-1})}{1-\bar{a}_t}x_t \\ &= \frac{\sqrt{\bar{a}_{t-1}}\beta_t}{1-\bar{a}_t}\left(\frac{1}{\sqrt{\bar{a}_t}}(x_t - \sqrt{1-\bar{a}_t}\epsilon)\right) + \frac{\sqrt{a_t}(1-\bar{a}_{t-1})}{1-\bar{a}_t}x_t \\ &= \frac{1}{\sqrt{a_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}}\epsilon)\end{aligned} \tag{2.11}$$

which is the key equation **DDPM** used for sampling.

The sampling process practically executes the reverse diffusion process using the following line of reasoning: the generation process is initiated at $t = T$ by sampling from the last diffusion step $x_T \sim \mathcal{N}(0, 1)$ which is modeled by a normal Gaussian, obviously. This happens repeatedly until $t = 0$ is reached. The network makes a prediction of noise in the sample $\tilde{\epsilon} = p_\partial(x_t, t)$ and then approximates the mean of the process at $t - 1$ using 2.12.

$$\tilde{\mu}_\partial = \frac{1}{\sqrt{a_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}}\tilde{\epsilon}) \tag{2.12}$$

Hence, the next sample at $t - 1$ is sampled from the Gaussian distribution as below:

$$x_{t-1} \sim \mathcal{N}(\tilde{\mu}_\partial, \sigma_t^2 I) \tag{2.13}$$

until x_0 is reached, as explicitly described in the sampling algorithm outline in figure 2.4. For notation purposes, we will define an additional variable called transition variance $\tilde{\beta}_t$:

$$\tilde{\beta}_t = \sigma_t^2 \tag{2.14}$$

Since we just have to predict the noise added, we can use the MSE loss between the predicted noise and the actual noise added to the image.

2.3.2 DDIM Sampling

A notable challenge associated with the DDPM process lies in the time required for image generation, at first place, and sample generation, in general, during inference time. While DDPM models can yield visually impressive samples, the necessity for 1,000 model passes to generate a single outcome poses a considerable drawback. Although processing an image through the model 1,000 times might be accomplished swiftly on a GPU, the

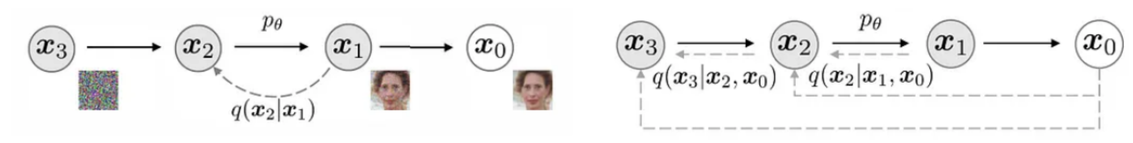


Figure 2.5. *Graphical models for diffusion (left) and non-Markovian (right) inference models. Source [2]*

duration substantially extends when executed on a CPU. Consequently, there arises an imperative to devise strategies to expedite the generation process.

The DDIM paper[48] introduces a way to speed up the generation process with the least possible tradeoff between sample quality and generation time during the inference. It achieves this goal by redefining the diffusion process as a non-Markovian process as explicitly displayed in figure 2.5. The left figure depicts the graphical model describing the original DDPM paper which requires all past denoising steps from time T to time $t-1$ to finally obtain the resulting denoised sample at time t. DDPMs are conceptualized as Markov Chains, implying that the generation of a sample at a given time, t, is contingent upon the completion of the entire chain preceding t. The DDIM paper proposes a method to make the process non-Markovian (in the right figure), enabling the bypassing of steps in the denoising process, eliminating the necessity to visit all preceding states before reaching the current state. The key feature of DDIMs compared to the DDPMs is they can be instantly applied after training a model. Consequently, DDPM models can seamlessly transition into DDIMs without the need for retraining an entirely new model.

The fact that the stage of training may as well be omitted in the context of making the transition from a DDPM to a DDIM, only the reverse diffusion process needs to be individually modelled. To this end, we begin our analysis for the revrse diffusion process in **DDIM** by highlighting similarities and differences between the new approach and DDPM. The reverse process in DDPM endeavors to traverse the diffusion chain in the opposite direction, encompassing T steps in the reverse order. However, as shown in eq. 2.10, the reverse process involves an approximation of the clean sample x_0. Provided we substitute $t-1$ for t in 2.5, we obtain:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{a}_{t-1}}\mathbf{x}_0, (1-\bar{a}_{t-1})\mathbf{I}) \tag{2.15}$$

which subsequently yields:

$$x_{t-1} \leftarrow \sqrt{\bar{a}_{t-1}}x_0 + \sqrt{1-\bar{a}_{t-1}}\epsilon_{t-1} \tag{2.16}$$

Based on a specific ϵ_t measured at the previous step t, the equation can be reformulated as:

$$x_{t-1} = \sqrt{\bar{a}_{t-1}}(\frac{x_t - \sqrt{1-\bar{a}_{t-1}}\epsilon_\partial^{(t)}(x_t)}{\bar{a}_t}) + \sqrt{1-\bar{a}_{t-1}-\sigma_t^2}\epsilon_\partial^{(t)} + \sigma_t\epsilon_t \tag{2.17}$$

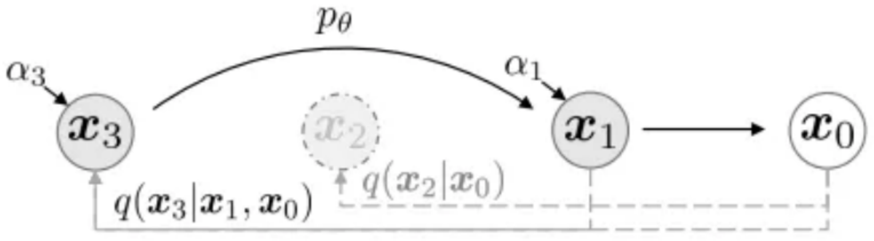


Figure 2.6. *Non-Markovian reverse and forward process. Source [2]*

Combining 2.15, 2.16 and 2.4 we end up with:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_0, \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{a}_{t-1}}x_0 + \sqrt{1 - \bar{a}_{t-1} - \sigma_t^2}\frac{x_t - \sqrt{\bar{a}_t}x_0}{\sqrt{1 - \bar{a}_t}}, \sigma_t^2\mathbf{I}) \tag{2.18}$$

where

$$\sigma_t^2 = \tilde{\beta}_t = \sqrt{\frac{1 - \bar{a}_{t-1}}{1 - \bar{a}_t}\beta_t} \tag{2.19}$$

In the classical DDPM setting, this configuration is established. To transition to DDIM, a parameter η is introduced, and the transformation is characterized by the following equation:

$$\sigma_t^2 = \eta\tilde{\beta}_t \tag{2.20}$$

The newly introduced parameter is used to control the magnitude of the stochastic component. Setting $\eta = 0$ appears to be particularly beneficial when fewer steps of the reverse process are applied and that specific type of process is known as **Denoising Diffusion Implicit Model**. The above formulation is still consistent with DDPM when $\eta = 1$. We obtain a DDIM when $\sigma = 0$, thus equation 2.17 is transformed into:

$$x_{t-1} = \sqrt{\bar{a}_{t-1}}(\frac{x_t - \sqrt{1 - \bar{a}_{t-1}}\epsilon_\vartheta^{(t)}}{\bar{a}_t}) + \sqrt{1 - \bar{a}_{t-1}}\epsilon_\vartheta^{(t)} \tag{2.21}$$

Observe the absence of supplementary noise in the data. This exemplifies the essence of DDIM's methodology. When $\sigma = 0$, the denoising process turns entirely deterministic, with the sole source of noise being the initial noise at x_0, as no additional noise is introduced during the denoising process.

In the absence of noise in the reverse process, determinism prevails, obviating the need for a Markov Chain. Markov Chains are tailored for probabilistic processes, and in this scenario, a non-Markovian process proves more suitable, affording the flexibility to bypass steps in the sequence, as demonstrated in diagram 2.6.

The diagram above presents an exemplary process during which we have the capability to skip from step x_3 to x_1 by totally omitting x_2. The new diffusion process is modelled as a subsequence, τ, which is a subset of the original diffusion sequence. For instance, we could sample every other diffusion step in the diffusion process to get a subsequence of $\tau = [0, 2, 4, \ldots, T-2, T]$.

Regarding the value of parameter η in 2.20, what is qualified, ultimately, as an ideal choice to model the variance of the diffusion model as an **interpolation between DDIMs and DDPMs** by opting a value between 0 and 1 for the parameter - we need to inject a degree of stochasticity in the process, but not to an utmost extent. It can generally be assumed that DDIM:

- Offers better sample quality at fewer steps.
- Allows for deterministic matching between the starting noise and the generated sample.
- Performs worse than DDPM for large numbers of steps (such as 1000).

A detailed examination of the final point enhances practical understanding when implementing diffusion models. DDIMs exhibit superior performance to DDPMs when the number of steps taken is below the original T steps. While DDPM excels at the original 1,000 steps, DDIM closely parallels this performance when generating samples with significantly fewer steps. This introduces a discernible tradeoff between image quality and generation time when using a DDIM, a dynamic not present in the original DDPM. Consequently, the application of DDIM allows for the generation of high-quality samples with a substantially reduced number of steps.

2.4 Conditional Generation: Guided Diffusion

A pivotal element in image generation and, in broader terms, motion generation, involves conditioning the sampling process to manipulate the generated samples, a concept often denoted as **guided diffusion**. Certain methodologies extend this idea by incorporating image embeddings (or text/audio embeddings in the context of motion generation) into the diffusion process to "guide" the generation. In mathematical terms, guidance entails conditioning a prior data distribution $p(\mathbf{x})$ with a condition $\mathbf{y}$, i.e. the class label or an image/text embedding, resulting in $p(\mathbf{x}|\mathbf{y})$. To transform a diffusion model p_∂ into a conditional diffusion model, we can introduce conditioning information $\mathbf{y}$ at each diffusion step. In this setting, equation 2.7 is reformulated as follows:

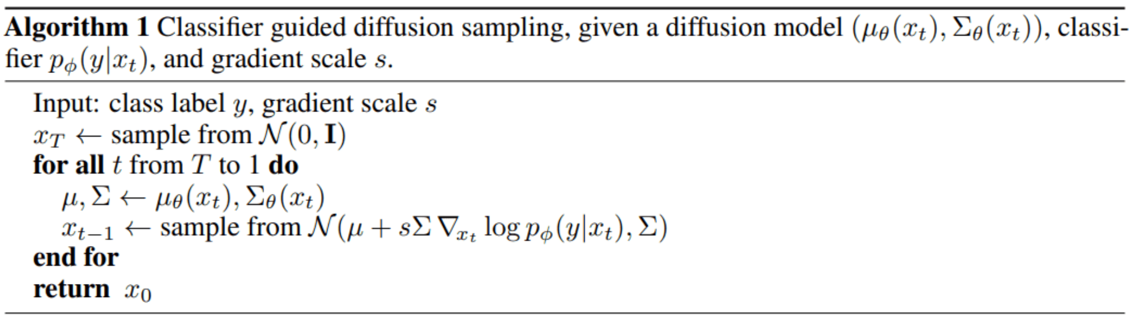
$$p_\partial(\mathbf{x}_{0:T}|\mathbf{y}) = p_\partial(\mathbf{x}_T)\prod_{i=1}^{T} p_\partial(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}) \tag{2.22}$$

The incorporation of conditioning at each timestep offers a plausible rationale for the impressive generation of samples based on a text prompt. In general, guided diffusion models aim to learn $\nabla log p_\partial(\mathbf{x}_t|\mathbf{y})$. Hence, using the Bayes rule, we can reformulate to:

$$\nabla_{x_t} log p_\partial(\mathbf{x}_t|\mathbf{y}) = \nabla_{x_t} log \frac{p_\partial(\mathbf{y}|\mathbf{x_t})p_\partial(\mathbf{x}_t)}{p_\partial(\mathbf{y})} = \nabla_{x_t} log(p_\partial(\mathbf{y}|\mathbf{x_t})) + \nabla_{x_t} log(p_\partial(\mathbf{x}_t)) \tag{2.23}$$

$p_\partial(\mathbf{y})$ is removed since the gradient operator $\nabla_{\mathbf{x_t}}$ refers only to $\mathbf{x}_t$, thus the gradient for y is zeroed out. By adding a guidance scalar term s, we obtain:

$$\nabla_{x_t} log p_\partial(\mathbf{x}_t|\mathbf{y}) = \nabla_{x_t} log(p_\partial(\mathbf{x}_t)) + s \cdot \nabla_{x_t} log(p_\partial(\mathbf{y}|\mathbf{x_t})) \tag{2.24}$$



Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

```
Input: class label $y$, gradient scale $s$
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all $t$ from $T$ to 1 do
    $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
    $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma\,\nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
end for
return $x_0$
```

Figure 2.7. *Algorithm of classifier guided diffusion DDPM sampling. Source [3]*

By adopting this formulation, it becomes pertinent to differentiate between **classifier-guided** and **classifier-free guidance**. Subsequently, we will introduce two categories of methods, each belonging to a distinct family, with the common objective of incorporating label information. These two categories will be elucidated and thoroughly examined in the subsequent sections.

2.4.1 Classifier Guidance

Classifier guidance was introduced in [3] and effectively relies on a classifier to direct the diffusion model towards generating samples belonging to a specified class. [3] demonstrated that we can employ a supplementary model, a classifier $f_\phi(\mathbf{y}|\mathbf{x}_t, t)$ to guide the diffusion toward the target class y during training. To achieve that, we can train a classifier $f_\phi(\mathbf{y}|\mathbf{x}_t, t)$ on the noisy sample $\mathbf{x}_t$ to predict its class y. Then we can use the gradients $\nabla log(f_\phi(\mathbf{y}|\mathbf{x}_t))$ to guide the diffusion process.

In practical terms, the diffusion models analyzed in the preceding sections are tweaked accordingly to match this emerging need: a class-conditional diffusion model with mean $\mu_\partial(\mathbf{x}_t|y)$ and variance $\Sigma_\partial(\mathbf{x}_t|y)$. Provided that:

$$p_\partial \sim \mathcal{N}(\mu_\partial, \Sigma_\partial) \tag{2.25}$$

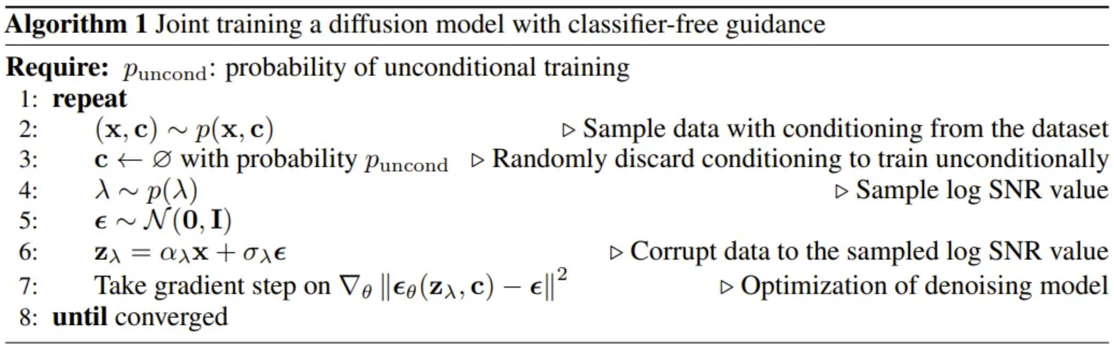
we can utilize the guidance formulation elucidated in the previous section (i.e. eq.2.24) to demonstrate that the mean is perturbed by the gradients of $log(f_\phi(\mathbf{y}|\mathbf{x}_t))$ of class y yielding:

$$\hat{\mu}(\mathbf{x}_t|y) = \mu_\partial(\mathbf{x}_t|y) + s \cdot \Sigma_\partial(\mathbf{x}_t|y)\nabla_{x_t} log(p_\partial(\mathbf{y}|\mathbf{x_t}, t) \tag{2.26}$$

In [49], the authors expanded on the idea described by 2.26 and employ CLIP embeddings to guide the diffusion. CLIP, as suggested in [6], consists of an image encoder g and a text encoder h which aim at producing an image and text embeddings $g(\mathbf{x}_t)$ and $h(c)$, respectively, wherein c is the text caption. Therefore, we can perturb the gradients with the two embeddings' dot product by transforming (2.26) into:

$$\hat{\mu}(\mathbf{x}_t|c) = \mu_\partial(\mathbf{x}_t|c) + s \cdot \Sigma_\partial(\mathbf{x}_t|c)\nabla_{x_t} g(\mathbf{x}_t)h(c) \tag{2.27}$$

As a result, they manage to steer the generation process toward a user-defined text caption. The algorithm implemented the described process is displayed in figure 2.7. It



```
Algorithm 1 Joint training a diffusion model with classifier-free guidance
Require: $p_{\text{uncond}}$: probability of unconditional training
1: repeat
2:     $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$                    ▷ Sample data with conditioning from the dataset
3:     $\mathbf{c} \leftarrow \varnothing$ with probability $p_{\text{uncond}}$   ▷ Randomly discard conditioning to train unconditionally
4:     $\lambda \sim p(\lambda)$                                     ▷ Sample log SNR value
5:     $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
6:     $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\epsilon}$                     ▷ Corrupt data to the sampled log SNR value
7:     Take gradient step on $\nabla_\theta \left\| \boldsymbol{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \boldsymbol{\epsilon} \right\|^2$   ▷ Optimization of denoising model
8: until converged
```

Figure 2.8. *Training a diffusion model for classifier-free guidance. Source [2]*

is important to clarify that the authors of [3] suggested different algorithms for classifier guidance in DDPM and DDIM settings respectively. In figure 2.7, the algorithm for DDPM only is displayed.

2.4.2 Classifier-Free Guidance

Classifier-Free Guidance refines classifier guidance by dispensing with the classifier while still furnishing class guidance to the model. Assuming we can integrate class information into our diffusion model, we can configure the model to generate samples both with and without class distinctions. Using the same formulation as in 2.24, we can define a classifier-free guided diffusion model as:

$$\nabla logp(\mathbf{x}_t|\mathbf{y}) = s \cdot \nabla log(p(\mathbf{x_t}|\mathbf{y})) + (1 - s) \cdot logp(\mathbf{x_t}) \tag{2.28}$$

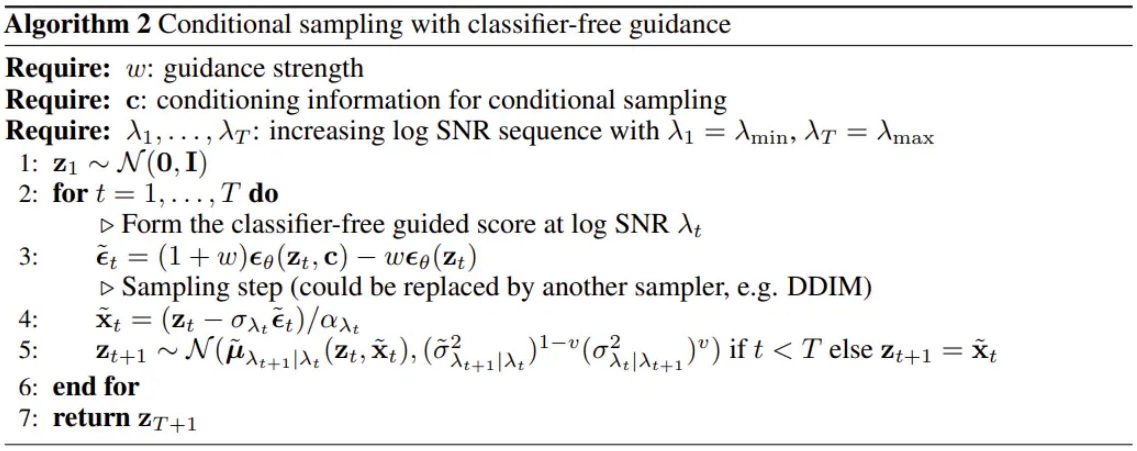
Guidance can be achieved without a second classifier model as proposed by [50]. Instead of training a separate classifier, a conditional diffusion model $\epsilon_\partial(\mathbf{x}_t|\mathbf{y})$ was trained jointly with an unconditional model $\epsilon_\partial(\mathbf{x}_t|\mathbf{0})$. In fact, they employ an identical neural network for this purpose. During the training phase, they stochastically set the class variable y to 0, thereby exposing the model to both the conditional and unconditional setups as is obvious in the equation below:

$$\hat{\epsilon}_\partial(\mathbf{x}_t|\mathbf{y}) = s \cdot \epsilon_\partial(\mathbf{x}_t|\mathbf{y}) + (1 - s) \cdot \epsilon_\partial(\mathbf{x}_t|\mathbf{0}) = \epsilon_\partial(\mathbf{x}_t|\mathbf{0}) + s \cdot \epsilon_\partial(\mathbf{x}_t|\mathbf{y}) - s \cdot \epsilon_\partial(\mathbf{x}_t|\mathbf{0}) \tag{2.29}$$

This admittedly paradoxical process has two major advantages:

- It employs a solitary model to direct the diffusion process.
- It facilitates the guidance process, particularly when conditioning on information that proves challenging to predict using a classifier, such as text embeddings.

As intuitively inferred, instead of a class, y may as well be a text embedding. In other words, to add classifier-free guidance to our diffusion model, all that needs to be done is train the model to generate samples with class information and without class information as visualized in figure 2.8.



Algorithm 2 Conditional sampling with classifier-free guidance

```
Require: $w$: guidance strength
Require: $\mathbf{c}$: conditioning information for conditional sampling
Require: $\lambda_1, \dots, \lambda_T$: increasing log SNR sequence with $\lambda_1 = \lambda_{\min}$, $\lambda_T = \lambda_{\max}$
1: $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = 1, \dots, T$ do
       ▷ Form the classifier-free guided score at log SNR $\lambda_t$
3:     $\tilde{\boldsymbol{\epsilon}}_t = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) - w\boldsymbol{\epsilon}_\theta(\mathbf{z}_t)$
       ▷ Sampling step (could be replaced by another sampler, e.g. DDIM)
4:     $\tilde{\mathbf{x}}_t = (\mathbf{z}_t - \sigma_{\lambda_t}\tilde{\boldsymbol{\epsilon}}_t)/\alpha_{\lambda_t}$
5:     $\mathbf{z}_{t+1} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\lambda_{t+1}|\lambda_t}(\mathbf{z}_t, \tilde{\mathbf{x}}_t), (\tilde{\sigma}^2_{\lambda_{t+1}|\lambda_t})^{1-v}(\sigma^2_{\lambda_t|\lambda_{t+1}})^v)$ if $t < T$ else $\mathbf{z}_{t+1} = \tilde{\mathbf{x}}_t$
6: end for
7: return $\mathbf{z}_{T+1}$
```

Figure 2.9. *Training a diffusion model for classifier-free guidance. Source [2]*

The training loop undergoes a slight modification to effectively train the model in generating output samples with and without class information. p_uncond stands for the probability of replacing a class with a null class to force the model to learn how to generate 'images' without class information. Further information on the training loop operation and, more specifically, on loss optimization will be provided in section 2.5. However, to concisely explain the algorithm exhibited in 2.8, we would like to to delineate the process explicitly, step by step, while simultaneously clarifying notation:

1. Normal loop over epochs.
2. Sample images x and their respective classes c.
3. With a probability p_uncond, we turn some of the classes into null classes.
4. Sample timestep t.
5. Sample noise ϵ from a normal distribution.
6. Create the noisy sample z_t.
7. Train the model ϵ_θ using normal MSE loss between the predicted noise and real noise.

It is noteworthy that the training loop closely mirrors the DDPM training loop, with the primary distinction being the inclusion of class information, alongside the introduction of a probability mechanism to nullify the class information.

After training, the generation process, namely the sampling stage, is also slightly altered as can be seen in figure 2.9 and summarized in the following bullets:

- Sample noise from a normal distribution.
- Normal sampling loop. Loop from time $t = 1$ to time $t = T$.
- Get the noise output from the model given the null class $\epsilon_\theta(z_t)$ and given the class $\epsilon_\theta(z_t, c)$ of the image you want to generate. Interpolate these noise predictions to obtain the new noise prediction $\tilde{\epsilon}_t$.

- Calculate the next step sampling from a normal distribution which uses the interpolated noise prediction rather than the usual noise prediction.

The sampling loop closely resembles the DDPM sampling loop, with a single modification:

$$\tilde{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) = (1 + w) \cdot \epsilon_\theta(\mathbf{z}_\lambda, \mathbf{c}) - w \cdot \epsilon_\theta(\mathbf{z}_\lambda) \tag{2.30}$$

As apparent by carefully looking at 2.30, the noise prediction requires two forward passes of the same image z_t. One forward pass calculates the predicted noise not conditioned on a desired class, and the other calculates the predicted noise conditioned on the desired class information.

When $w = 0$, the model is a normal DDPM with class information. When $w > 0$, we utilize classifier-free guidance. The goal is to produce an image of class c. The idea is that the class-informed model will generate an output about the class we want to generate, but the class signal could be stronger. To enhance the signal derived from the class information, we can diminish the influence of the model without class information, which would typically generate random samples. As w increases, we progressively eliminate more "null" samples. Theoretically, by removing additional information associated with the null class, we augment the information available for the desired class.

This approach proves effective up to a certain threshold. A w value falling within the range of $[5, 20]$ yields satisfactory results according to literature, but this may vary depending on the context of the generated samples. However, employing a high w value removes an excessive amount of signal from the sample, essentially resulting in the production of random noise due to the significant removal of signal. Here emerges the necessity to highlight a drawback of classifier guidance, which involves a tradeoff between FID and IS. FID assesses both the quality and mode coverage of the latent space, whereas IS focuses solely on 'image' quality.

Observing the trend, as w increases (indicating greater guidance), the FID score decreases while the IS score increases. This implies that with higher w values, *'images'* exhibit higher quality but demonstrate reduced variance.

2.5 Training Diffusion Models

The training procedure, in a general scope, is realized using the Algorithm 1 demonstrated in 2.4 and is implemented by looping over epochs through the steps enumerated below for each training step.

1. Use **forward process** to generate a sample $x_t \sim q(x_t|x_0)$ for a t sampled uniformly at $[1, T]$.

    - Sample timestep t from a uniform distribution $t \sim \mathcal{U}(1, T)$.
    - Sample ϵ from a normal Gaussian $\epsilon \sim \mathcal{N}(0, 1)$.
    - Compute noisy input sample x_t for training via $x_t = \sqrt{\bar{a}_t} x_0 + \sqrt{1 - \bar{a}_t}\epsilon$

2. **Compute the approximation of noise** $\hat{\epsilon}_t = p_\partial(x_t, t)$ using the model with parameters ∂.

3. Minimize the error between ϵ_t and $\hat{\epsilon}_t$ by optimizing parameters ∂.

Note that we do not have to model the entire diffusion process as a single process, but rather we can model each individual timestep individually. Doing this will speed up training and will likely lead to a more stable training setting. If we sample the value of t uniformly for each training image, the model should be able to learn how to model all values of t while learning how to model the real image distribution.

To this point, it is essential to provide a few more details regarding how exactly a diffusion model is being optimized during training, which loss functions are being utilized etc. During training, a diffusion model is optimized by minimizing a loss function that captures the discrepancy between the generated images and the target distribution. Typically, the loss function includes components that encourage the generated images to resemble the true data distribution while also promoting diversity and realism.

Delving into further technical details, if we take a more careful look at 2.2, we can notice that the combination of **q** and **p** is very similar to a VAE. Thus, we can train it by optimizing the **NLL** of the training data, which measures the likelihood of generating the observed data under the model's distribution, comprising the most common choice for training a diffusion model. Minimizing the NLL loss effectively aligns the model's distribution with the true data distribution. After a series of calculations, which we do not need to delve into here, the evidence lower bound (ELBO) can be expressed as follows:

$$\begin{gathered} logp(\mathbf{x}) >= \mathbb{E}_{q(x_1|x_0)}[logp_\partial(\mathbf{x}_0|\mathbf{x}_1)] - \mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)) - \\ \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}[\mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, x_0)||p_\partial(\mathbf{x}_{t-1}|\mathbf{x}_t)] = L_0 - L_T - \sum_{t=2}^{T} L_{t-1} \end{gathered} \tag{2.31}$$

where:

$$\mathcal{D}_{\mathrm{KL}}(P||Q) = \int_{-\infty}^{\infty} P(\mathbf{x}) \log \frac{P(\mathbf{x})}{Q(\mathbf{x})} \tag{2.32}$$

and

$$L_t = \mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}_t|\mathbf{x}_t, \mathbf{x}_0)||p_\partial(\mathbf{x}_t|\mathbf{x}_{t+1}) \tag{2.33}$$

Certainly, analyzing these terms would provide clarity and deeper understanding. Let's break down the components of the evidence lower bound (ELBO) to gain insights into their roles and contributions to the overall objective.

1. The $\mathbb{E}_{q(x_1|x_0)}[logp_\partial(\mathbf{x}_0|\mathbf{x}_1)]$ term can been as a reconstruction term, similar to the one in the ELBO of a VAE. In [1] this term is learned using a separate decoder.

2. $\mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))$ shows how close $\mathbf{x}_T$ is to the standard Gaussian. Note that the entire term has no trainable parameters so it's ignored during training.

3. The third term $\sum_{t=2}^{T} L_{t-1}$ formulates the difference between the desired denoising steps $p_\partial(\mathbf{x}_{t-1}|\mathbf{x}_t)$ and approximated steps $q(\mathbf{x}_{t-1}|\mathbf{x}_t, x_0)$. $q(\mathbf{x}_{t-1}|\mathbf{x}_t, x_0)$ is used for

the approximation instead of $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ because, as illustrated in [43], the quantity $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ only becomes tractable when conditioned on $\mathbf{x}_0$.

Through the analysis of the evidence lower bound (ELBO), it becomes apparent that maximizing the likelihood effectively translates into learning the denoising steps $\mathbf{L}_t$. This insight underscores the fundamental objective of training the diffusion model to accurately denoise the input data distribution.

Intuitively, a painter (our generative model) needs a reference image $\mathbf{x}_0$ to slowly draw, through the reverse diffusion process $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$, step-by-step a new sample. As long as $\mathbf{x}_0$ is provided as a reference, we are capable of sampling $\mathbf{x}_t$ at noise level corresponding to diffusion step t conditioned on $\mathbf{x}_0$. Thus, by combining equations 2.10 and 2.11, each timestep will now yield a mean $\hat{\mu}_t$ (our target) which only depends on $\mathbf{x}_t$. Therefore we can use a neural network to approximate $\epsilon_\partial(\mathbf{x}_t, t)$ to approximate ϵ and consequently the mean:

$$\hat{\mu}_\partial(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\partial(\mathbf{x}_t, t)) \tag{2.34}$$

Thus, the loss function (the denoising term in the ELBO) can be formulated as follows:

$$\begin{aligned}\mathbf{L}_t &= \mathbb{E}_{\mathbf{x}_0,t,\epsilon}[\frac{1}{2\cdot\|\Sigma_\partial(x_t,t)\|_2^2}\cdot\|\hat{\mu}_t - \mu_\partial(\mathbf{x}_t,t)\|_2^2] \\ &= \mathbb{E}_{\mathbf{x}_0,t,\epsilon}[\frac{\beta_t^2}{2\alpha_t\cdot(1-\bar{\alpha}_t)\|\Sigma_\partial\|_2^2}\cdot\|\epsilon_t - \epsilon_\partial(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\cdot\epsilon, t)\|^2]\end{aligned} \tag{2.35}$$

This insight effectively demonstrates that rather than predicting the mean of the distribution, the model will predict the noise ϵ at each timestep t.

The authors of [1] proceeded to a few simplifications to the actual loss term as they ignore the weighting term in 2.35. Surprisingly, the simplified version outperforms the full objective:

$$\mathbf{L}_t^{simple} = \mathbb{E}_{\mathbf{x}_0,t,\epsilon}\|\epsilon - \epsilon_\partial(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\cdot\epsilon, t)\|^2 \tag{2.36}$$

Additionally, they decided to keep the variance fixed and have the network learn only the mean, as also elaborated on in a precedent chapter. This approach was subsequently enhanced by [51], who opted to allow the network to learn the covariance matrix Σ as well, leading to improved results.

Chapter 3

Motion Representation in Joint Kinematics

3.1 Fundamentals

Motion representation techniques in joint kinematics aim to describe the movement of joints in the human body. The fields of application for motion representation techniques in joint kinematics are diverse and encompass various domains. These techniques are essential in various fields from healthcare and sports science to robotics and entertainment, driving advancements in research, technology, and human performance. To elaborate further, let's examine how joint kinematics is applied in a plethora of domains and explore its practical uses within each by providing a few characteristic examples across these domains, ending up with its application in animation, which constitutes the primary field of focus in the context of the current thesis.

1. **Biomechanics**:In biomechanics, motion representation techniques are used to analyze human movement patterns, joint kinematics, and muscle dynamics. These techniques help biomechanists understand the mechanics of human motion, study gait analysis, assess athletic performance, and design rehabilitation programs using methods such as Inverse Kinematics, Forward Kinematics[52] and Static Optimization [53]. Motion capture systems, such as *Vicon* or *Kinect*[54], combined with sophisticated motion representation methods, enable researchers to quantify and analyze the complex interactions between different body segments during movement.

2. **Robotics**: Motion representation techniques play a crucial role in robotics for modeling the motion of robot manipulators, robotic arms, and mobile robots. These techniques are used to describe the **pose (position and orientation)** of robot end-effectors, plan trajectories, and control robot motion. By accurately representing joint kinematics, robotics engineers can design robots that perform precise tasks in diverse environments, such as industrial automation, manufacturing, healthcare (e.g. surgical operations), and space exploration.

3. **Sports Science**: Motion representation techniques are applied in sports science to analyze athletic performance, biomechanics of high intensity sports movements, and injury prevention. Researchers use motion capture technology in collabotation with advanced motion representation methods to optimize training programs,

and improve athletic performance. By quantifying joint kinematics and movement patterns, sports scientists can provide valuable insights to coaches, trainers, and athletes, leading to more efficient training strategies and injury rehabilitation protocols.

4. **Rehabilitation Engineering**: Motion representation techniques are utilized in rehabilitation engineering to assess movement disorders, monitor rehabilitation progress, and develop assistive devices. These techniques help rehabilitation specialists analyze the kinematics and dynamics of impaired joints, track improvements over time, and customize rehabilitation interventions for individual patients with the contribution of simulation programs such as OpenSim[55]. By integrating motion capture systems with biomechanical models and motion representation methods, rehabilitation engineers can design personalized therapies and assistive technologies to aid individuals with mobility impairments.

5. **Computer Graphics and Animation**: Motion representation techniques are fundamental in computer graphics and animation for creating realistic simulations of movement. These methods are employed to animate characters, simulate physical interactions, and render lifelike animations in movies, video games and mixed reality applications. By employing sophisticated motion representation methods, animators can achieve smooth and natural-looking motion, express emotions through character animation, and simulate complex physical phenomena with high fidelity.

The motion generation techniques to be examined within this study are primarily associated, naturally arising, with the aforementioned final area of application. Therefore, we will proceed by showcasing motion representation methods predominantly utilized in this context.

3.2 Motion representation alternatives

3.2.1 3D Keypoints

In joint kinematics, **3D positions** (also known as 3D keypoints) refer to the spatial coordinates that describe the location of a joint or a point $p \in \mathcal{R}^3$ in three-dimensional space. These positions are typically represented using Cartesian coordinates $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ relative to a reference frame. While other coordinate systems like polar or spherical coordinates could theoretically be utilized, they are not commonly employed in the paradigm under examination. In this context, only the Cartesian coordinate system is utilized. The equations for calculating 3D positions depend on the chosen coordinate system. In Cartesian coordinates, the 3D position of a point is represented by its x, y, and z coordinates relative to a reference frame. 3D positions can be either *global* or *local*. **Global** positions are defined with respect to a fixed reference point or coordinate system, often the **origin of the coordinate system**. These positions are absolute and do not change regardless of the position or orientation of other joints in the skeleton. Global positions are typically used to represent the overall position of joints within a larger environment. On the other hand,

local positions, are defined relative to a specific parent joint or coordinate system that may move or change orientation. These positions are relative and can vary depending on the context or frame of reference. Local positions are often used to describe the position of components or features relative to a parent object, such as the position of a limb relative to the body in a character animation.

In summary, global 3D positions are absolute and defined with respect to a fixed reference point, while local 3D positions are relative and defined with respect to a specific object or coordinate system. In deep network architectures utilized for human motion generation, local positions are often prioritized over global position representation. The rationale behind this preference will be thoroughly examined in upcoming chapters.

3.2.2 Euler Angles Rotation

Euler angles describe the orientation of a rigid body in 3D space using three angles that represent rotations about distinct axes (typically X, Y, and Z axes). Euler angles are typically denoted as α, β and γ, where each angle represents a rotation about a different axis. The order of rotations can vary, leading to different Euler angle conventions, such as *XYZ*, *ZYX* etc. Euler angles are applied sequentially to rotate a coordinate system. For instance, in the XYZ convention, a rotation by angles α, β and γ about the x, y and z axes, respectively, can be represented as:

$$R_{xyz}(\alpha, \beta, \gamma) = R_z(\gamma) \cdot R_y(\beta) \cdot R_x(\alpha) \tag{3.1}$$

where

$$R_x(\partial) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\partial) & -\sin(\partial) \\ 0 & \sin(\partial) & \cos(\partial) \end{pmatrix} \tag{3.2}$$

The resulting rotation matrix represents the composite rotation about the specified axis x.

While intuitive, Euler angles suffer from gimbal lock(occurs when two of the rotation axes align.), where certain orientations can cause ambiguity or loss of one degree of freedom. These shortcomings of the Euler angles representation prompt the development of alternative rotation representations like quaternions and axis-angle representations for specific applications.

3.2.3 Rotation Matrices

Rotation matrices are fundamental in representing three-dimensional rotations. They describe how coordinates are transformed when an object is rotated about a specific axis. A rotation matrix is a square $3x3$ matrix that represents a rotation in three-dimensional space. It preserves distances and angles, meaning that points on the surface of an object, for instance a human skeleton in our own paradigm, remain the same distance apart and angles between lines are preserved after rotation.

Rotation matrices are orthogonal matrices, meaning that their inverse is equal to their transpose: $R^{-1} = R^T$. Multiplying two rotation matrices results in another rotation matrix, implying that rotations can be composed or concatenated. Rotations about the coordinate axes (x, y, z) can be represented by specific rotation matrices as the formula 3.2 dictates.

To rotate a point $\mathbf{p} = (x, y, z)^T$ in three-dimensional space using a rotation matrix R, you multiply the rotation matrix by the column vector representing the point: $\mathbf{p'} = R \cdot \mathbf{p}$.

Last but not least, there is a one-to-one correspondence between 3x3 rotation matrices and unit quaternions. Each rotation matrix can be uniquely represented by a unit quaternion, and vice versa, as will be analyzed in a forthcoming chapter.

Rotation matrices play a crucial role in computer graphics, robotics, physics, and engineering for describing and manipulating three-dimensional orientations and transformations. They provide a concise and computationally efficient representation of rotations, enabling precise control and analysis of spatial configurations and motion.

3.2.4 Axis-Angle Representation

The axis-angle representation is a method used to describe rotations in three-dimensional space by specifying an axis of rotation and the angle of rotation about that axis. It is intuitive and straightforward to visualize, making it popular for applications where interpretability is important.

To delve into further detail, in axis-angle representation, a rotation in three dimensions is characterized by a unit vector $\mathbf{u} = (u_x, u_y, u_z)$ representing the axis of rotation and an angle ∂ representing the magnitude of the rotation about that axis. Together, the vector $\mathbf{u}$ and the angle ∂ define the rotation uniquely. The axis-angle representation $\mathbf{a}$ can be expressed as a combination of the axis vector $\mathbf{u}$ and the angle of rotation ∂:

$$\mathbf{a} = \partial \mathbf{u} \tag{3.3}$$

Given an axis-angle representation $\mathbf{a}$, the corresponding rotation matrix R can be calculated using Rodrigues' rotation formula:

$$R = \cos(\partial)I + (1 - \cos(\partial))\mathbf{u}\mathbf{u}^T + \sin(\partial)[\mathbf{u}] \tag{3.4}$$

where I is the $3x3$ identity matrix, $[\mathbf{u}]$ is the skew-symmetric matrix associated with the axis vector $\mathbf{u}$. Conversely, given a rotation matrix R, you can extract the axis-angle representation using mathematical techniques such as the eigenvalue decomposition[56] or the axis-angle formula.

To illustrate certain beneficial properties, axis-angle representation is a minimal parameterization of rotations, as it requires only four numbers (three for the axis direction and one for the angle) to represent a rotation in three dimensions. It is globally non-redundant, meaning that each unique axis-angle representation $\mathbf{a}$ corresponds to a unique rotation matrix R. Additionally, axis-angle representation allows for straightforward interpolation between two rotations. Linear interpolation (lerp) or spherical linear interpolation (slerp) can be used to smoothly transition from one rotation to another by interpolating their axis-angle representations.

Axis-angle representation provides an intuitive and efficient way to represent rotations in three dimensions. It is widely used in robotics, computer graphics, biomechanics, and other fields where precise control and manipulation of orientations are required. Additionally, it offers advantages such as compactness, uniqueness, and ease of interpolation compared to other rotation representations like *Euler angles*.

3.2.5 Quaternions Representation

Quaternions are a mathematical construct used to represent rotations in three-dimensional space. They offer several advantages over other representations, such as Euler angles and rotation matrices. Quaternions comprise an extension of complex numbers, comprising a scalar part (real) and a vector part (imaginary).

A quaternion q is typically represented as $q = s + v$ where s is the scalar part (real) and v is the vector part (imaginary), often denoted as $v = xi + yj + zk$. In quaternion representation, a rotation is described by a unit quaternion:

$$q = cos(\frac{\partial}{2}) + sin(\frac{\partial}{2})\mathbf{u} \tag{3.5}$$

where ∂ is the angle of rotation and u is the unit axis vector. The unit axis vector $\mathbf{u}$ represents the axis of rotation.

When it comes down to useful mathematical properties, quaternion multiplication is non-commutative, meaning that $q_1 \cdot q_2 \neq q_2 \cdot q_1$, which secures that the defined rotations are unique. In other words, the issue of **gimbal lock** is no longer present in this form of representation. Moreover, quaternion multiplication is defined by the distributive property and the rules for multiplying imaginary units: $i^2 = j^2 = k^2 = i \cdot j \cdot k = -1$. Quaternion addition and subtraction are performed component-wise.

To rotate a point $\mathbf{p} = (x, y, z)$ using a quaternion q, the quaternion $\mathbf{p}'$ representing the rotated point is obtained by quaternion multiplication: $\mathbf{p}' = q \cdot \mathbf{p} \cdot q^*$ where q^* represents the conjugate of q, and quaternion multiplication is performed using the rules of quaternion algebra. Additionally, quaternions offer a convenient way to interpolate between two rotations using spherical linear interpolation (*slerp*). *Slerp* ensures constant angular velocity throughout the interpolation, making it suitable for smooth animations and camera movements.

The benefits of using quaternion representation can be succinctly summarized with the following points:

- Quaternions are compact and require only four parameters to represent a rotation, compared to the nine parameters of a 3x3 rotation matrix.

- They avoid the issues of gimbal lock associated with Euler angles and the numerical instability of rotation matrices.

- Quaternions provide a smooth and efficient way to interpolate between rotations, making them ideal for animations and robotics applications.

Quaternions are widely used in computer graphics, robotics, aerospace engineering, and physics for representing and manipulating three-dimensional rotations. Their properties, such as compactness, numerical stability, and ease of interpolation, make them a versatile and powerful tool for describing rotations in three-dimensional space.

3.2.6 6D Rotation Representation

The **6D rotation representation**, also known as the full 6D pose representation, describes the orientation and position of an object in three-dimensional space using six degrees of freedom (DOF). Unlike simpler representations like Euler angles or rotation matrices, the 6D representation captures both the rotational and translational components of the object's pose. The 6D rotation representation consists of three components for orientation (rotation) and three components for position (translation). The rotational component typically uses quaternion or rotation matrix representation to describe the orientation of the object in 3D space. The translational component represents the position of the object's origin in three-dimensional space using Cartesian coordinates (x, y, z). These translational coordinates can be either absolute (**global**) or relative to the parent joint in the kinematic tree.

In further detail, The orientation component can be represented by a quaternion q or a $3x3$ rotation matrix R, while the position component is represented by a translation vector $\mathbf{t} = (x, y, z)$. The 6D pose representation can be written as a tuple $(q, \mathbf{t})$ or as a homogeneous transformation matrix:

$$T = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.6}$$

Here, R is the rotation matrix representing the orientation, $\mathbf{t}$ is the translation vector representing the position, and $\mathbf{0}$ is a $1x3$ zero vector.

The 6D rotation representation is a compact and comprehensive way to describe the pose of an object in three-dimensional space as it captures both rotational and translational components, allowing for precise positioning and orientation of objects in a 3D environment. The use of quaternions or rotation matrices ensures numerical stability and avoids issues such as gimbal lock associated with simpler rotation representations. Interpolation between two 6D poses involves interpolating both the orientation and position components separately.

In terms of fields of application, the 6D rotation representation is widely used in computer vision, robotics, augmented reality, virtual reality and 3D graphics applications. In robotics, it is used to represent the pose of robotic manipulators, end-effectors, and mobile robots for navigation and manipulation tasks. In computer vision, it is used for object tracking, pose estimation, registration, and 3D reconstruction from images or point clouds. All in all, it is found extremely useful in these fields owing to its compactness, numerical stability and versatility, which make it an ideal form of representation for spatial manipulation, navigation and perception.

3.3 Motion Capture Datasets for Training

Motion capture datasets aimed for artificial human motion generation are crucial resources for training and evaluating machine learning models and algorithms in the field of computer graphics, animation, robotics, and related areas. These datasets typically contain recorded human motion data captured from sensors or motion capture systems, also known as **mocap** systems, along with associated metadata such as skeletal information, annotations, and environmental conditions. Motion capture systems (e.g. **Vicon**[54], **XSens**[57]) are specialized hardware setups designed to capture the movements of human actors or objects. These systems typically use markers placed on the body or clothing of the subject, which are tracked by multiple cameras positioned around the capture volume. Cameras can be optical (e.g., infrared cameras) or non-optical (e.g., electromagnetic sensors), and they capture the movement of markers in three-dimensional space.

There are several motion capture datasets available for artificial human motion generation, though the number of publicly available datasets may be limited compared to other domains, posing a strong challenge and, subsequently, an obstacle in order for the field of human motion generation and editing to flourish further. Thus, despite the availability of existing datasets in absolute terms, there may be challenges and limitations associated with them, such as:

- *Limited diversity*: Some datasets may have a limited variety of motions, which can affect the generalization ability of models trained on them.
- *Data quality*: Noise, occlusions, and artifacts in motion capture data can affect the quality and reliability of training data.
- *Scalability*: : Generating large-scale datasets with diverse and realistic human motions can be time-consuming and resource-intensive.
- *Privacy concerns*: Datasets containing human motion data may raise privacy concerns, particularly if they include identifiable individuals.

These challenges are addressed through various strategies, including data augmentation, transfer learning, synthetic data generation, collaborative efforts, and privacy-preserving techniques, to advance research in this field - it would be needless to further elaborate on this, though. On the contrary, it is crucial that we enumerate and furnish the requisite details regarding the datasets employed for training the models/networks corresponding to the architectures investigated in the subsequent chapters.

3.3.1 KIT dataset

KIT MotionLanguage Dataset[39] is to date the only available dataset comprising both 3*D* human motions and their annotated textual descriptions, which consists of 3,911 motion sequences and 6,278 sentences, and is focused on locomotion movements. Following the same philosophy with **KIT**, there are a number of existing datasets of 3D motion captured human motions, such as CMU Mocap[58], Human3.6M[59], MoVi[60] and BABEL[61], in the form of everyday actions and sports movements. However, none of them possesses language descriptions of the motions.

A data aggregation from multiple motion capture databases was carried out to include them all in a dataset using a unified representation that is independent of the capture system or marker set, making it easy to work with the data regardless of its origin. To obtain motion annotations in natural language, a crowd-sourcing approach was applied and a web-based tool specifically built for this purpose, the Motion Annotation Tool, was employed. The dataset contains 3911 motions with a total duration of 11.23 hours and 6278 annotations in natural language that contain 52903 words. the **KIT Whole-Body Human Motion Database**, which was captured using a sampling frequency of 100 Hz. For human subjects, a standardized marker set that consists of 56 markers is employed.

The dataset is structured as follows: Each entry comprises four files, including the raw motion data generated by the capture system, the motion data converted using a reference model, annotations written in natural language, and supplementary metadata. Each entry is linked to a set of annotations, establishing a one-to-many relationship. This data is presented in a straightforward *JSON-based* format: each file corresponding to an entry contains a simple array of strings, representing all associated annotations.

The KIT dataset employs the **Master Motor Map(MMM)** representation which proposes joint angle parameters by adopting a uniform skeleton structure with 50 DoFs as demonstrated in [20]. A preprocess procedure was executed so as to transform joint rotation angles to $J = 21$ joints *XYZ* coordinates, yielding $p_m \in \mathrm{R}^{3J}$ and global trajectory t_root for the root joint. The preprocessed representation can be formulated as:

$x^i = \{p_m, t_{root}\}$

3.3.2 HumanML3D dataset

The motivation behind creating the HumanML3D dataset, as described in [62], stemmed from the realization that existing endeavors in generating 3D human motions from descriptions were sporadic and fell short of satisfaction, primarily due to the reliance on a single dataset in existing human motion generation methods—the KIT-ML dataset. Unfortunately, this dataset was limited in size, prompting the need for a more comprehensive alternative. Consequently, this deficiency was rectified by the development of a dedicated dataset, HumanML3D, which comprises 44970 textual descriptions corresponding to 14616 3D human motions, thus yielding three textual descriptions coupled with each motion sequence on average. HumanML3D encompasses a diverse array of action types, including locomotive actions, among others.

The HumanML3D dataset originates from a amalgamation of motion sequences from the HumanAct12[63] and AMASS[64] datasets, two large-scale datasets of 3D human motion captures that are publicly accessible. They contains motions from a variety of human actions, such as daily activities (e.g., 'walking', 'jumping'), sports (e.g, 'swimming', 'karate'), acrobatics (e.g, 'cartwheel') and artistry (e.g, 'dancing'), but, unfortunately, these two datasets were short of textual descriptions.

Consequently, several processing steps were needed to end up with the final outcome. A textual annotation process via the Amazon Mechanical Turk (AMT), where native English-speaking turkers were hired and asked to describe a motion with at least 5 words. 3 textual descriptions were asked for each motion clip from distinct workers. A manual post-processing step was undertaken to filter out abnormal textual descriptions.

Additional post-processing was carried out for normalization purposes. Motions were scaled to 20 FPS, and those longer than 10 seconds were randomly cropped to 10-second ones - they were then retargeted to a default human skeletal template and properly rotated to **face Z+ direction** initially. Human skeletal templates will be comprehensively addressed in an upcoming section.

The HumanML3D dataset comprises the largest and most diverse collection of scripted human motions, consisting of 14616 motions and 44970 descriptions composed by 5371 distinct words. The total length of motions amounts to 28.59 hours, in which the average motion length is 7.1 seconds. The minimum and maximum duration are $2s$ and $10s$ respectively. In terms of the textual descriptions, their average and median lengths are 12 and 10 words, respectively.

The *HumanML3D Representation Format* proposes a motion representation $x^{1:L}$ inspired by motion features in character control suggested in [65]. This redundant representation is quite suited to neural models, particularly VAEs. Specifically, the $i - th$ pose x^i is defined by a tuple of root angular velocity $\dot{r}^a \in \mathrm{R}$ along Y-axis, root linear velocities ($\dot{r}^x, \dot{r}^z \in \mathrm{R}$) on XZ-plane, root height $r^y \in \mathrm{R}$, local joints positions $\mathbf{j}^p \in \mathrm{R}^{3N}$, velocities $\mathbf{j}^v \in \mathrm{R}^{3N}$ and rotations $\mathbf{j}^r \in \mathrm{R}^{3N}$ in root space and binary foot-ground contact features $\mathbf{c}^f \in \mathrm{R}^4$ by thresholding the heel and toe joint velocities, where N denotes the joints number, yielding:

$x^i = \{\dot{r}^a, \dot{r}^x, \dot{r}^z, r^y, \mathbf{j}^p, \mathbf{j}^v, \mathbf{j}^r, \mathbf{c}^f\}$.

3.4 Skeletal Templates in Motion Generation

In the context of human motion generation, skeletons play a crucial role as they provide a structured representation of the human body's underlying anatomy and kinematic structure. A skeleton is a hierarchical structure composed of interconnected bones or joints representing the articulated structure of the human body.

In computer graphics, animation, and biomechanics, skeletons are often represented as hierarchical trees, where each bone or joint is a node in the tree, and the connections between them represent the articulations between body parts. Skeletons are typically represented using *joint-based* models or *bone-based* models. In joint-based models, each joint represents a point of articulation between bones, and rotations at these joints determine the pose of the skeleton. In bone-based models, bones are represented as rigid segments, connecting adjacent joints, and transformations (e.g., rotations and translations) are applied to these bones to pose the skeleton. The motion generation models employed in this study are exclusively oriented around joints. As a result, it is imperative for us to utilize **joint-based** skeletal templates for our objectives.

Skeletons serve as the underlying structure for generating realistic and natural-looking human motions. Motion generation algorithms often manipulate the pose of the skeleton by applying rotations and translations to its joints or bones. By animating the skeleton, motions can be synthesized for various applications, including computer animation, virtual reality, gaming, biomechanics research, and robotics.

Motion capture data, which records the movements of real humans or creatures, is often applied to skeletons using a process called **retargeting**. Retargeting involves mapping the motion capture data onto a target skeleton with a different structure or proportions, enabling the reuse of captured motions for different characters or creatures. Skeletons provide a common representation that facilitates retargeting across different characters or creatures. This circumstance prompted the scientific community to establish baseline skeletal templates. These templates enable the retargeting of 3D motion capture data, recorded by diverse devices or sophisticated mocap systems, to a standardized skeleton acting as a reference. The most common out of these templates is, clearly, the SMPL skeleton and, secondarily, the KIT dataset skeleton.

3.4.1 The SMPL skeleton

The **SMPL skeleton** stands out as the most commonly employed in $3D$ motion capture datasets published over the last few years. Its topology is depicted in figure 3.1. The SMPL skeleton was initially introduced in [5] and has been extensively used because it comprises a simplified skeleton topology that captures the main anatomical features and DOFs of the human body while maintaining simplicity and efficiency for computational purposes.

The skeleton topology in SMPL follows a hierarchical structure, with each joint representing a point of articulation between bones. The joints are organized in a hierarchical manner, typically starting from a root joint (e.g., pelvis) and branching out to other body parts such as the spine, limbs, and head. The hierarchy reflects the natural anatomical structure of the human body, with parent-child relationships between joints representing the kinematic chain of the body.

The SMPL skeleton is consisted of **24 joints** as can be seen in 3.1. Depending on the motion data representation, each joint can be represented via $3D$ keypoints or some kind of rotation representation (e.g. quaternions, rotation matrices) or a combination of them.

The SMPL skeleton was qualified over other skeleton topologies to operate as a baseline topology owing to the fact it avoids excessive complexity that could lead to computational overhead or numerical instability. Indeed, the straightforwardness of the skeleton topology enhances the scalability and usability of the SMPL model for various applications, including artificial animation generation and subsequent animation pipelining in computer graphics and game engines, such as **Unreal Engine**. These aspects will be further elucidated in upcoming chapters. As expected, the widespread consideration of the SMPL skeleton as the universal standard lead to ensuring consistency and interoperability across different implementations and applications. Standardization facilitates the exchange of data and models between researchers and practitioners, enabling collaboration and reproducibility in the field of human body modeling and animation.

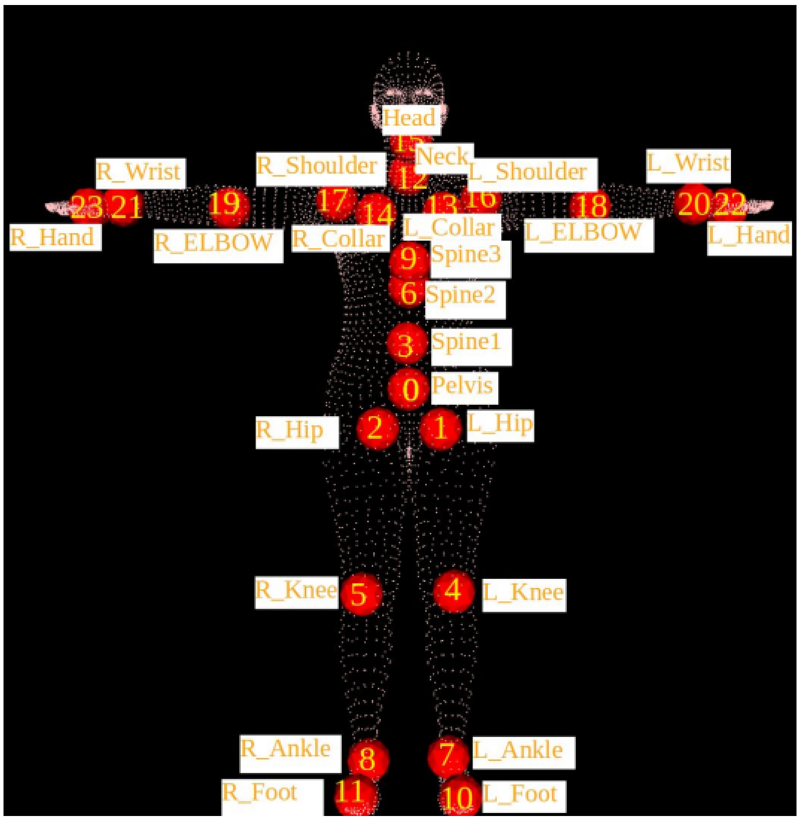


Figure 3.1. *SMPL skeleton topology. Source: [4]*

3.5 The SMPL model

3.5.1 Overview

The SMPL model is a widely used parametric model for representing human body **shape** and **pose** in computer graphics, computer vision and related fields. It was first introduced in [5] and has played a pivotal role in the context of human motion representation in diverse settings as it fulfils a two-fold purpose: faithfully representing the motion performed by a human skeleton while effectively depicting the shape of the subject's body, thereby facilitating the customization of human motion.

In further detail, the paper addresses the need for a compact and expressive model that can accurately represent human body shape and pose variations across different individuals. It introduces the SMPL model as a solution to this problem, which captures both body shape and pose using a *low-dimensional parameterization*. In short, it could be claimed that the SMPL model represents the human body as a combination of a linear shape space and a pose-dependent deformable mesh. It defines a **linear blend skinning (LBS)** function that maps a set of pose-dependent joint rotations to the vertices of a mesh. The model consists of a mean body shape and a set of shape parameters that capture deviations from the mean shape. Pose is represented by a set of joint rotations parameterized using rotation matrices (3.2.3) or quaternion (3.2.5) representations.

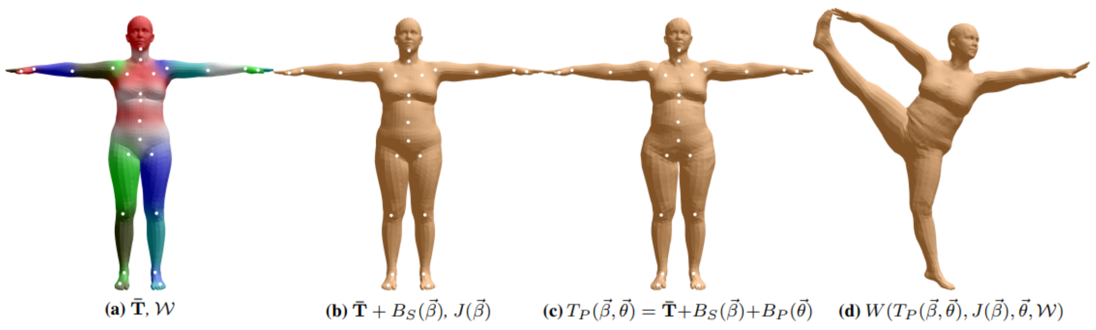

(a) $\bar{\mathbf{T}}, \mathcal{W}$
(b) $\bar{\mathbf{T}} + B_S(\vec{\beta}), J(\vec{\beta})$
(c) $T_P(\vec{\beta},\vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta})$
(d) $W(T_P(\vec{\beta},\vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W})$

Figure 3.2. *The SMPL model components. Source: [5]*

3.5.2 Parameterization

Regarding the parameterization, body shape is parameterized using a low-dimensional linear subspace defined by **principal component analysis (PCA)** on a training dataset of body scans. The resulting principal components comprise body shape blend shapes. The parameters of the model are learned from data including the rest pose template, blend weights, pose-dependent blend shapes, identity-dependent blend shapes and a regressor from vertices to joint locations.

Pose is parameterized by joint rotations, with each joint represented by three degrees of freedom (DOFs) for rotation in 3D space. The pose-dependent deformations are modeled using a set of pose-dependent corrective blend shapes. Unlike similar prior models, such as [66], the pose-dependent blend shapes are a linear function of the elements of the pose rotation matrices.

The SMPL model operational philosophy is efficiently depicted in figure 3.2. The SMPL model decomposes body shape into two components: identity-dependent shape and non-rigid pose-dependent shape. The SMPL model employs a vertex-based skinning approach that incorporates corrective blend shapes. Each blend shape is represented as a vector of concatenated vertex offsets. The process commences with an artist-created template mesh $\bar{\mathbf{T}} \in \mathrm{R}^{3N}$ where N = 6890(vertices) and is consisted of 23 joints, leaving the root joint out of consideration. The mesh has the same topology for men and women. Overall, the SMPL is defined by:

1. a mean template mesh $\bar{\mathbf{T}}$ in the zero pose $\vec{\partial}^*$
2. a set of blend weights $\mathrm{W} \in \mathrm{R}^{NxK}$
3. a blend shape function $\mathrm{B}_S(\vec{\beta}) : \mathrm{R}^{|\vec{\beta}|} \rightarrow \mathrm{R}^{3N}$ that takes as input a vector of shape parameters $\vec{\beta}$ and outputs a blend shape sculpting the subject identity
4. a function $\mathcal{J}(\vec{\beta}) : \mathrm{R}^{|\vec{\beta}|} \rightarrow \mathrm{R}^{3K}$ to predict K joint locations as a function of shape parameters $\vec{\beta}$
5. a pose-dependent blend shape function $\mathrm{B}_P(\vec{\partial}) : \mathrm{R}^{|\vec{\partial}|} \rightarrow \mathrm{R}^{3N}$ that takes as input a vector of pose parameters, $\vec{\beta}$, and accommodates the effect of the pose-dependent deformations

The corrective blend shapes of these functions are added to the rest pose as illustrated in 3.2(c). Finally, a standard blend skinning function $W()$ (most prevalently linear) is applied to rotate the vertices around the estimated joint centers with smoothing defined by the blend weights. The result is a model $\mathrm{M}(\vec{\beta}, \vec{\partial}; \Phi) : \mathrm{R}^{|\vec{\beta}| x |\vec{\partial}|} \to \mathrm{R}^{3N}$ that maps shape and pose parameters to vertices as shown in 3.2(d). Φ represents the learnt model parameters. Given a particular skinning method (e.g. LBS) our goal is to learn Φ to correct for limitations of the method so as to model training meshes. Using the standard linear blend skinning function $\mathrm{W}(\bar{\mathbf{T}}, \mathbf{J}, \vec{\partial}, \mathcal{W}) : \mathrm{R}^{|\vec{\beta}| x |\vec{\partial}| x 3N x 3K} \to \mathrm{R}^{3N}$, which takes vertices in the rest pose $\bar{\mathbf{T}}$, joint locations $\mathbf{J}$, a pose $\vec{\partial}$ and the blend weights $\mathcal{W}$, we obtain the posed vertices.

Each element of $w_{i,k}$ the learned blend weight matrix $\mathcal{W}$ represents how much the rotation of part k effects the vertex i. Note that for securing compatibility with existing rendering engines, $\mathcal{W}$ is assumed to be sparse, thus allowing at most four parts to influence a vertex.

Our model $M(\vec{\partial}, \vec{\beta}; \Phi)$ can be explicitly defined through the following two equations:

$$M(\vec{\partial}, \vec{\beta}) = W(T_p(\vec{\partial}, \vec{\beta}), J(\vec{\beta}), \vec{\partial}, \mathcal{W}) \tag{3.7}$$

$$T_p(\vec{\partial}, \vec{\beta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\partial}) \tag{3.8}$$

where $B_S(\vec{\beta})$ and $B_P(\vec{\partial})$ are vectors of vertices representing offsets from the template and are also known as shape and pose blend shapes, respectively. Therefore, with a bit of creative thinking, one can discern that the joint centers are influenced by both body shape and the template mesh, which undergoes deformation through blend skinning, accounting for both pose and shape variations.

Up to this juncture, it's crucial to delve deeper into two pivotal components of the SMPL model: the shape blend shapes $B_S(\vec{\beta})$ and pose blend shapes $B_P(\vec{\partial})$. The body shapes of different people are represented by a linear function B_S such that:

$$B_S(\vec{\beta}; \mathcal{S}) = \sum_{i=1}^{|\vec{\beta}|} \beta_n \mathbf{S}_n \tag{3.9}$$

where $|\vec{\beta}|$ is the number of linear shape coefficients, and the $\mathbf{S}_n \in \mathrm{R}^3$ represent orthonormal principal components of shape displacements. Notationally, the values to the right of a semicolon represent learnt parameters, while those on the left are parameters set by an animator, yielding the immediate conclusion that $B_S(\vec{\beta}; \mathcal{S})$ is dependent exclusively upon matrix $\mathcal{S}$, which is learnt from registered training meshes, where $\mathcal{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_{|\vec{\beta}|}] \in \mathrm{R}^{3N x |\vec{\beta}|}$. Figure 3.2(b) illustrates the application of these shape blend shapes to the template $\bar{\mathbf{T}}$ to produce a new body shape.

Turning to pose blend shapes, we define function $\mathcal{R} : \mathrm{R}^{|\vec{\partial}|} \to \mathrm{R}^{9K}$ which maps a pose vector $\vec{\partial}$ to a vector of concatenated part relative rotation matrices. Given that the SMPL rig is consisted of 23 joints, $\mathcal{R}$ is a vector of $23x9 = 207$ elements. $\mathcal{R}$ is non-linear with respect to ∂ as they are related through sine and cosine functions. The effect of the pose blend shapes, on the other hand, is supposed to be linear in $\mathcal{R}^*(\vec{\partial}) = \mathcal{R}(\vec{\partial}) - \mathcal{R}(\vec{\partial}^*)$ where $\vec{\partial}^*$ denotes the rest pose. Let $\mathcal{R}_n(\vec{\partial})$ be the *n-th* element of $\mathcal{R}(\vec{\partial})$, then the vertex deviations from the rest template are expressed as:

$$B_P(\vec{\partial};\mathcal{P}) = \sum_{i=1}^{9K}(\mathcal{R}_n(\vec{\partial}) - \mathcal{R}_n(\vec{\partial}^*))\mathbf{P}_n \tag{3.10}$$

where the blend shapes, $\mathbf{P}_n \in \mathrm{R}^{3N}$, are vectors representing the vertex displacements. Here, $\mathcal{P} = [\mathbf{P}_1, ..., \mathbf{P}_{9K}] \in \mathrm{R}^{3Nx9K}$ is a matrix of all 207 pose blend shapes. In this way, the pose blend shape function $B_P(\vec{\partial};\mathcal{P})$ is explicitly defined by the matrix $\mathcal{P}$.

To proceed to **joint positions**, we have to enumerate some fundamental principals. Firstly, various body shapes entail different joint locations, with each joint represented by its $3D$ location in the rest pose. Ensuring the accuracy of these joint locations is paramount; otherwise, artifacts may arise when posing the model using the skinning equation. Hence, we define the joints as a function of the body shape $\vec{\beta}$, to maintain fidelity and mitigate potential artifacts, as follows:

$$J(\vec{\beta};\mathcal{J},\bar{\mathbf{T}},\mathcal{S}) = \mathcal{J}(\bar{\mathbf{T}} + B_S(\vec{\beta};\mathcal{S})) \tag{3.11}$$

where $\mathcal{J}$ is a matrix that transforms rest vertices into rest joints. We learn the regression matrix, $\mathcal{J}$, from examples of different people in many poses. This matrix models which mesh vertices are important and how to combine them to estimate the joint locations.

In summarizing the parameterization process, one may assert that the complete set of model parameters for the SMPL model can be defined as $\Phi = \{\bar{\mathbf{T}}, \mathcal{W}, \mathcal{S}, \mathcal{J}, \mathcal{P}\}$. Once this set of parameters are learnt and kept fixed, new body shapes and poses are created and animated by varying $\vec{\beta}$ and $\vec{\partial}$ respectively.

3.5.3 Model Optimization

Given an input image or point cloud of a human subject, the SMPL model is fitted to the observed data by optimizing the shape and pose parameters to minimize the difference between the model and the observations. Model fitting can be performed using optimization techniques such as gradient descent.

Delving into further detail, the goal is to train the parameters $\Phi = \{\bar{\mathbf{T}}, \mathcal{W}, \mathcal{S}, \mathcal{J}, \mathcal{P}\}$ to minimize vertex reconstruction error on two datasets, namely the *multi-pose* and the *multi-shape* datasets. Each dataset contains meshes with the same topology as our template mesh $\bar{\mathbf{T}}$ that have been aligned to high-resolution 3D scans - these aligned meshes “registrations.” Since the SMPL model segregates shape and pose, they are trained independently,thus streamlining the optimization process. $\{\mathcal{J}, \mathcal{W}, \mathcal{P}\}$ were trained using the multi-pose dataset while $\{\bar{\mathbf{T}}, \mathcal{S}\}$ were trained with the shape dataset. Separate models are trained for men and women. Our examination will be confined to a high-level overview of the optimization process, as providing further detail on this matter falls beyond the scope of the present work.

Pose parameter training

On the first end, namely the pose parameters training, an objective function consisting of a data term E_D is minimized along with a set of regularization terms $\{E_J, E_Y, E_P, E_W\}$. The data term penalizes the squared Euclidean distance **between registration vertices and model vertices**. The symmetry regularization term E_Y penalizes left-right asymmetry encourages symmetric template meshes and,more importantly, symmetric joint locations, as, according to literature, enforcing symmetry produces models that are visually more intuitive for animation purposes. Our model is hand-segmented into 24 parts. This segmentation is used to compute an initial estimate of the joint centers and a regressor $\mathcal{J}_I$ from vertices to these centers. This regressor computes the initial joints by taking the average of the ring of vertices connecting two parts. When estimating the joints for each subject, they are regularized to be close to this initial prediction through the regularization term E_J. To help prevent overfitting of the pose-dependent blend shapes, they are regularized towards zero through the regularization term E_J. In similar fashion, the blend weights are regularized towards the initial weights $\mathcal{W}_I$, which are computed by simply diffusing the segmentation. Hence, the overall training loss for $\{\mathcal{W}, \mathcal{P}\}$ is formulated as:

$$E_* = E_D + \lambda_Y E_Y + \lambda_J E_J + \lambda_P E_P + E_W \tag{3.12}$$

where $\lambda_Y = 100$, $\lambda_J = 100$ and $\lambda_P = 25$ were set empirically. Optimizing the above yields a template mesh and joint locations for each subject, but the aim is to predict joint locations for new subjects with new body shapes. To that end, the regressor matrix $\mathcal{J}$ is learnt to predict the training joints from the training bodies. This approach encourages sparsity of the vertices used to predict the joints. Making weights positive and add to one discourages predicting joints outside the surface. These constraints enforce the predictions to be in the convex hull of surface points.

Shape parameter training

The SMPL shape space is defined by a mean and principal shape directions $\{\bar{\mathbf{T}}, \mathcal{S}\}$ It is computed by running PCA on shape registrations from the aforementioned multi-shape database after **pose normalization**. This normalization is critical to ensure that pose and shape are modeled separately. Pose normalization transforms a raw registration $\mathbf{V}_j^S$ into a registration $\hat{\mathbf{T}}_j^S$, in the rest pose $\vec{\theta}^*$. This normalization is critical to ensure that pose and shape are modeled separately but we will not refrain from delving further into the pose normalization process to avoid excessive elaboration. Once the pose $\vec{\theta}_j^*$ is known, $\hat{\mathbf{T}}_j^S$ is obtained through minimizing:

$$\hat{\mathbf{T}}_j^S = \underset{\hat{\mathbf{T}}}{\operatorname{argmin}} \, \| W(\hat{\mathbf{T}} + B_P(\vec{\theta}_j; \mathcal{P}), \mathcal{J}\hat{\mathbf{T}}, \vec{\theta}_j, \mathcal{W}) - \mathbf{V}_j^S \|^2 \tag{3.13}$$

This process computes the shape that, when posed, aligns with the training registration. This resultant shape is termed the pose-normalized shape. PCA is then run on $\{\hat{\mathbf{T}}_j^S\}_{j=1}^{S_{subj}}$ to obtain $\{\bar{\mathbf{T}}, \mathcal{S}\}$. This procedure is devised to maximize the explained variance of

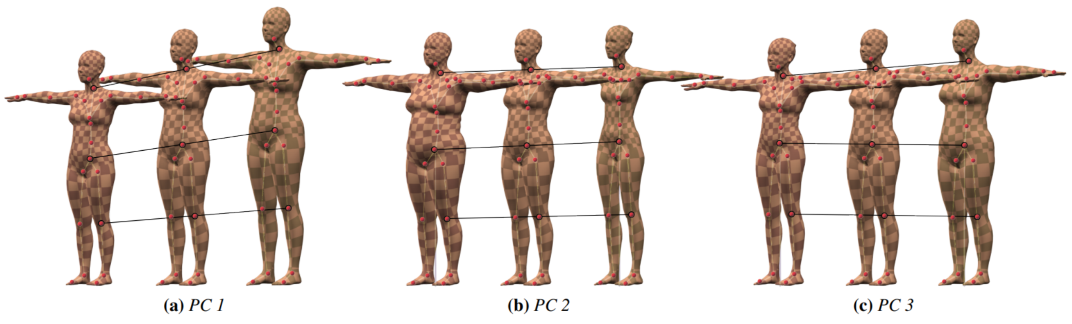


Figure 3.3. ***Shape Blend Shapes***: *Joint locations (red dots) vary as a function of body shape and are predicted using the learned regressor $\mathcal{J}$. Source: [5]*

vertex offsets in the rest pose, considering a constrained number of shape directions. It is crucial to emphasize that optimizing pose is of paramount importance when constructing a shape basis from vertices. Without this step, pose variations of the subjects in the shape training dataset would be embedded in the shape blend shapes. Consequently, the resulting model would not be accurately decomposed into shape and pose components. Moreover, it is worth noting that this approach differs from methods like SCAPE[67] or BlendSCAPE[68], where PCA is conducted in the space of per-triangle deformations. Unlike vertices, triangle deformations do not exist in a Euclidean space. Therefore, PCA on vertices is more principled and aligns with the registration data term, which comprises squared vertex disparities.

Figure 3.3 illustrates the first three shape components, demonstrating how the joint locations change with variations in body shape. The spheres represent the joint positions, computed from the surface meshes using the learned joint regression function. Additionally, the lines connecting the joints across the standard deviations depict how the joint positions linearly vary with shape.

Overall Optimization

Owing to the complexity of the SMPL optimization process, it is deemed essential to offer a concise overview of the entire procedure.

Pose parameters $\vec{\partial}_j$ are initialized as per in eq.3.12 by minimizing the difference between the model and the registration edges, using an average template mesh obtained by 3.13. $\{\hat{\mathbf{T}}^P, \hat{\mathbf{J}}^P, \mathcal{W}, \mathcal{P}, \Theta\}$ are estimated in an alternating manner to minimize eq.3.12. Subsequently, $\mathcal{J}$ is estimated from $\{\hat{\mathbf{T}}^P, \hat{\mathbf{J}}^P\}$ and PCA is run on pose normalized subjects $\{\hat{\mathbf{T}}_j^S\}_{j=1}^{S_{subj}}$ to obtain $\{\bar{\mathbf{T}}, \mathcal{S}\}$. The final model is is defined by $\{\mathcal{J}, \mathcal{W}, \mathcal{P}, \bar{\mathbf{T}}, \mathcal{S}\}$. Gradients are computed with automatic differentiation using the the Chumpy framework introduced in [69].

3.5.4 Impact

SMPL models can be animated significantly faster than real time on a CPU using standard rendering engines. Consequently SMPL addresses an open problem in the field; it makes a realistic learned model accessible to animators, allowing them to realistically animate human bodies, which is exactly why we devote so much time and space to it in the present work.

Additionally, since SMPL relies on standard skinning techniques, it seamlessly integrates with existing 3D animation software. Specifically, for a given body shape, we generate the subject-specific rest-pose template mesh and skeleton (including estimated joint locations). We export SMPL as a rigged model with pose blend shapes in Autodesk's Filmbox (FBX) file format, ensuring cross-platform compatibility. The model loads as a typical rigged mesh and can be animated as usual in standard 3D animation software.

Pose blend weights can be precomputed, baked into the model, and exported as an animated FBX file. Such files can be directly loaded into animation packages and played. We have tested the animated FBX files in Maya, Unity, Blender and, lately, in Unreal Engine. Pose blend weights can also be computed on the fly given the pose given a pose $\vec{\partial}_t$ at time t.

Naturally, the SMPL model also entails certain limitations that need to be acknowledged. Firstly, SMPL exclusively accounts for joint angles and shape parameters, omitting factors such as breathing, facial motion, muscle tension, or any changes independent of skeletal joint angles and overall shape. However, these factors could potentially be incorporated as additional additive blend shapes. Secondly, although most model parameters are learned, not all are. The segmentation of the template into parts, the topology of the mesh, and the zero pose are manually defined. While theoretically these aspects could also be learned, the expected improvements would likely be marginal relative to the significant effort required.

Overall, SMPL aims to create a skeletally-driven human body model capable of capturing body shape and pose variation as effectively as, or even better than, previous models. It achieves this while maintaining compatibility with existing graphics pipelines and software. To achieve these goals, SMPL utilizes standard skinning equations and defines body shape and pose blend shapes that modify the base mesh. The model is trained on thousands of aligned scans of different people in various poses.

3.6 The CLIP model

CLIP (Contrastive Language-Image Pretraining) is a neural network model developed by OpenAI and was first introduced in [6]. It is designed to understand images and text jointly in a single model. One of the remarkable features of CLIP is its ability to generalize across a wide range of tasks without task-specific training. This means that CLIP can perform tasks such as image classification, object detection, and image-text retrieval without needing additional fine-tuning on specific datasets for each task, demonstrating remarkable versatility in a plethora of different tasks such as image and text search, content moderation, image generation conditioned on text prompts, and more. Unlike traditional image recognition models that are trained solely on images, or natural language processing models that focus only on text, CLIP can comprehend both modalities simultaneously. However, in a CLIP model, the only interaction between the image and text domains is a single dot product in a learned joint embedding space. This pivotal property of CLIP is what renders it a valuable asset in the context of human motion generation.

3.6.1 Architecture

CLIP is based on a transformer[70] architecture, similar to models like GPT (Generative Pre-trained Transformer)[71] and BERT[72] (Bidirectional Encoder Representations from Transformers). This architecture allows CLIP to effectively process and represent both images and text. In the human motion paradigm, CLIP proves useful with text encoding, which involves transforming text tokens into numerical feature vectors that serve as a factor of supervision.

The transformer architecture, which relies heavily on self-attention mechanisms. Self-attention allows the model to weigh the importance of different words or image regions when processing each input token. This mechanism enables CLIP to capture long-range dependencies and relationships between elements in both images and text. This is the key characteristic of CLIP that can be critical in the domain of human motion generation - it is hugely important that long textual descriptions which significantly differ verbally produce similar text embeddings in terms of cosine distance, so that the generated motions are similar as should be.

The text encoder in CLIP is based on a Transformer architecture, as described in [70], with modifications outlined in [73]. As a base size, a 63M-parameter model with 12 layers, each 512 units wide and comprising 8 attention heads, is utilized. The Transformer operates on a lower-cased **BPE** representation of the text, with a vocabulary size of 49,152 as explained in [74]. To ensure computational efficiency, the maximum sequence length is capped at 76 tokens. The text sequence is enclosed within [SOS] (start of sequence) and [EOS] (end of sequence) tokens, and the activations of the highest layer of the Transformer at the [EOS] token are considered as the feature representation of the text. These activations are layer-normalized and then linearly projected into the multi-modal embedding space. Masked self-attention is employed in the text encoder to preserve the capability to initialize with a pre-trained language model or incorporate language modeling as an auxiliary objective.

3.6.2 Training

CLIP is trained using a **contrastive learning** approach. Contrastive learning is a training paradigm where the model learns by contrasting similar pairs of inputs with dissimilar pairs. In the case of CLIP, during training, the model is presented with pairs of images and text descriptions. The objective is to maximize the similarity between embeddings of matching image-text pairs while minimizing the similarity between embeddings of mismatched pairs. This training strategy enables CLIP to understand the semantic relationships between images and text.

More specifically, given a batch of *N* (image, text) pairs, CLIP is trained to predict which of the *NxN* possible (image, text) pairings across a batch actually occurred. To do this, CLIP learns a with high pointwise mutual information as well as the names of all Wikipedia articles above a certain search volume. Finally all WordNet synsets not

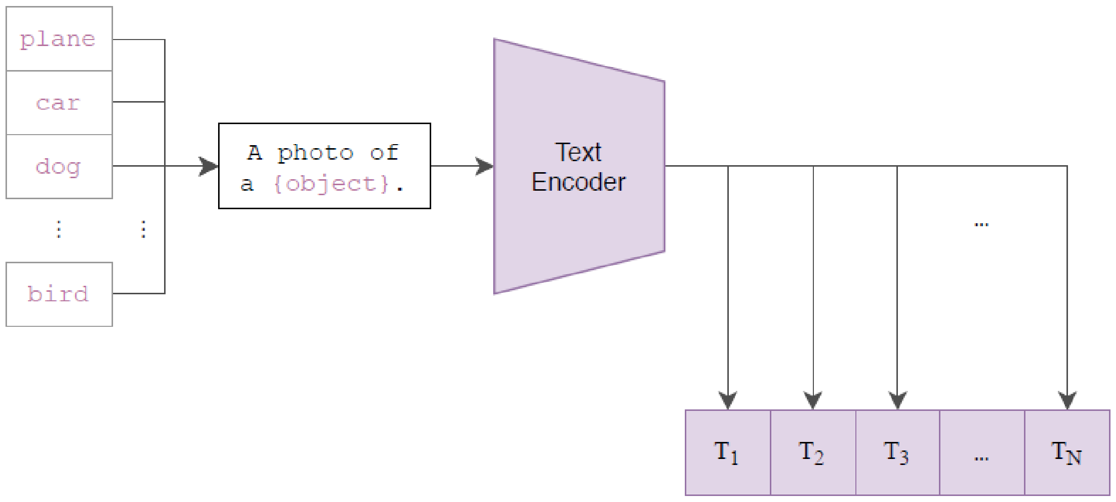


Figure 3.4. *Text encoder. Source: CLIP paper [6]*

already in the query list are added. multi-modal embedding space by jointly training an image encoder and text encoder to maximize the cosine similarity of the image and text embeddings of the N real pairs in the batch while minimizing the cosine similarity of the embeddings of the $N^2 - N$ incorrect pairings. A symmetric cross entropy loss over these similarity scores. This loss function practically quantifies the agreement between embeddings of image-text pairs. It penalizes the model when embeddings of matching pairs are far apart and when embeddings of mismatched pairs are too close. To conclude the training specifics, CLIP employs techniques such as batch normalization and scaling of embeddings to ensure stability and convergence during training,. These techniques help prevent issues like vanishing or exploding gradients and ensure that the model can effectively learn meaningful representations of both images and text.

3.6.3 Impact

CLIP is an example of using natural language as a training signal for learning about a domain other than language, As seen in the diffusion-based architectures to be presented and analyzed in the following chapters, the text encoder derived from CLIP (see figure 3.4) plays a pivotal role in efficiently channeling the influence of text supervision into our backbone models. Overall, the mathematical foundations of CLIP involve a combination of transformer-based architectures, contrastive learning principles, and techniques for embedding normalization and scaling, all working together to enable the model to understand and represent both images and text in a unified manner, entailing a significant advancement in multimodal AI.

Chapter 4

Diffusion-based Architectures in Human Motion Synthesis

In recent years, the field of computer graphics and animation has witnessed remarkable advancements in the generation and manipulation of human motion. With the growing demand for realistic character animation in various domains such as virtual reality, gaming, film production, and human-computer interaction, researchers have devoted substantial efforts to develop sophisticated techniques that can accurately model and animate human movements.

Among other suggested approaches, diffusion-based techniques have emerged as probably the most promising tools in the context of human motion generation and editing. Leveraging concepts from statistical physics, diffusion models are able to produce high-quality samples and can benefit from stable training compared to e.g. *GANs*. However, it relies on a long Markov chain of reverse diffusion steps to generate samples, so it can be computationally expensive and slower than *GANs or VAEs* - this shortcoming can be bypassed by using DDIM scheduling as analyzed in section 2.3.

This chapter provides a comprehensive overview of the state-of-the-art diffusion-based techniques for human motion generation and editing which are being comparatively evaluated in the following chapter. We delve into the key components of diffusion-based in terms of modelling, input/output motion representation, training datasets, computational complexity, etc. In this way, the latest advancements in the field of diffusion-based motion synthesis algorithms will be encompassed demonstrating how these methods leverage large-scale motion databases to learn spatio-temporal patterns and generate plausible human motions in diverse scenarios.

The application of diffusion-based techniques investigated in this chapter are additionally concerned with motion editing tasks, such as motion *in-betweening* (temporal editing), *spatial or joint* editing and motion *root trajectory control.* The ultimate goal of fulfilling the seamless manipulation of motion sequences is destined to enable users to modify, blend, and refine animations, securing motion realism and physical plausibility at the same time.

Throughout this chapter, we highlight exemplary works that have showcased the greatest efficacy and versatility among diffusion-based techniques in the paradigms of human motion synthesis and editing. By elucidating the underlying principles and methodologies, we aim to provide readers with a comprehensive understanding of the current state-of-the-art methods, laying the groundwork for the comparative evaluation displayed in the following chapter.

4.1 Motion Diffusion Model (MDM)

In the generation paradigm, such as motion generation, a diffusion model endeavors to learn how to produce additional samples of a given concept, such as motions, with a degree of diversity compared to the existing dataset. Conceptually, a diffusion model aims to grasp the essence of a target concept within the generation framework. To achieve this objective, it systematically introduces noise into input samples, such as motion samples, until the sample becomes fully "diffused," indicating that the noise is uniformly distributed across the sample, including joint rotations and positions. Subsequently, the neural network learns to separate the noise from the "true signal" and generate a novel motion. Once the neural network is trained under this framework, the trained diffusion model essentially samples noise from a normal (Gaussian) distribution and generates a new motion by eliminating the noise. The process of adding noise is iterated for a predefined number of steps, typically set to 1000 in MDM (Motion Diffusion Models). The magnitude of noise added at each step increases exponentially, while the percentage of the original image diminishes exponentially over the course of the noising process, namely while the number of diffusion steps is being increased.

4.1.1 Motion Representation & Operational Modes

The MDM's goal, no matter what the employed mode is, is to generate a sequence of poses $x_t^{1:N}$ given an arbitrary condition c, where N is the number of frames of the generated motion and t corresponds to the step of the denoising process (from 1 to 1000). Unconditioned motion generation is also possible, which we denote as the null condition $c = \emptyset$. The generated motion $x^{1:N} = \{x^i\}_{i=1}^N$ is a sequence of human poses represented by either joint rotations or positions $x^i \in \mathrm{R}^{JxD}$ where J is the number of joints and D is the dimension of the joint representation. In MDM, D is a **feature vector of size** 263 for the *HumanML3D dataset* and of size 251 for the *KIT dataset* for each frame. In further detail, the feature vector is composed of:

- root height (y-dimension) (1)
- root angular velocity along y-axis (1)
- root linear velocities on xz-plane (2)
- joint positions (3x21) - (3X20 for KIT)
- joint rotations in 6D continuous rotation representation (6x21) - (6x20 for KIT)
- joint velocities (3x22) - (3x21 for KIT)

- binary features obtained by thresholding the heel and toe joint velocities to emphasize the foot ground contact (4)

MDM is trained on three different settings, namely, in mere generation mode (text-to-motion and action-to-motion) and in two editing modes, namely *temporal inpainting* and *spatial inpainting*, especially for the **upper body**.

In the generation mode, the model accepts as input a text prompt that directs the generation and, optionally, the motion length (in number of frames requested) of the generated motion. In the editing modes, the model additionally receives a motion tensor of dimensions *NxD* where *N* is the number of requested frames and *D* stands for the dimensionality of the motion joint representation, namely the feature vector fully describing the motion for each frame.

In the in-betweening (temporal inpainting) mode, the model additionally accepts a temporal mask determining **the frames that are destined to be edited** (e.g. from frame 60 to frame 100), while in the body part editing mode (e.g. the upper body editing model which is available in MDM model registry) a spatial mask determining **which joints are about to be edited (across all frames)**. Spatiotemporal masking is also possible with the code implementation available but has not been evaluated. An optimization towards this direction is proposed in one of the following chapters.

In both settings (motion generation and motion editing), context embeddings are used as supervision factors by the model. More specifically, a time embedding implying the noise level of the current diffusion step (ranging from 1 to 1000 in the MDM) and a text embedding, namely a feature vector created by the input text prompt tokenization

4.1.2 Network Architecture, Sampling & Optimization

A notable design-choice is the prediction of the sample, rather than the noise, in each diffusion step. This facilitates the use of established geometric losses on the locations and velocities of the motion, such as the foot contact loss. As we demonstrate, MDM is a generic approach, enabling different modes of conditioning and different generation tasks which are analyzed in the following paragraph.

In further detail, the model is illustrated in figure 4.1. G is implemented with a straightforward transformer encoder-only architecture. The transformer architecture is temporally aware, enabling learning arbitrary length motions, and is well-proven for the motion domain as shown in [75], [76]. The noise time-step t and the condition code c are each projected to the transformer dimension by separate feed-forward networks, then summed to yield the token z_{tk}. Each frame of the noised input x_t is linearly projected into the transformer dimension and summed with a standard positional embedding. z_{tk} and the projected frames are then fed to the encoder. Excluding the first output token (corresponding to z_{tk}), the encoder result is projected back to the original motion dimensions - after predicting the motion for each frame with the transformer encoder G - and serves as the prediction $\hat{x}_0$. Text-to-motion is implemented by encoding the text prompt to c with CLIP text encoder, which was extensively analyzed in paragraph 3.6.

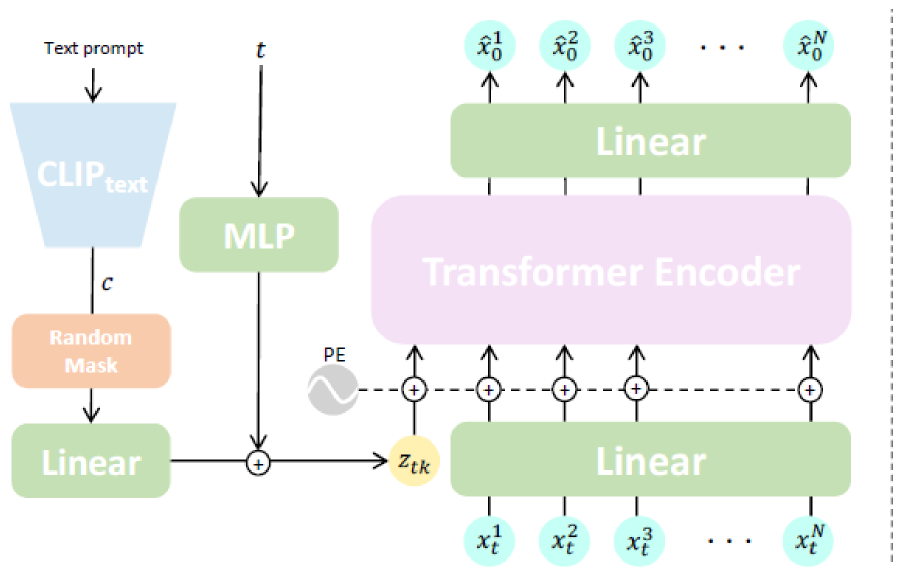

Text prompt
CLIP_text
c
Random Mask
Linear
t
MLP
PE
z_{tk}
Linear
Transformer Encoder
Linear
$\hat{x}_0^1$ $\hat{x}_0^2$ $\hat{x}_0^3$ ... $\hat{x}_0^N$
x_t^1 x_t^2 x_t^3 ... x_t^N

Figure 4.1. *MDM architecture overview. Source: [7]*

Diffusion is modeled as a Markov noising process $\{x_t^{1:N}\}_{t=0}^{T}$ where $x_0^{1:N}$ is sampled from the normal distribution:

$$q(\mathbf{x}_t^{1:N}|\mathbf{x}_{t-1}^{1:N}) = \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_{t-1}^{1:N}, (1-\alpha_t)\mathrm{I}) \tag{4.1}$$

In our context, conditioned motion synthesis models the distribution $p(x_0|c)$ as the reversed diffusion process of gradually cleaning x_T. Instead of predicting ϵ_t as shown in 2.5, the signal $\hat{x}_0 = G(\mathbf{x}, t, c)$ is directly derived by minimizing the objective:

$$\mathcal{L} = \mathrm{E}_{x_0\sim q(x_0|c), t\sim[1,T]}[||x_0 - G(x, t, c)||_2^2] \tag{4.2}$$

In the realm of motion generation, generative networks are typically regulated using geometric losses, as recommended by [75]. These losses serve to enforce physical principles and deter artifacts, promoting the emergence of natural and cohesive motion. MDM employs three prevalent geometric losses for regulation:

- positions (in case rotations are predicted) with the optimization term:

$$\mathcal{L}_{pos} = \frac{1}{N}\sum_{i=1}^{N}||FK(\hat{x}_0^i) - FK(x_0^i)||_2^2 \tag{4.3}$$

- foot contact with the optimization term:

$$\mathcal{L}_{foot} = \frac{1}{N-1}\sum_{i=1}^{N-1}||FK(\hat{x}_0^{i+1}) - FK(\hat{x}_0^i)\cdot f_i||_2^2 \tag{4.4}$$

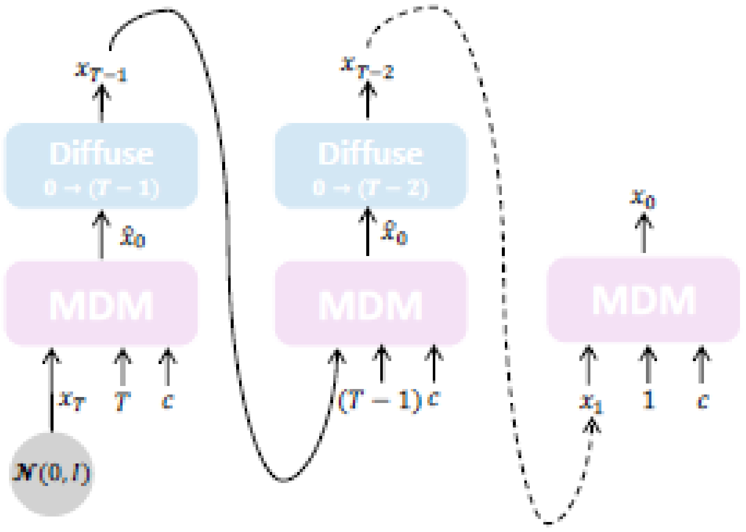


Figure 4.2. *MDM sampling. Source: [7]*

- velocities with the optimization term:

$$\mathcal{L}_{vel} = \frac{1}{N-1} \sum_{i=1}^{N-1} \|(x_0^{i+1} - x_0^i) - (\hat{x}_0^{i+1} - \hat{x}_0^i)\|_2^2 \tag{4.5}$$

$FK(*)$ denotes the forward kinematic function converting joint rotations into joint positions (otherwise, it denotes the identity function). $f_i \in \{0, 1\}^J$ is the binary foot contact mask for each frame i. Specifically relevant to feet, this loss determines their contact with the ground based on binary ground truth data. Essentially, it addresses the issue of foot-sliding by nullifying velocities when the feet are in contact with the ground.

The overall training loss is formulated as:

$$\mathcal{L} = \mathcal{L}_{simple} + \lambda_{pos}\mathcal{L}_{pos} + \lambda_{foot}\mathcal{L}_{foot} + \lambda_{vel}\mathcal{L}_{vel} \tag{4.6}$$

The models have been trained with $T = 1000$ noising steps and a cosine noise schedule.

Sampling from $p(x_0|c)$ is carried out iteratively as described in [1]. In every time step t, a clean sample $x_0 = G(x_t, t, c)$ is predicted and noises back to x_{t-1} from $t = T$ until x_0 is achieved as demonstrated in figure 4.2. The idea is that higher values of t construct high-level features of the object and lower levels of t construct more fine-grained features in the *image-motion*. The encoder G is trained using **classifier-free guidance** as explained in 2.4.2. In practice, G learns both the conditioned and the unconditioned distributions by randomly setting $c = \emptyset$; for 10% of the samples, such that $G(x_t, t, \emptyset)$ approximates $p(x_0)$. The bigger the percentage, the more we boost diversity against fidelity. Subsequently, when sampling G diversity and fidelity trade-off is adjusted by interpolating or even extrapolating the two variants using s, namely:

$$G_s(x_t, t, c) = G(x_t, t, \emptyset) + s(G(x_t, t, c) - G(x_t, t, \emptyset)) \tag{4.7}$$

The MDM models provided have been trained using $s = 2.5$.

4.2 PriorMDM

PriorMDM was inspired to address challenges such as the scarcity of annotated motion data, the focus of former methods primarily on single-person motions as well as a lack of refined control. In this study, three forms of composition based on diffusion priors are introduced: sequential, parallel, and model composition. Through sequential composition, the challenge of generating long sequences is addressed. A method called **DoubleTake** is proposed for inference, enabling the generation of extended animations comprised of sequences of prompted intervals and their transitions, using a prior trained exclusively for short clips. With parallel composition, progress is made toward two-person generation. Initially, using two fixed priors and a few examples of two-person training data, a communication block termed ComMDM is developed to coordinate interaction between the resulting motions. In this section, though, we will not focus on this novelty of PriorMDM. When it comes to employing model composition, individual priors are trained to complete motions that realize a prescribed motion for specific joints. Subsequently, DiffusionBlending, an interpolation mechanism, is introduced to effectively blend several such models, facilitating flexible and efficient fine-grained joint and trajectory-level control and editing.

Overall, the contribution of PriorMDM is demonstrating that pretrained diffusion-based motion generation models can serve as priors for composition, facilitating out-of-domain motion generation and streamlined control. Despite the perception of diffusion models requiring extensive data, three approaches are presented here, circumventing the cost hurdle through the utilization of the aforementioned prior. This enables the accomplishment of non-trivial tasks even in few-shot or zero-shot scenarios.

4.2.1 Motion Representation & Operational Modes

The types of composition that are located within the scope of the present work are:

- **sequential composition**, where short sequences are concatenated to create a single long and coherent motion yielding the **Double Take** method/mode. Double Take is capable of composing n generated motions over time, including the transitions between them, thus enabling the efficient generation of long motion sequences in a zero-shot manner. Because of the composite manner in which it generates, DoubleTake enables separate manipulation for each motion segment, all the while ensuring coherence in motion and transitions. Double Take was not included in the comparative evaluation as there exists no comparable method to date. However, it was leveraged to develop a bespoke denoising module, as will be illustrated later.

- **model composition**, where the motions generated by models with different control capabilities are blended together for composite control, yielding the **finetuned motion control** method/mode. Fine-tuning the prior towards accomplishing end-effector or root control yields remarkable results while controlling even just a single end-effector. Additionally, a DiffusionBlending technique is introduced which generalizes classifier-free guidance to compose together different fine-tuned models and thus enables cross combinations of keypoints' control on the generated motion. This enables precise and flexible control for human motion that comprises a key capability for animation systems. The finetuned motion control holds significant importance in our comparative evaluation, particularly concerning root (or pelvis) trajectory control.

The input representation format is identical to the respective generative and editing modes of MDM, with DoubleTake corresponding to MDM's Text-to-Motion generation utility and finetuned motion control to corresponding to the -spatial- editing mode. Hence, human motion is represented as a sequence of poses $X = \{x^i\}_{i=1}^N$ where $x_i \in R^D$ represent a single pose. Specifically, the SMPL representation is used for the experiments conducted on the BABEL dataset [61], including joint rotations and global positions on top of a single human identity ($\beta = 0$). For all other experiments, HumanML3D representation analyzed in 4.1.1 is used, which is composed of joint positions, rotations,velocities and foot contact information.

The sole difference between Double Take and MDM's text-to-motion generation exposed mode is that Double Take, instead of using a unique text prompt and a unique duration (framewise) as input, receives a series of text prompts and a series of durations, respectively. When fed with a single text prompt and duration, Double Take degenerates into classical MDM. Similarly, finetuned motion control differs from the body part editing mode in that it, by default, restricts only one joint (e.g.) the pelvis and leaves the rest joints' motion free to be diffused out of the sampled noise.

4.2.2 Network Architecture, Sampling & Optimization

The model employed as a generative prior as explained above is a **pretrained MDM**. MDM enables both high-quality generation and generic conditioning that together comprise a more than decent baseline for new motion generation tasks, which is the reason behind it being used as a baseline model to build upon, as happens with PriorMDM.

DoubleTake consists of two phases for every diffusion iteration:

- in the **first take**, the individual motions, or intervals, are generated together in the same batch, each aware of the context of its neighboring intervals. In the first take, each interval is generated as an individual sample in the denoised batch, such that each one is conditioned on its own text prompt and maintains a handshake with its neighboring intervals through the denoising process. Handshake denoted as τ is defined as a short (about a second long) prefix or suffix of the motion, such that the prefix of the current motion is forced to be equal to the suffix of the previous motion, ensuring transitions are smooth and eye-pleasing. Each interval maintains two such handshakes as displayed in figure 4.3. The handshake is maintained by simply overriding τ with the frame-wise average of the relevant suffix and prefix at each denoising step, allowing PriorMDM to generate arbitrarily long sequences that

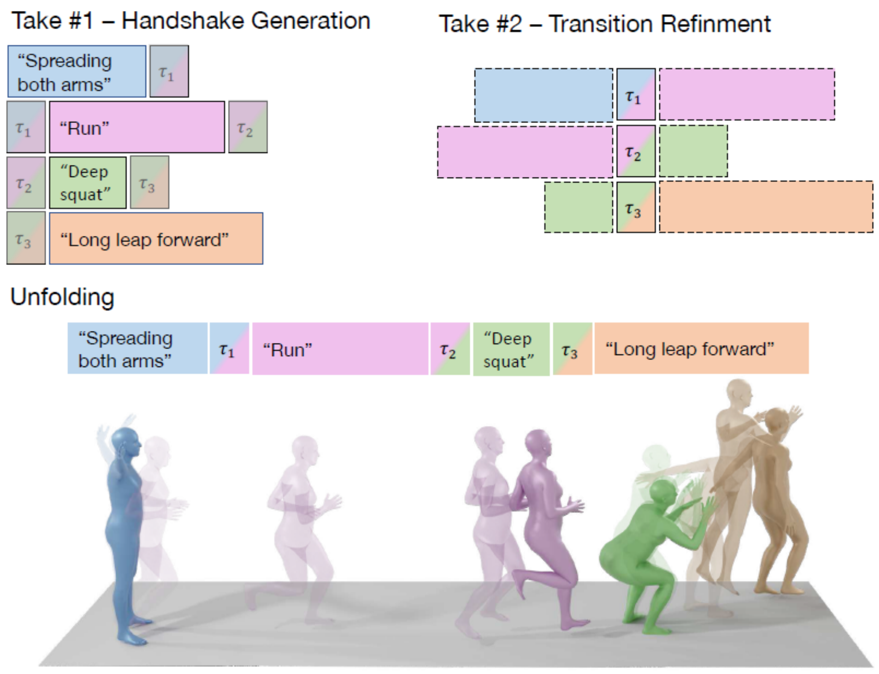


Figure 4.3. *Double Take method Overview. Source: [8]*

depend on the past and future motions while being aware of the whole sequence during the generation of each interval. Formally, handshakes are forced to be equal at the end of each denoising iteration as follows:

$$\tau_i = (1 - \vec{a}) \odot S_{i-1}[-h :] + \vec{a} \odot S_i[: h] \tag{4.8}$$

where S_i denotes the $i - th$ sequence $a_j = \frac{j}{h}, \forall j : j \in [0 : h)$.

- the **second take** refines the transitions between intervals to better match those generated in the previous phase. During this phase, the batch is reshaped as demonstrated in figure 4.3, such that in each sample a transition sandwich (S_i, τ_i, S_{i+1}) be obtained. Subsequently, the sandwich is partially noised for T' noising steps and denoised back to $t = 0$ under the suggested *soft-masking* feature to refine transitions. In further detail, in a regular inpainting mask, The content is either entirely drawn from the input or is entirely produced. A soft inpainting scheme is suggested in which each frame is assigned a soft mask value between 0 and 1 that dictates the amount of refinement the second take performs on top of the first take's outcome. To this end, the masks M_{soft} and M_{hard} are defined for the interval S and the handshake τ respectively with a short, b frames long, linear transition between the mask values as demonstrated in figure 4.4.

Finally, the long sequence is recreated through **unfolding** it, namely reshaping each sequence and transition back to its linear place as also shown in figure 4.4. Figure 4.3 visualizes the steps the Double Take method executes in an enlightening manner. Published Double Take version for long sequence generation employs a **fixed MDM** trained on the HumanML3D dataset. The training parameters mentioned above were set to:

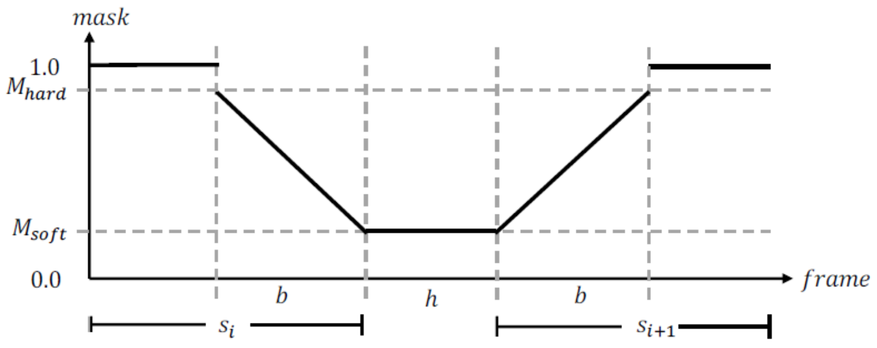


Figure 4.4. *Soft blending Overview. Source: [8]*

$T' = 700, M_{hard} = 0.85, M_{soft} = 0.1, b = 10, h = 20$ (yielding an one-second long transition interval).

Regarding the **finetuned motion control**, it practically comprises a mode aimed to generate full-body motion controlled by a user-define set of input features. These features may be root trajectory, a single joint trajectory or any combination of them. The motion synthesis in this paradigm is destined to produce a self-coherent motion that semantically adheres to the control signal. For instance, when specifying the root trajectory of a person to move backward, the output motion is expected to have the legs adjusted to walking backward. In essence, the method operates by masking out the noise applied to the ground-truth features about to be controlled during the forward pass of the diffusion process. This means that during training, the ground-truth control features propagate to the input of the model, and thus, the model learns to rely on these features when trying to reconstruct the rest of the features, as illustrated in Algorithm 1 in figure 4.5. On the **sampling** end, after the prediction x_0 of the model is obtained, the editing features are injected into it. Then, during the forward pass from x_0 to x_{t-1}, the control features are masked out of the noise in order for them to be cleanly propagated into the model.

To control cross combinations of the joints (i.e. both the root and the left wrist), the core idea of the classifier-free approach2.4.2 is exploited to introduce **DiffusionBlending**. The classifier-free approach suggests interpolating or extrapolating between the conditioned model G and the unconditioned model $G^{\emptyset}$. This idea was found capable of generalizing to any two "aligned" diffusion models G_a and G_b that are conditioned on c_a and c_b respectively. Sampling with two conditions simultaneously is implemented as:

$$G_s^{a,b}(\mathrm{X}_t, t, c_a, c_b) = G_s^a(\mathrm{X}_t, t, c_a) + s(G_s^b(\mathrm{X}_t, t, c_b) - G_s^a(\mathrm{X}_t, t, c_a)) \tag{4.9}$$

with the scale parameter s trading-off the significance of the two control signals. All motion control experiments, and the respective models available for downloading, were conducted above HumanML3D dataset with text-conditioning and a classifier-free guidance scale of 2.5 and their weights were initialized using the same classical MDM instance. Batch size was set to 64.

Algorithm 1 Fine-tuning method

```
repeat
    $x_0 \sim q(x_0)$
    $t \sim \text{Uniform}(\{1, \dots, T\})$
    $\epsilon \sim \mathcal{N}(0, I)$
    $\epsilon\,[trajectory] = 0$                              ▷ **Our addition**
    Take gradient descent step on:
        $\nabla_\theta \| x_0 - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t\right) \|$
until Converged
```

Algorithm 2 Sampling method

```
$x_0^{(T)} = 0$
for $t = T, \dots, 0$ do
    $x_0^{(t)}\,[trajectory]$ =given trajectory ▷ **Original in-painting**
    $\epsilon \sim \mathcal{N}(0, I)$
    $\epsilon\,[trajectory] = 0$                              ▷ **Our addition**
    $x_0^{(t-1)} = \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t\right)$
end for
```

Figure 4.5. *Soft blending Overview. Source:[8]*

4.3 OmniControl

OmniControl[9] is a novel approach introduced for integrating flexible spatial control signals into a text-conditioned human motion generation model based on the diffusion process. OmniControl is special in the sense that it is capable of integrating flexible spatial control signals over any joint at different times using a single model. It introduces analytic spatial guidance to ensure the generated motion closely adheres to the input control signals while also incorporating realism guidance to refine all joints' motion for producing more coherent motion with no visually displeasing artifacts. Both spatial and realism guidance are deemed crucial and mutually complementary for achieving a balance between control accuracy and motion realism. Through their combination, OmniControl generates motions that are realistic, coherent, and aligned with spatial constraints. OmniControl is rumored to not only significantly improve pelvis control compared to state-of-the-art methods but also to yield promising results when incorporating constraints on other joints. The main novelty OmniControl introduces compared to other state-of-the-art methods is a control module that uses both spatial and realism guidance to effectively balance the control accuracy and motion realism in the generated motion. Inspired by classifier guidance (analyzed in 2.4.1) and ControlNet[77], hybrid guidance, consisting of spatial and realism guidance, is introduced to incorporate spatial control signals into human motion synthesis.

4.3.1 Motion Representation & Operational Modes

OmniControl model receives a textual prompt **p** and an additional spatial control signal $c \in R^{NxJx3}$ as input aiming at generating a human motion sequence $\mathbf{x} \in R^{NxD}$, where N is the motion sequence length in frames, J is the number of joints and D stands for the dimension of the human pose feature vector (e.g. $D = 263$ for the HumanML3D dataset). The spatial constraints, denoted as c, encompass the xyz positions of each joint across all frames. Only a subset of joint locations are actually specified as spatial constraints for human motion generation, with the positions of other joints being assigned zero values. This approach allows for flexible specification of which joints are controlled, enabling control over motion generation for any joint at any given time instance (keyframe).

In human motion generation, the redundant data representation suggested in **MDM**[7], and additionally adopted by **PriorMDM**[8], which include pelvis velocity, local joint positions, velocities and rotations of other joints in the pelvis space as well as the foot contact binary labels is the most widespread representation technique in literature lately. Such representations are easier to learn and can produce realistic human motions. However, they seem to struggle to handle sparse (in time dimension) constraints on the pelvis and incorporate any spatial control signal on joints other than the pelvis.

To overcome this limitation, the relative representations are transformed into global ones within the proposed spatial guidance method. This enables flexible control over any joints at any given time, according to the authors. The model continues to utilize local human pose representations as its input and output. As a result, the control signal remains effective for all preceding frames beyond the keyframe of the control signal, as the gradients can be backpropagated to them. This enables the spatial guidance to densely perturb the motions, even when the spatial constraints are extremely sparse.

Regarding the operational modes of OmniControl method, they practically constitute a single mode enabling spatiotemporal editing of all joints at any time as the authors report. All different ‘modes’ of running can be segregated in sub-cases substantially defined by the spatial control signal - for the frames and the joints for which the global coordinates are equal to 0, there is no constraint, while for all the non-zero values the constraint should be respected to the maximum - but not absolute - extent. The *sub-modes* we are referring to are:

1. dense spatial control in pelvis (spatial control for all frames)
2. sparse spatial control in pelvis (spatial control for a specific range of frames)
3. dense spatial control in limbs’ joints
4. dense spatial control onto head
5. sparse spatial control for limbs’ joints (for a specific range of frames)
6. dense spatial control with multiple joints (e.g. two limb joints)
7. motion in-betweeening, namely only first and last frame comprise constraints for one joint or even a combination of joints (sparse type of spatial control)

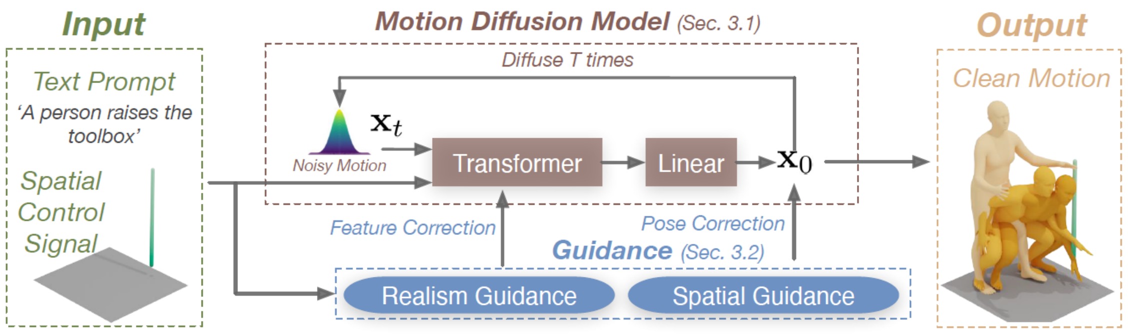


Figure 4.6. *Architecture Overview of OmniControl's model. Source:[9]*

4.3.2 Network Architecture, Sampling & Optimization

The model learns the reversed diffusion process of gradually denoising $\mathbf{x}_t$ starting from the pure Gaussian noise $\mathbf{x}_T$

$$P_\partial(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{p}) = \mathcal{N}(\mu_t(\partial), (1 - \alpha_t)\mathrm{I}) \tag{4.10}$$

where $\mathbf{x}_t \in \mathrm{R}^{NxD}$ denotes the motion at the t^{th} noising step and there are T diffusion denoising steps in total. Instead of predicting the noise at each diffusion step, OmniControl directly predicts the final clean motion $\mathbf{x}_0(\partial) = \mathrm{M}(\mathbf{x}_t, t, \mathbf{p}; \partial)$ where M is the motion generation model with parameters ∂. An overview of the model's architecture is illustrated in figure 4.6.

The spatial guidance module applies an analytic function $G(\mu_t, \mathbf{c})$ to approximate a classifier, enabling multiple efficient perturbations of the generated motion, which assess how closely the joint of the generated motion aligns with a desired spatial location $\mathbf{c}$. In spatial guidance, the gradient of the analytic function is employed to guide the generated motion in the desired direction by perturbing the predicted mean in every denoising step t using the following formulation:

$$\mu_t = \mu_t - \tau \nabla_{\mu_t} G(\mu_t, c) \tag{4.11}$$

where τ controls the strength of the guidance. G measures the Euclidean distance between the joint locations of the generated motion and the spatial constraints:

$$G(\mu, \mathrm{c}) = \frac{\sum_n \sum_j \sigma_{nj} ||c_{nj} - \mu^g_{nj}||_2}{\sum_n \sum_j \sigma_{nj}} \tag{4.12}$$

with $\mu^g = \mathrm{R}(\mu)$ where σ_{nj} is a binary value indicating whether the spatial control signal $\mathbf{c}$ contains a valid value at frame n for joint j. R($\cdot$) operator expresses the conversion from local joint positions to global absolute positions. An overview of the spatial guidance module is depicted in figure 4.7.

While the spatial guidance effectively enforces the controlled joints to adhere to the input control signals, it may leave other joints unchanged. This lack of influence on other joints may result in unrealistic motions, yielding artifacts such as motion incoherence

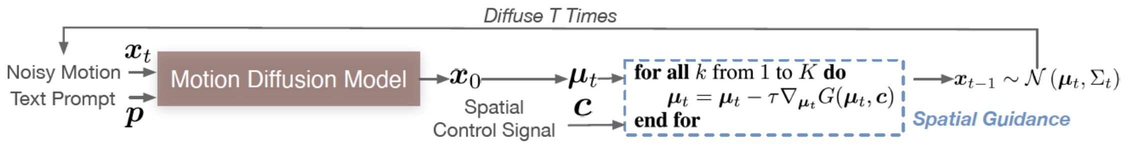


Figure 4.7. *Detailed overview of the spatial guidance module. Source:[9]*

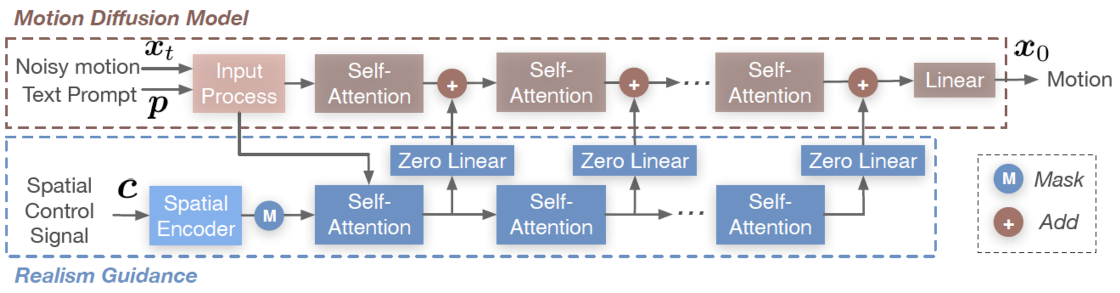
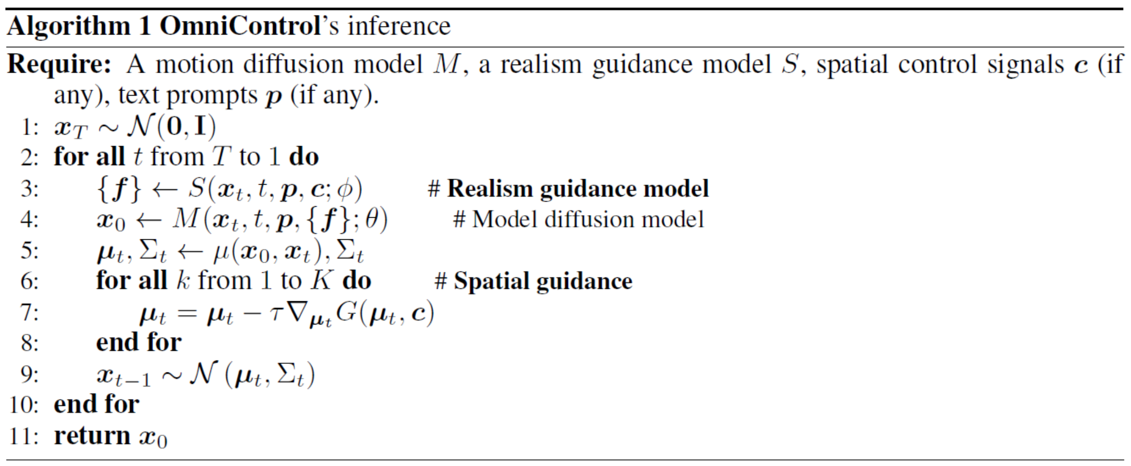


Figure 4.8. *Detailed overview of the realism guidance module. Source:[9]*

and **foot sliding**. **Realism guidance** module steps up to address these challenges by integrating into the network a trainable copy of the Transformer encoder in the motion diffusion model[7] to learn to enforce the spatial constraints. each of the Transformer layers is connected by a linear layer with both weight and bias initialized with zeros, rendering them ineffective for control at the outset. However, as training progresses, the realism guidance model learns the spatial constraints and incorporates the learned feature corrections into the corresponding layers of the motion diffusion model to implicitly adjust the generated motions as demonstrated in figure 4.8. In short, the residuals with respect to the features in each attention layer of the motion diffusion model are generated by the realism guidance. These residuals have the capability to densely and implicitly perturb the entire-body motion.

In detail, a spatial encoder F is employed to decode the spatial control signals **c** at each frame independently (as figure 4.8 implies). To effectively handle the sparse control signals in time, the features at frames where there are no valid control signals,$\mathbf{f}_n = o_n \mathrm{F}(\mathbf{c}_n)$,are masked out of noise. o_n is a binary label that is an aggregation of σ_{nj} in equation 4.12 such that o_n is valid when any of $\{\sigma_{nj}\}_{j=1}^{J}$ is equal to 1. Otherwise, it is marked as invalid by setting it to 0. The features $\mathbf{f}_n$ of spatial control signals at frame n are injected into the trainable copy of the Transformer. This aids the subsequent attention layers in identifying the locations of valid spatial control signals, thereby adjusting the corresponding features accordingly.

The model was trained using a batch size equal to **64** and a learning rate equal to **1e – 5** while using an **AdamW optimizer**[78]. CLIP is used for textual prompts conversion into feature emdeddings. The baseline motion diffusion model is a **classical MDM**[7]. Both the motion diffusion model and the realism guidance module, namely an MDM copy, resume the pretrained weights of MDM and are subsequently fine-tuned jointly. The spatial guidance is also used in training time. During training the textual prompt **p** is randomly masked using classifier-free guidance[79]. HumanML3D and KIT-ML benchmark datasets



Algorithm 1 OmniControl's inference

```
Require: A motion diffusion model $M$, a realism guidance model $S$, spatial control signals $\boldsymbol{c}$ (if
    any), text prompts $\boldsymbol{p}$ (if any).
 1: $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 2: for all $t$ from $T$ to 1 do
 3:     $\{\boldsymbol{f}\} \leftarrow S(\boldsymbol{x}_t, t, \boldsymbol{p}, \boldsymbol{c}; \phi)$      # Realism guidance model
 4:     $\boldsymbol{x}_0 \leftarrow M(\boldsymbol{x}_t, t, \boldsymbol{p}, \{\boldsymbol{f}\}; \theta)$      # Model diffusion model
 5:     $\boldsymbol{\mu}_t, \Sigma_t \leftarrow \mu(\boldsymbol{x}_0, \boldsymbol{x}_t), \Sigma_t$
 6:     for all $k$ from 1 to $K$ do      # Spatial guidance
 7:         $\boldsymbol{\mu}_t = \boldsymbol{\mu}_t - \tau \nabla_{\boldsymbol{\mu}_t} G(\boldsymbol{\mu}_t, \boldsymbol{c})$
 8:     end for
 9:     $\boldsymbol{x}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)$
10: end for
11: return $\boldsymbol{x}_0$
```

Figure 4.9. *OmniControl inference method's pseudo-code. Source:[9]*

were seelcted to perform training on as well as to validate the effectiveness of OmniControl. Finally, **DDPM** sampling[1] is employed with $T = 1000$ denoising steps.

To summarize, OmniControl was introduced as a potent approach for controlling human joints dynamically during text-driven motion generation using the diffusion process. By synergizing spatial and realism guidance, OmniControl achieves realistic human motion generation while adhering to input spatial control signals. The algorithm effectively dictating the whole inference process is displayed in the code snippet of figure 4.9.

4.4 Guided Motion Diffusion (GMD)

Despite the rapid advancement of diffusion models in the context of generating human motion based on natural language descriptions, incorporating spatial constraints, such as prescribed motion trajectories and obstacles, was a challenge yet to be addressed. Guided Motion Diffusion (GMD)[10] is a novel approach targeting to respond to this challenge by integrating spatial constraints into the motion generation process. GMD introduces an effective feature projection scheme, which adjusts the motion representation to enhance coherence between spatial information and local poses. Alongside a new imputation formulation, this enhancement ensures that the generated motion adheres reliably to spatial constraints, including global motion trajectories. Furthermore, in scenarios with sparse spatial constraints (i.e. sparse keyframes), a new dense guidance approach converting a sparse signal into denser signals is suggested. **GMD is method analogous to OmniControl** in terms of operational modes but differs significantly as will be elaborated on in the forthcoming sections.

For notational purposes, let **y** be a partial target value in an input **x** that are about to be imputed. The imputation region of **y** on **x** is denoted as M_y^x and a projection P_y^x that resizes **y** to that of **x** by filling in zeros. This segregation is necessary for the comprehension of the following sections.

4.4.1 Motion Representation & Operational Modes

The denoising diffusion models receive the root features as input, namely a motion tensor with dimensions $(L, 4)$ where L is the number of frames of the desired motion. Additionally, a text prompt is provided.

The opetational modes of GMD are the following:

- **trajectory-conditioned generation**: This task aims at generating a realistic motion $\mathbf{x}$ that matches a given trajectory $\mathbf{z}$. The objective is to minimize the distance between the generated motion and the given trajectory. Despite the apparent simplicity of this task, a traditional DPM faces the challenge of ensuring coherence in the generated motion. However, our emphasis projection method effectively tackles this issue.

- **keyframe-conditioned generation**: The locations of ground positions at specific times can be used to define locations that we wish the generated motion to reach. This task is a generalized version of the trajectory-conditioned generation where only a partial and potentially sparse trajectory $\mathbf{y} \in R^{2xM}$ is given with M reflecting the number of keyframes provided. A mask M^z_y describe the key motion steps by mapping the sparse trajectory to a dense one is also needed. Generating both the trajectory and motion simultaneously under a conditioning signal can pose challenges and potentially lead to poor quality motion. To tackle this issue, a two-step approach is proposed by generating a trajectory z that satisfies the keyframe locations, at first, and then generating the motion x given the trajectory z as depicted in figure 4.10.

- **obstacle avoidance motion generation**: namely navigating around obstacles while traveling from point A to B. This problem is dealt with using two goal functions: one that navigates from A to B called G^{loc}_x and a second one that pushes back when the human model crosses the obstacle's boundary, called G^{obs}_x, which consumes a parameter c that expresses the safe distance from the obstacles. Obstacles are considered as cyclic objects, thus can be explicitly defined by their center in the trajectory plane (e.g. (x, z)) and their radius r. These two goal functions are combined additively to obtain the final goal function: $G_x(\mathbf{x}) = G^{loc}_x(\mathbf{x}) + G^{obs}_x(\mathbf{x})$

4.4.2 Network Architecture, Sampling & Optimization

Motion representation in GMD is different compared to the diffusion-based techniques presented in previous sections. GMD authors claim that the **HumanML3D** 263-**dimensional representation** uses just 4 values to represent global orientation and 259 values for local pose in each frame, yielding considerable sparseness in global orientation representation. This imbalance may lead the model to prioritize local pose details excessively, causing it to perceive guided global orientation as noise. Consequently, discrepancies such as foot skating can arise. Additionally, sparse spatial control signals, such as target locations on the ground, are defined only at a few keyframes preventing

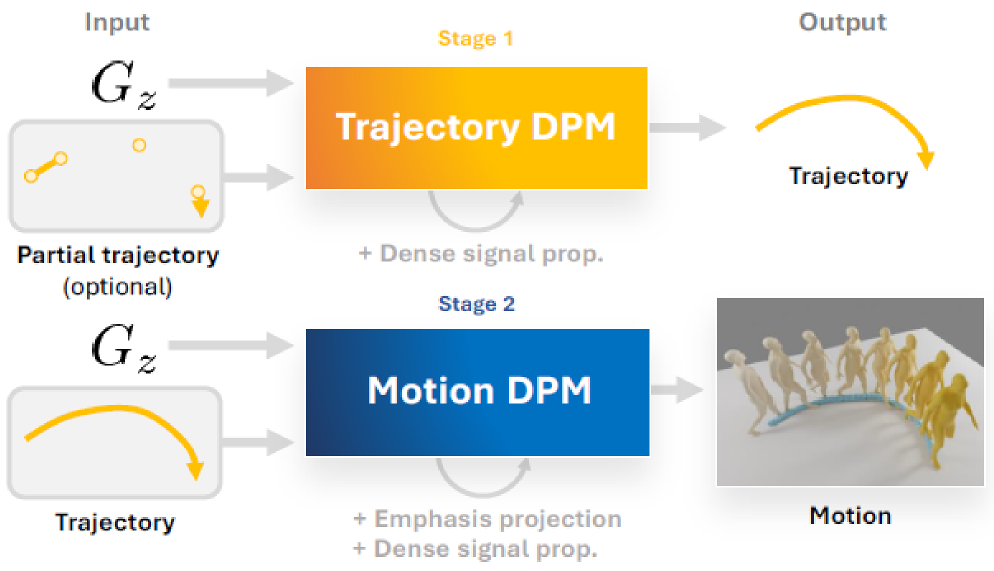


Figure 4.10. ***GMD's two stage pipeline***. *DPM stands for diffusion probabilistic model. Source:[10]*

the model from adhering to such sparse guidance. This difficulty arises because guiding the motion generation model with **sparse control signals is akin to guiding an image diffusion model with only a few pixels**.

GMD introduces two novel modules to overcome these difficulties:

- **Emphasis projection**: a technique to adjust the relative importance of different parts of the representation vector. This adjustment fosters coherence between spatial information and local poses, facilitating spatial guidance.
- **Dense signal propagation**: a conditioning method aimed at addressing the sparse guidance issue.

The integration of these two submodules into the backbone Unet-based architecture[80] results in an effective spatially controllable motion generation method that enables the unexplored synthesizing of motions based on free-text and spatial conditioning. GMD is a two-stage pipeline visualized in figure 4.10.

The high-level line of reasoning used in both stages of this two-stage pipeline is modeling a full-body human motion that satisfies a certain scalar goal function $G_x(\cdot)$ that takes in a motion representation $\mathbf{x}$ and measures the distance of $\mathbf{x}$ from the goal aiming to zero it out at some point, thus, in mathematical terms this means that, let X be a random variable associated with $\mathbf{x}$, the goal is to model the following conditional probability using a diffusion-based model: $p(\mathbf{x}|G_x(\mathrm{X}) = 0)$. This can be extended to $p(\mathbf{x}|G_x(\mathrm{X}) = 0, d)$ where d is a conditional signal such as text prompts. Numerous challenging tasks in motion modelling can be encapsulated into a goal function G_z that only depends on a prescribed trajectory $\mathbf{z}$ and not the entire motion $\mathbf{x}$. Let $\mathbf{z} \in \mathrm{R}^{LxM}$ be the trajectory part of $\mathbf{x}$, where M is the motion sequence length framewise and $L = 2$ describing the ground pelvis position of human body, and $\mathbf{z}^{(i)}$ be the trajectory position in time step i for notation purposes.

On the **emphasis projection** end, the most straightforward approach for minimizing

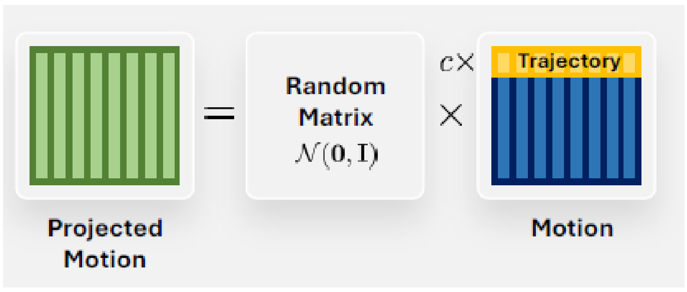


Figure 4.11. *Emphasis projection module (matrix projection). Source:[10]*

the goal function $G_{\mathbf{z}}(\cdot)$ is analyzing how trajectories that minimize $\mathbf{z}^* = argmin_{\mathbf{z}} G_{\mathbf{z}}(\mathbf{z})$ look like. For a trajectory conditioning task, a whole trajectory $\mathbf{z}^*$ is already provided, thus the task is to generate the rest of the motion $\mathbf{x}$. Bearing this in mind, imputation and inpainting techniques are employed by supplying the diffusion model with a $\mathbf{x}-$*shaped* motion, comprising a projection of trajectory $\mathbf{z}$ onto the entire motion $\mathbf{x}$, to guide the generation process. Since the imputing trajectory $\mathbf{z}^*$ is a small subset of the whole motion $\mathbf{x}(L \ll N = 263)$, it is observed that the diffusion process 'ignores' the change from imputation and fails to make appropriate changes on the rest of $\mathbf{x}$. This results in an incoherent local motion that is not aligned or well coordinated with the imputing trajectory. This problem is encountered by giving more emphasis on the trajectory part of motion $\mathbf{x}$ through the **emphasis projection** module. This is achieved by utilizing a random matrix $\mathrm{A} = \mathrm{A}'\mathrm{B}$ where $\mathrm{A} \in \mathrm{R}^{NxN}$ is a matrix with elements randomly sampled from $\mathcal{N}(0, 1)$ and $\mathrm{A} \in \mathrm{R}^{NxN}$ is a diagonal matrix whose trajectory-related diagonal indexes are c and the rest are 1 for emphasizing those trajectory elements, as illustrated in figure 4.11. Rotation and ground location of the pelvis, (rot, x, z), are emphasized in $\mathbf{x}$ by c times and the projection of $\mathbf{z}$ on $\mathbf{x}$ is formulated as $x_{proj} = \frac{1}{N-3+3c^2}\mathrm{A}x$. The noising process of the projected motion becomes:

$$q(\mathbf{x}_t^{proj}|\mathbf{x}_0^{proj}) = \mathcal{N}(\sqrt{a_t}\mathbf{x}_0^{proj}, (1-a_t)\mathbf{I}) \tag{4.13}$$

There in no variation on how the denoising diffusion model that works on the projected motion $p_\partial(\mathbf{x}_{t-1}^{proj}|\mathbf{x}_t^{proj})$ operates and treats $\mathbf{x}_t^{proj}$ - identically to how it would handle $\mathbf{x}_t$. The emphasis projection module completes its operation by imputing on the produced projected motion $\mathbf{x}^{proj}$.

Dense signal propagation module, on the other hand, leverages another way to minimize the goal function $G_{\mathbf{z}}(\cdot)$ is by adjusting the sample of each diffusion step $\mathbf{x}_{t-1}$ toward a region with lower G_z, namely using classifier guidance. The direction of change corresponds to a score function $\nabla_{\mathbf{x}_t} logp(G_x(\mathrm{X}_t) = 0|\mathbf{x}_t)$ which can be approximated as a direction $\Delta_{\mathbf{x}_0} = \nabla_{\mathbf{x}_0} G_z(P_x^z \mathbf{x}_{0,\partial})$ that reduces the goal function. The generative process by modifying the diffusion model's prediction as $\mathbf{x}_0 = \mathbf{x}_{0,\partial} + \Delta_{\mathbf{x}_0}$. While imputation requires the minimizer $\mathbf{z}^*$ of G_z, which might not be easy to obtain or may not be unique, this trick only requires the easier-to-obtain direction of change.

To turn a sparse signal into a dense signal, domain knowledge is necessary. One way to achieve this is by using a denoising function $\{(\mathbf{x}_t) = \mathbf{x}_0$, which is trained on a motion dataset to denoise by gathering information from the nearby motion frames. With the ability to relate a single frame to a plethora of other frames, the denoising function is capable of expanding a sparse signal into a denser one. Backward propagation through the denoising function { can be employed to take advantage of this. Therefore, a dense classifier guidance can be obtained as follows:

$$\nabla_{\mathbf{x}_t} logp(G_x(\mathrm{X}_t) = 0|\mathbf{x}_t) \approx \nabla_{\mathbf{x}_t} G_z(P^z_x f(\mathbf{x}_t)) \tag{4.14}$$

with $P^z_x f(\mathbf{x}_t)$ being $\mathbf{z}$-shaped. While an external function can be used as f, it is observed that the existing denoising diffusion model $\mathbf{x}_{0,\partial}(\mathbf{x}_t)$ itself is a motion denoiser, thus can be used to turn a sparse signal into a dense signal without the need for an additional model. In practice, this process amounts to computing the gradient of G with respect to $\mathbf{x}_t$ through $x_{0,\partial}(\mathbf{x}_t)$ using autodifferentiation.

The next structural component of the **dense guidance signal** module concerns the application of classifier guidance together with imputation. Whenever available, it is desirable that signals from both imputation and classifier guidance techniques to help guide the generative process be utilized. Imputation is explicit but may encounter sparsity in time, while classifier guidance is indirect but dense. To use the direct signal from imputation wherever available (with mask M^x_z) and the rest from classifier guidance (with mask $1 - \mathrm{M}^x_z$), imputation-aware classifier guidance can be formulated as:

$$\mu_t = \tilde{\mu}_t - (1 - M^x_z) \odot s\Sigma_t \nabla_{\mathbf{x}_t} G_z(P^z_x f(\mathbf{x}_t)) \tag{4.15}$$

where $\tilde{\mu}$ is an imputed sampling mean replaced by $\tilde{\mu}_{proj}$.

Lastly, a design choice on the backbone diffusion model needs to be addressed. A crucial priority in this context is to minimize the influence of model's bias under the guidance signal. Conceptually, the denoising diffusion models usually make less and less change near the final outcome. This is in tandem with the guidance signal that gradually decreases over time due to Σ_t. Thus an ϵ_∂ model is qualified over a $\mathbf{x}_{0,\partial}$ model for the trajectory refinement stage as the former approach forces the model to maximize its influence on the sampling mean at $t = T$ which is alignment with the guidance signal which begins at a maximum value and gradually fades out.

The trajectory and motion architectures of **GMD** are both based on U-NET with Adaptive Group Normalization(AdaGN)[3]. However, this model adapted this model for sequential prediction tasks by using $1D$ convolutions. The architecture overview is depicted in figure 4.12 while the Adaptive Group Normalization is depicted in figure 4.13. A simplified overview of our GMD's **1D UNET + AdaGN** architecture that is designed to process two input signals: the time step $\psi(t)$ and a text-prompt embedding $\mathbf{w}$. The time step is encoded using sinusoidal functions, while the text-prompt embedding is generated by the CLIP3.6 text encoder model. The ResBlock with Adaptive Group normalization component of the model uses the conditioning signal from the MLP, shared across all ResBlocks, is projected by first applying a Mish activation and then a resizing linear projection specific to each ResBlock. All kernel sizes are 5.

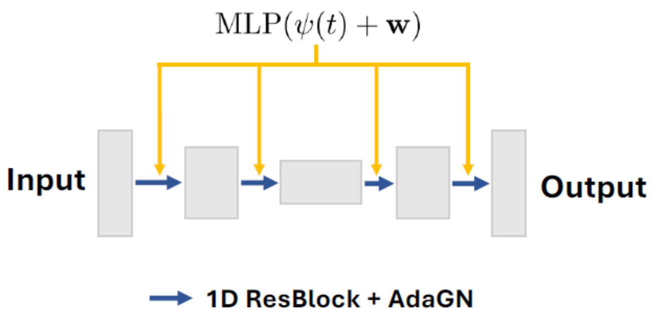


Figure 4.12. *Source:[10]*

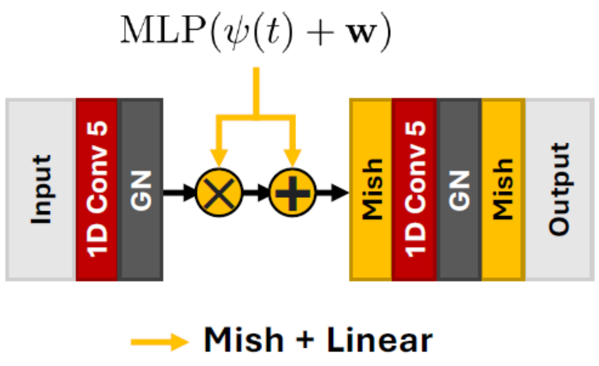


Figure 4.13. *Source:[10]*

Regarding the training procedure, a batch size of 64 for the motion DPM and a batch size of 512 for the trajectory DPM. No dropout was used in all of the GMD's models: both trajectory and motion. AdamW optimizer with a learning rate of 0.0001 and weight decay of 0.01 was employed. Gradient clipping with a threshold value for the norm set to 1, as it was found to increase training stability. The trajectory DPM optimized for noise ϵ prediction, while the motion DPM optimized for motion $\mathbf{x}_0$ prediction.

4.5 MotionDiffuse

MotionDiffuse[11] is a flexible and adaptable framework for motion generation, capable of producing a wide range of motions aligned with detailed textual descriptions. Notably, it is positioned as a counterpart to the MDM[7] method, with both being released around the same time. In contrast to traditional DDPM[1] methods, which are limited to generating fixed-size outputs, a Cross-Modality Linear Transformer is introduced to enable motion synthesis with variable motion lengths. Rather than establishing a direct mapping between textual inputs and motion outputs as seen in prior work[81], the method proposed aims to guide the generation process softly with input texts. To preserve uncertainties during the denoising process, the noise terms are conditioned on the input texts using multiple transformer decoder layers for each denoising step, maintaining the influence of textual conditions on motion generation remains probabilistic and, thus, promoting the creation of diverse motion sequences based on the provided textual prompts.

It was the first method to incorporate **DDPM** in motion synthesis paradigm permitting the efficient conditioning on text descriptions to generate motions in a probabilistic style. Moreover, MotionDiffuse stands out as a pioneering method in multi-level manipulation, offering the capacity to manage fine-grained text descriptions that engage the entire body (e.g., 'a person is drinking water while walking') and time-varying signals (e.g., 'a person is walking and then running').

4.5.1 Motion Representation & Operational Modes

The motion sequence Θ is an array of (∂_i), where $i \in \{1, 2, .., F\}$ represents the pose state where $\partial_i \in \mathrm{R}^D$ in the $i-th$ frame and F is the number of frames. The representation of each pose state ∂_i is distinct in different datasets. It generally contains joint rotation, joint position, joint velocity, and foot contact conditions. MotionDiffuse is robust to the various motion representations. The HumanML3D dataset and KIT dataset are employed to train and evaluate the proposed methods for the text-driven motion generation task, thus the respective input motion representations can be consumed by the MotionDiffuse model.

The operational settings upon which the MotionDiffuse architecture was trained on are:

- Text-conditioned Motion Generation
- Action-conditioned Motion Generation (not examined in the present work)
- *Spatially-diverse text-to-motion generation task (T2M-S)* . T2M-S requires the generated motion sequence to contain multiple actions on different body parts (e.g. 'a person is running and drinking water simultaneously').
- *Temporally-diverse text-to-motion generation task (T2M-T)*. T2M-T model is expected to generate a long motion sequence, which includes multiple actions in a specific order spanning over different time intervals (e.g. 'a person is walking and then running').

4.5.2 Network Architecture, Sampling & Optimization

The backbone model of the *MotionDiffuse* architecture is a denoising diffusion probabilistic model (DDPM), thus, obviously, **DDPM** sampling is employed. For the denoising process, we propose a Cross-Modality Linear Transformer to process input sequences conditioned on the given text prompts. Two types of conditional control signals were additionally used for experimentation:

- *part-aware text controlling*, namely assigning different text conditions to different body parts so that accurate control of the individual parts of the body is achieved.
- *time-varied controlling*, namely division the whole sequence into several parts and assigns independent text conditions for each interval. This provides the capability to synthesize arbitrary-length motion sequences that incorporate several actions.

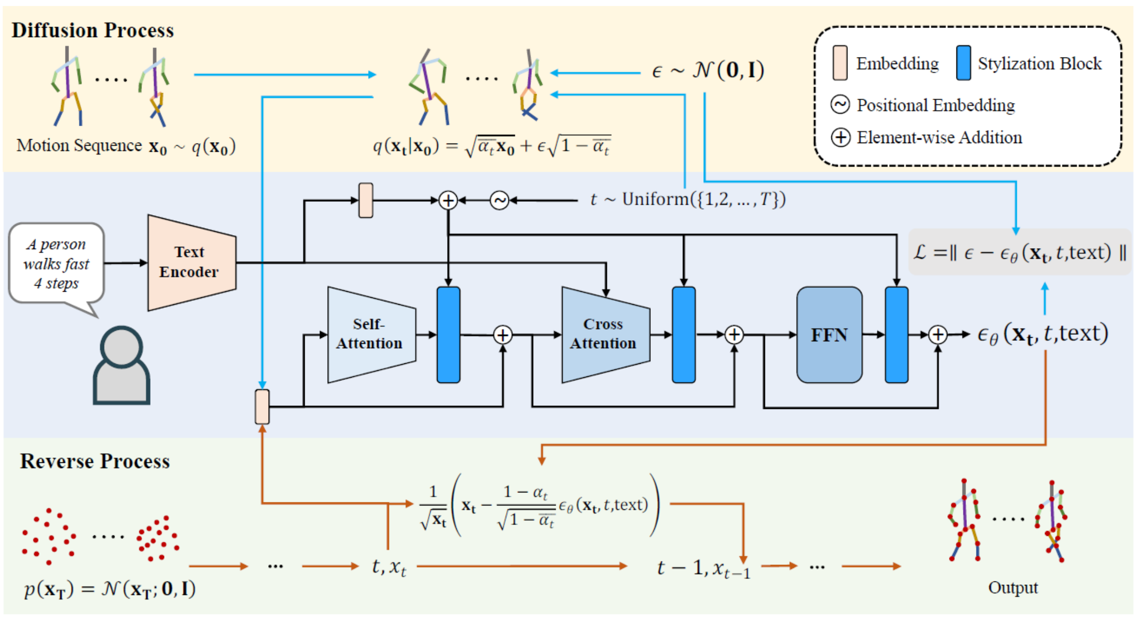

Diffusion Process
Motion Sequence $x_0 \sim q(x_0)$
$q(x_t|x_0) = \sqrt{\bar{\alpha}_t x_0} + \epsilon\sqrt{1-\bar{\alpha}_t}$
$\epsilon \sim \mathcal{N}(0, I)$
Embedding
Stylization Block
Positional Embedding
Element-wise Addition
$t \sim$ Uniform({1,2, ..., T})
A person walks fast 4 steps
Text Encoder
Self-Attention
Cross Attention
FFN
$\mathcal{L} = \| \epsilon - \epsilon_\theta(x_t, t, \text{text}) \|$
$\epsilon_\theta(x_t, t, \text{text})$
Reverse Process
$\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t, \text{text})\right)$
$p(x_T) = \mathcal{N}(x_T; 0, I)$
t, x_t
$t-1, x_{t-1}$
Output

Figure 4.14. *Overall network architecture of MotionDiffuse. Source:[11]*

The overall pipeline of MotionDiffuse architecture is displayed in figure 4.14. Based on the analysis carried out in chapter and, more specifically, using the equations 2.4 and 2.5, it is easy to sample noise ϵ and then directly generate $\mathbf{x}_t$. Instead of predicting $\mathbf{x}_{t-1}$, the choice to predict the noise term ϵ is qualified. Hence, the constructed network is about to fit $\epsilon_\partial(\mathbf{x}_t, t, c)$, where c is a textual description. The model parameters are optimized to decrease a mean squared error as:

$$\mathcal{L} = \mathrm{E}_{t\in[1,T],\mathbf{x}_0\sim q(\mathbf{x}_0),\epsilon\sim\mathcal{N}(0,\mathbf{I})}[\|\epsilon - \epsilon_\partial(\mathbf{x}_t, t, c)\|] \tag{4.16}$$

This is the only loss function optimized during model training. To generate samples from the given text description, the denoising of sequence $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; 0, \mathbf{I})$. Therefore, the motion sequence is denoised step by step and finally a clean motion sequence $\mathbf{x}_0$ is obtained, which is conditioned on the given text.

The backbone diffusion model is a **Cross-Modality Linear Transformer**, which is consisted a text encoder and a motion decoder. For the text features' extraction, a classical transformer[70] is employed. The input data first passes through an embedding layer to obtain the embedding feature from raw text and then is further processed by a series of transformer blocks. Each block contains two components - a **multi-head attention** module and a **feed-forward network (FFN)**. To enhance the generalization ability, the parameter weights of a pre-trained CLIP model3.6 are used to initialize the first several layers. Overall, we could declare that the text encoder is the same with text encoder described in the **CLIP ViTB/32**[6] with four more transformer encoder layers.

The Linear Self-attention module aims at enhancing motion features by modeling correlations between different frames. The principal advantage of self-attention is to get an overview of the input sequence and is thus useful to predict the injected noise ϵ. Due to the increased computational complexity of the attention weights' estimation, especially when the motion sequence length is high, Efficient Attention as introduced in [82] is preferred in order to accelerate the self-attention module. Efficient attention in the diffusion model provides an additional advantage by explicitly aggregating global information. This contrasts with classical self-attention, which tends to focus more on pair-wise relations. The inclusion of global information in the feature map enhances the understanding of the semantic meaning of the motion sequence.

Text embeddings are fused into motion sequences as a conditional signal through the operation of the **Linear Cross Attention**, which guarantees that the generated motion is conditioned on the given text. As also mentioned before, in each denoising step, the output is conditioned on the timestamp t except for the textual provided. The **Stylization Block** component is responsible for injecting timestamp t into the generation process. This block is applied after each Linear Self-attention block, Linear Cross-attention block and FFN block as shown in 4.14. Overall, the latent dimension of the text encoder and the motion decoder are 256 and 512,respectively.

A *body-part independent control* scheme is also introduced. As explained above, MotionDiffuse predicts the noise $\epsilon(\mathbf{x}_t, t, c)$ dimensioned in R^{FxD} (see section 4.5.1). This noise term determines the denoising direction of the whole body. Inspired by the application of the latent code interpolation, *noise interpolation* is proposed to separately control the different parts of the human body. Suppose there are n text descriptions $\{c_i\}$ for different body parts $\{s_i\}$. The aim is to calculate the noise term $\epsilon = \{\epsilon_i^{joint}\}, i \in [1, m]$ where ϵ_i^{joint} represents the noise term for the $i - th$ body part and m denotes the number of partition. $\epsilon_i^{joint} = \epsilon_\partial(\mathbf{x}, t, c_i) \in \mathrm{R}^{FxD}$ is calculated first and the total ϵ^{part} is calculated as $\epsilon^{part} = \sum_{i=1}^{m} \epsilon_i^{part} \cdot M_i$ where $M_i\{0, 1\} \in \mathrm{R}^D$ is a binary vector to show which body part should we focus on.

Finally, the number of diffusion steps for the diffusion model's training procedure is set to 1000 , and the variances β are linear from 0.0001 to 0.02. An Adam optimizer is used to train the model with a learning rate of 0.0002, while the total batch size is 1024.

4.6 Motion Latent Diffusion (MLD)

The **MLD**[12] method focuses exclusively in conditional human motion generation, crafting believable sequences of human motion based on diverse conditional inputs, such as action categories or textual descriptions. In general, a robust VAE aimed to extract a compact and meaningful latent representation for a human motion sequence comprises the cornerstone of MLD. Instead of directly linking the raw motion data to the conditional inputs using a diffusion model, a diffusion process within the motion latent space is executed. MLD generates lifelike motion sequences that align with the specified conditional inputs while significantly reducing computational complexity during both training and inference phases compared to other state-of-the-art methods.

The main issue MLD tried to address is that it is challenging to learn a probabilistic mapping function from the textural descriptors to the motion sequences since the distributions between the natural language descriptors and motion sequences are quite different. Modern techniques intend to encounter this problem by either using VAEs for motion and text encoding into a joint feature space or using a conditional diffusion model for human motion synthesis to learn a more powerful probabilistic mapping from the textual descriptors to human motion sequences and improve the synthesized fidelity and diversity. In the former case, since the distributions of natural languages and motion sequences are highly different, forcibly aligning these two simple Gaussian distributions into a compatible distribution might result in misalignments, while, in the latter case, the raw motion sequences are partly time-axis redundant and diffusion models in raw sequential data introduce computational overhead in both the training and inference stages. Besides, due to potential noise present in raw motion data captured by motion capture systems, powerful diffusion models could inadvertently be overfit to probabilistic mappings from the conditional inputs to noisy motion sequences, resulting in artifacts in the generated output. Thus, the hybrid approach suggested in MLD combines the advantages of the latent space-based and the conditional diffusion-based to overcome shortcomings of previous methods.

4.6.1 Motion Representation & Operational Modes

For the MLD settings of interest, two motion representations types are employed:

- the classical **SMPL-based** motion parameters, which is mostly used in motion capture. This method entails the calculation of motion/pose parameters ∂ and shape parameters β. $\partial \in \mathrm{R}^{3X23+3}$ is rotation vectors for 23 joints and a root, and β are the weights for linear blended shapes as further elaborated in 3.5.2. This representation is especially popular in marker-less motion capture. By involving the global translation r, the representation is formulated as:

$$x^i = \{r, \partial, \beta\} \tag{4.17}$$

- the **HumanML3D**(see more details in paragraph 3.3.2) representation, a redundant hand-crafted motion feature with a combination of joints features mostly employed in character animation. This type of representation is prevalently prefered over others owing to the fact it permits to generation methods to mitigate the foot-sliding issue through the binary foot contact features it entails.

- the **MMM representation format**(see further details in paragraph 3.3.1), which is conceptually attached to the *KIT dataset*

Operational modes in MLD are pretty straightforward and can be summarized in the following bullets:

- *Unconditional Generation*

- *Text-To-Motion Generation*

- *Action-To-Motion Generation*

4.6.2 Network Architecture, Sampling & Optimization

A crucial component of MLD architecture is concerned with compressing and reconstructing human motion for the learning of diffusion models. They prefer VAEs over GANs for this purpose since the latter are more difficult to train. After the model has learnt a representative and low-dimensional motion latent space for diverse human motion sequences, a diffusion process acts on this representation and consequently arrives at a motion latent-based diffusion model for conditional human motion synthesis. The goal of MLD is to generate a human motion $\hat{x}^{1:L} = \{\hat{x}^i\}_{i=1}^{L}$ in a non-deterministic way, where L denotes the frames' number. Motion sequence $x^{1:L}$ is encoded into a latent $z = \mathcal{E}(x^{1:L})$ and then, z, is decoded into a motion sequence using a motion decoder $\mathcal{D}$, i.e. $\hat{x}^{1:L} = \mathcal{D}(z) = \mathcal{D}(\mathcal{E}(x^{1:L}))$.

The motion VAE $\mathcal{V}$, based on a transformer-based architecture, consists of a transformer encoder $\mathcal{E}$ and a transformer decoder $\mathcal{D}$. The motion VAE $\mathcal{V} = \{\mathcal{E}, \mathcal{D}\}$ is trained on the motion $x^{1:L}$ reconstruction through optimizing the *MSE* loss and the *Kullback-Leibler (KL)* loss. The encoder targets at producing a representative, low-dimensional latent space with high informative density, and the decoder at reconstructing the latent space feature vectors into motion sequences.

In further detail, the motion encoder $\mathcal{E}$ receives as input learnable distribution tokens and frame-wise motion features $x^{1:L}$ of arbitrary length L. The embedded distribution tokens are employed as Gaussian distribution parameters μ and σ of the motion latent space $\mathcal{Z}$ to reparameterize latent variables $z \in \mathrm{R}^{nxd}$. The motion decoder $\mathcal{D}$ relies on the architecture of the transformer decoder with cross attention mechanism, which takes the L number of zero motion tokens as queries nad generates a motion sequence $x^{1:L}$ consisted of L frames. To further enhance the latent representation, long skip-connections are attached to the transformer-based encoder $\mathcal{E}$ and decoder $\mathcal{D}$ as visualized in figure 4.15. The impact of the latent's space dimensions on the quality of the generated motion sequences is significant and was found to be optimal for $n = 7$ and $d = 256$.

Diffusion models in the human motion synthesis paradigm are trained with a transformer-based denoiser $\epsilon_\theta(\mathbf{x}_t, t)$ by annealing the random noise to motion $\{\hat{\mathbf{x}}_t^{1:N}\}_{t=1}^{T}$ iteratively. The diffusion on latent space is modeled as a Markov noising process in the traditional way: $q(z_t|z_{t-1}) = N(\sqrt{a_t}z_{t-1}, (1-a_t)\mathrm{I})$. Let $\{z_t\}_{t=0}^{T}$ be the noising sequence and $z_{t-1} = \epsilon_\theta(z_t, t)$ for the $t-th$ denoising step. For the most generic case of unconditional generation, the loss function is formulated as:

$$L_{MLD} = \mathrm{E}_{\epsilon,t}[||\epsilon - \epsilon_\theta(z_t, t)||_2^2] \tag{4.18}$$

where $\epsilon \sim \mathcal{N}(0, 1)$ and $z_0 = \mathcal{E}(x^{1:L})$. During the training of ϵ_θ, the encoder $\mathcal{E}$ is frozen to compress motion into z_0. The samples of the diffusion forward process are from the latent distribution $p(z_0)$. During the diffusion reverse stage, ϵ_θ first predicts $\hat{z}_0$ with T iterative denoising steps and then $\mathcal{D}$ decodes $\hat{z}_0$ to motion results with one forward pass.

The MLD model is also capable of conditional motion generation $\mathcal{G}(c)$ by applying the conditional generation $p(z|c)$ with c standing for the textual condition. $\mathcal{G}(c)$ is implemented with a conditional denoiser $\epsilon_\theta(z_t, t, c)$ which can share a common motion VAE model. The text embedding is provided by a denoiser $\tau_\theta(c) \in \mathrm{R}^{mxd}$ thus the denoiser

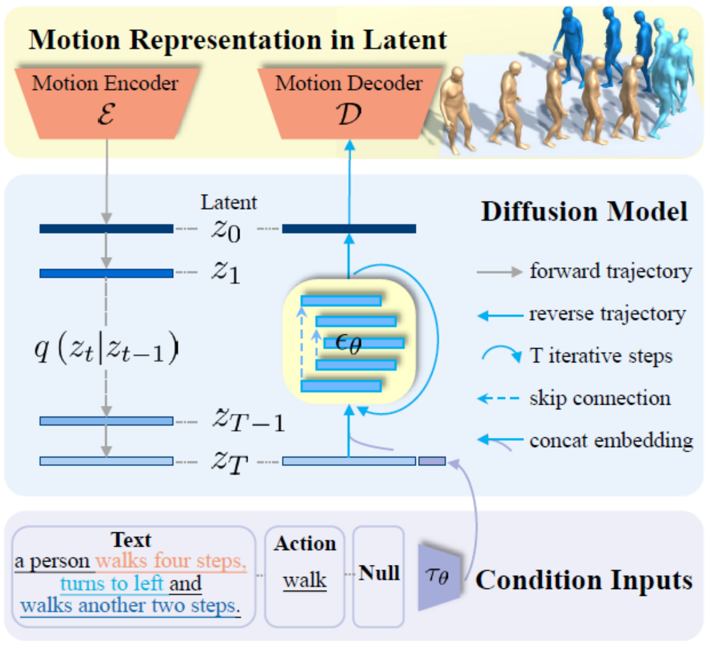


Figure 4.15. *MLD method overview. Source:[12]*

ϵ_∂ is expressed as $\epsilon_\partial(z_t, t, \tau_\partial)$. CLIP3.6 text encoder is employed to map text prompts to embeddings $\tau_\partial^w(w^{1:N}) \in R^{1xd}$. The conditional generation loss function is thus formulated as:

$$L_{MLD} = E_{\epsilon,t}[||\epsilon - \epsilon_\partial(z_t, t, \tau_\partial(c))||_2^2] \tag{4.19}$$

The denoiser ϵ_∂ is learned using classifier-free diffusion guidance 2.4.2, practically reflecting a trade-off to boost sample quality by reducing diversity in conditional diffusion models. Specifically, it learns both the conditioned and the unconditioned distribution with 10% dropout of the labeled samples and a linear combination of the two models is employed to calculate noise:

$$\epsilon_\partial^s(z_t, t, c) = s\epsilon_\partial(z_t, t, c) + (1 - s)\epsilon_\partial^s(z_t, t, \emptyset) \tag{4.20}$$

Following a series of rigorous ablation studies aimed at identifying the optimal value, the parameter s was ultimately established at 7.5.

Regarding the training hyper-parameters, models are trained with the **AdamW optimizer** using a fixed learning rate of 10^{-4}. The batch size is set to 128 during the VAE training stage and 64 during the diffusion training stage separately. Each model was trained for 6000 epochs during VAE stage and 3000 epochs during diffusion stage. The number of diffusion steps is set to 1000 during training and the variances β_t are scaled linearly from $8.5x10^{-4}$ to 0.012.

The overall training procedure followed is visualized in a delicate manner in figure 4.15. It comprises a two-stage process: $\mathcal{V}$ is first learnt for motion representations in latent space and, then, a conditioned denoiser ϵ_∂ is learnt from the diffusion process $q(z_t|z_{t-1})$. During inference, the latent diffusion models predict the latent $\hat{z}_0$ from condition inputs τ_∂ and motion decoder $\mathcal{D}$ decomposes it to coherent human motion.

Chapter 5

Methods' Comparative Evaluation

5.1 Introduction

The exhibition of the results will delve into the foundational aspects encountered in human motion synthesis methods as documented in existing literature, focusing primarily on *Text-to-Motion Generation* and (root) *Trajectory Control*. While Trajectory Control may encompass adjustments to other joints within the human skeleton or even combinations thereof, as elaborated upon in preceding chapters, the control of the root trajectory—specifically the pelvis joint—emerges as particularly intriguing due to its ability to facilitate the navigation of a human avatar within a virtual $3D$ environment according to our specified requirements.

Moreover, the assessment will extend beyond the mere demonstration of metrics showcasing the superiority of one method over another in quantitative terms. It will encompass a qualitative evaluation of crucial elements within the generated motion. This evaluation will explore how faithfully a specific method translates the input text prompt into the resulting motion, shedding light on which types of text prompts are most rigorously adhered to. Furthermore, the assessment will address the visual appeal of the generated motion, considering factors such as fluidity, naturalness, and overall aesthetic quality, as well the computational efficiency, especially during **inference**. By adopting a holistic approach that incorporates both quantitative metrics and qualitative analysis, this examination aims to provide a comprehensive understanding of the strengths and limitations of each method, thus guiding future research and development efforts in the field of human motion synthesis as well as underlining the impact of diffusion models on the human motion synthesis paradigm, the power of which continues to be harnessed paving the way for transformative discoveries and breakthroughs.

Before delving into the comparison of results, it is imperative to provide a comprehensive explanation of the motion quality metrics outlined in the subsequent sections. This involves elucidating their mathematical formulations as well as their intuitive interpretations, offering readers a clear understanding of their significance and relevance in evaluating motion quality.

5.1.1 Quality Metrics

The quality metrics can be summarized in 4 different categories:

1. **Motion Quality**: **Frechet Inception Distance (FID)** is the principal metric to evaluate the distribution similarity between generated and real motions' distribution, calculated with the suitable feature extractor for each dataset as described in [40], [83].

2. **Generation Diversity**: **Diversity (DIV)** and **MultiModality (MM)** are employed to measure the motion variance across the whole set and the generated motion diversity within each text input separately. To evaluate **Diversity**, all generated motions are randomly sampled to two subsets of the same size X_d with motion feature vectors $\{x_1, x_2, ..., x_{\mathrm{X}_d}\}$ and $\{x_1^{'}, x_2^{'}, ..., x_{\mathrm{X}_d}^{'}\}$ respectively. In this case, diversity metric is formulated as:
$$DIV = \frac{1}{\mathrm{X}_d} \sum_{i=1}^{\mathrm{X}_d} ||x_i - x_i'|| \tag{5.1}$$
To evaluate MultiModality, a set of text descriptions with size J_m is randomly sampled from all descriptions. Then two subsets of the same size X_m are randomly sampled from all motions generated by $j - th$ text descriptions, with motion feature vectors $\{x_{j,1}, x_{j,2}, ..., x_{j,\mathrm{X}_m}\}$ and $\{x_{j,1}^{'}, x_{j,2}^{'}, ..., x_{j,\mathrm{X}_m}^{'}\}$ respectively. The multimodality is calculated as:
$$MM = \frac{1}{J_m \times \mathrm{X}_m} \sum_{j=1}^{\mathrm{J}_m} \sum_{i=1}^{\mathrm{X}_m} ||x_{j,i} - x_{j,i}^{'}|| \tag{5.2}$$

3. **Condition Matching**: For the text-to-motion task, [14] provides motion/text feature extractors to produce geometrically closed features for matched text-motion pairs, and vice versa. Under this feature space, motion-retrieval precision (**R-Precision**) first mixes generated motion with 31 mismatched motions and then calculates the text-motion top-1/2/3 matching accuracy, and Multi-modal Distance (MM Dist) that calculates the distance between generated motions and text.

4. **Time costs**: To evaluate the computing efficiency of diffusion models, especially the inference efficiency, Average Inference Time per Sentence (AITS) measured in seconds is qualified over other approaches. In our case, we calculate AITS on the test set of HumanML3D dataset, set the batch size to one, and ignore the time cost for model and dataset loading parts.

5.2 Text-to-Motion Generation

5.2.1 Quantitative Evaluation

The precision metric serves as a vital tool in evaluating the efficacy of human motion generation models. It goes beyond mere accuracy, capturing the model's capacity to produce not only correct but also plausible and contextually relevant motions, even when the top prediction doesn't align perfectly. It's evident that MotionDiffuse excels over the other two methods in terms of precision across all three paradigms (Top1, Top2, Top3) - surpassing MDM by a considerable margin and MLD by a narrower one, yet

Table 5.1. *Text-To-Motion generation evaluation on HumanML3D dataset (Precision)*

Methods	**R-precision (increasing)**		
	Top 1	**Top 2**	**Top 3**
Real Motions	0.511(0.03)	0.703(0.03)	0.797(0.02)
Motion Diffuse	0.491(0.01)	0.681(0.01)	0.782(0.01)
MDM	0.32(0.05)	0.498(0.04)	0.611(0.07)
MLD	0.481(0.03)	0.673(0.03)	0.772(0.02)

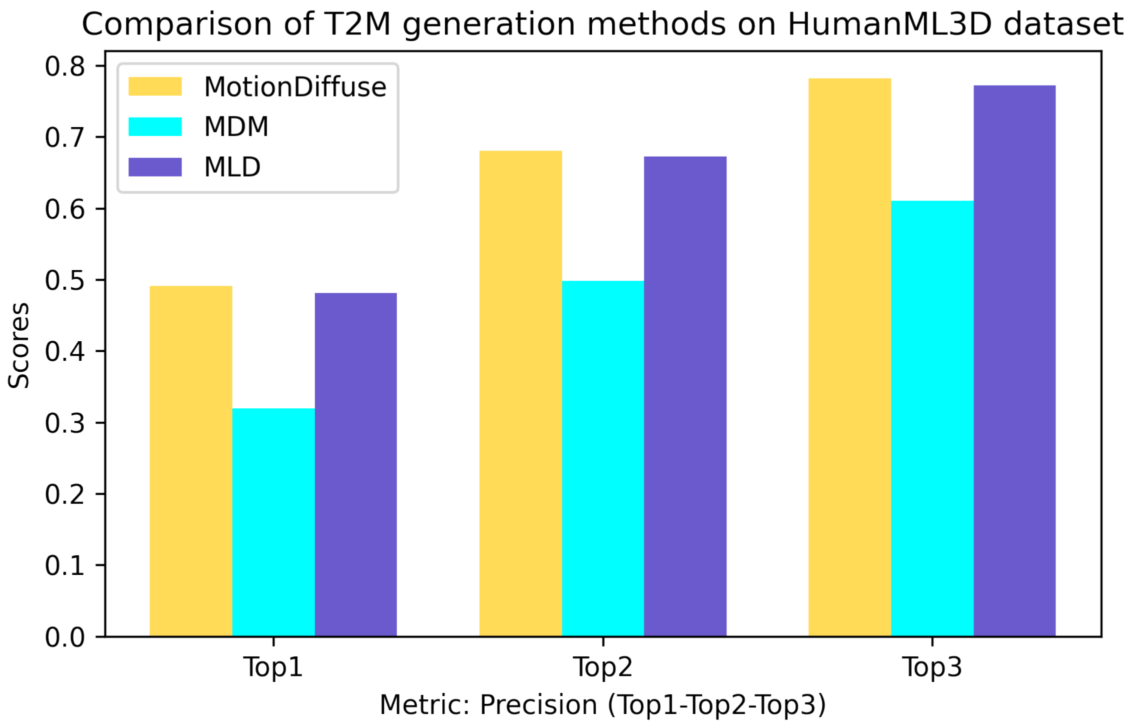


Figure 5.1. *Comparative results on HumanML3D dataset in terms of precision*

still demonstrating superiority. This significant outcome holds true for both datasets, as vividly illustrated in figures 5.1 and 5.2. Furthermore, tables 5.1 and 5.3, corresponding to the HumanML3D and KIT datasets respectively, provide additional insights, showing that MotionDiffuse and MLD achieve precision metrics closely resembling those observed in actual recorded motions, which is impressive at the least. These results underscore their ability to generate motion sequences aligned with the envisioned actions described in the text prompts provided as supervision to the models.

In the context of Condition Matching as defined in the former paragraph (5.1.1), we also need to address MMDist metric, most clearly reflecting the extent to which the generation process obeys to the textual description provided as input. As depicted in Figures 5.3 and 5.4, MotionDiffuse exhibits a slight edge over MLD, with both methods competing closely. Notably, both MotionDiffuse and MLD outperform MDM by a significant margin, albeit with a narrower gap in the HumanML3D dataset compared to the KIT dataset. Remarkably, these two superior methods, MotionDiffuse and MLD, trend towards values aligning more closely with real motions for the specific metric under consideration. As a

Table 5.2. *Text-To-Motion generation evaluation on HumanML3D dataset (Rest of the metrics)*

Methods	**FID**	**MM Dist**	**Diversity**	**MModality**
Real	0.002(0.00)	2.974(0.08)	9.503(0.65)	–
Motion Diffuse	0.630(0.01)	3.113(0.01)	9.410(0.49)	1.553(0.42)
MDM	0.544(0.44)	5.566(0.27)	9.559(0.86)	2.799(0.72)
MLD	0.473(0.13)	3.196(0.10)	9.724(0.82)	2.413(0.79)

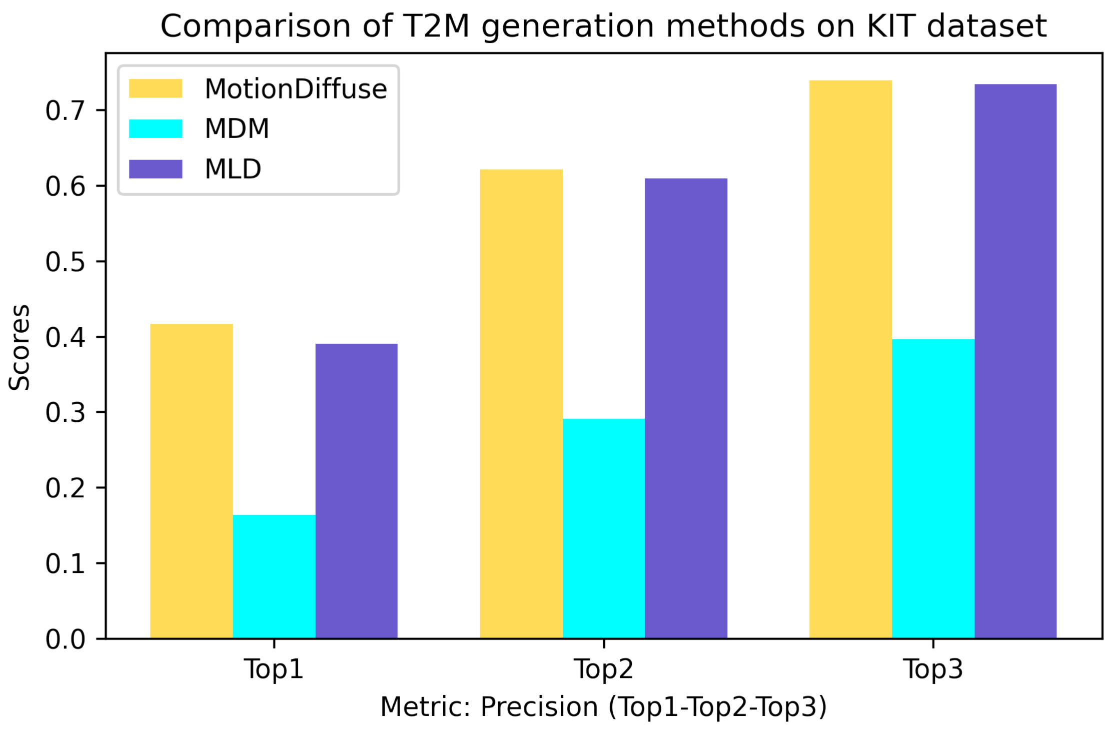


Figure 5.2. *Comparative results on KIT dataset in terms of precision*

result, MDM appears to significantly fall short in faithfully reflecting the text prompt onto the generated motion when compared to the other methods.

The Fréchet Inception Distance (FID) emerges as an equally vital metric in the evaluation of human motion synthesis, exhibiting a strong correlation with human judgment regarding motion quality. In practical terms, FID offers a more nuanced evaluation of generated samples, transcending simple visual inspection. It qualitatively ascends in a descending manner - lower values signify a closer alignment of generated motion distributions with the desired outcome. It is widely regarded as the metric that most aptly reflects the coherence and quality of motion synthesis.

In the HumanML3D dataset, MLD asserts its dominance over the other two methods in terms of FID, albeit by a narrow margin. However, in the KIT dataset, it significantly outperforms MotionDiffuse, with MDM closely trailing behind. These distinctions are vividly depicted in Figures 5.3 and 5.4. Upon examination of Tables 5.2 and 5.4, it becomes apparent that despite the advancements represented by state-of-the-art methods, there still exists a shortfall in approaching real motions in terms of FID. This observation

Table 5.3. *Text-To-Motion generation evaluation on KIT dataset (Precision)*

Methods	**R-precision (increasing)**		
	Top 1	**Top 2**	**Top 3**
Real Motions	0.424(0.05)	0.649(0.06)	0.779(0.06)
Motion Diffuse	0.417(0.04)	0.621(0.04)	0.739(0.04)
MDM	0.164(0.04)	0.291(0.04)	0.396(0.04)
MLD	0.390(0.08)	0.609(0.08)	0.734(0.07)

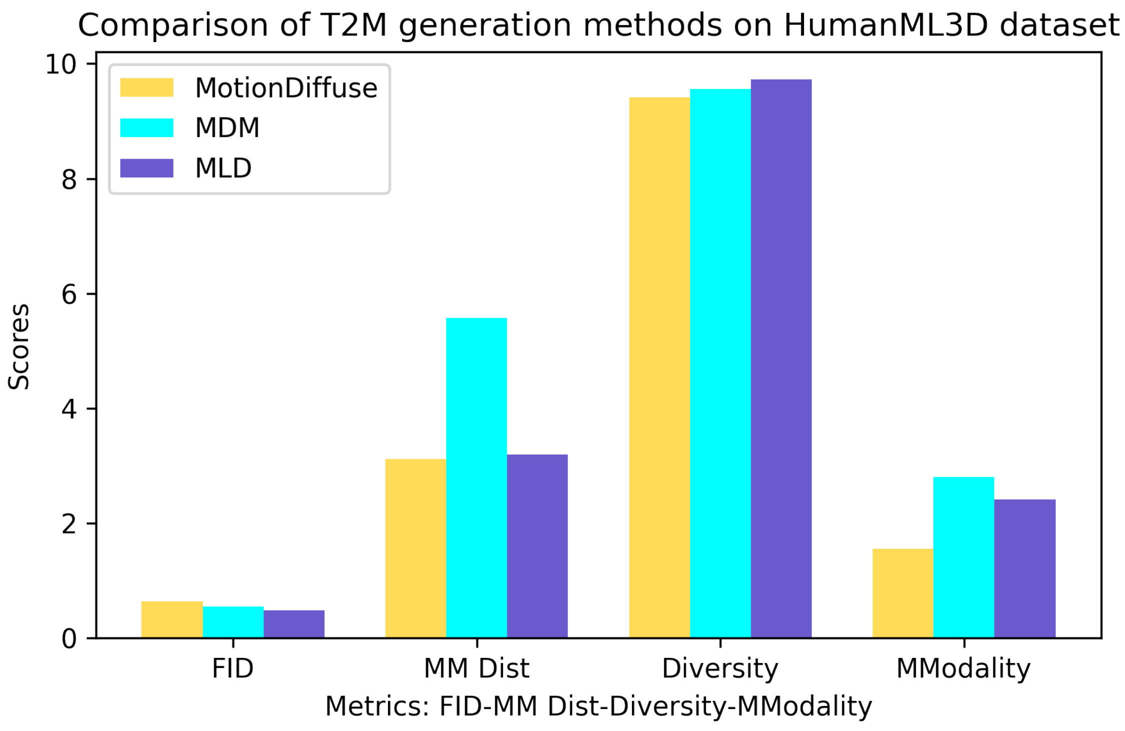


Figure 5.3. *Comparative results on HumanML3D dataset in terms of FID, DIV, MM Dist, MModality*

aligns with the inherent nature of the metric, which demands a level of fidelity that is challenging to achieve.

Diversity serves as a fundamental metric in the assessment of human motion synthesis, offering insights into the extent to which generated motions span the input feature space. Unlike some metrics, Diversity does not follow a linear trend; rather, the superiority of a method is determined by its ability to achieve diversity values closer to those of real motions. For a comprehensive understanding of the results, it is recommended to focus on Tables 5.2 and 5.3. In the HumanML3D dataset, MDM closely approaches the diversity value of real motions, with MotionDiffuse following closely behind and MLD trailing in last place. However, in the KIT dataset, MotionDiffuse outperforms MDM, while MLD consistently falls short.

The MultiModality metric measures the extent to which a generative model is capable of producing a wide range of plausible motions, capturing different styles, variations, and nuances in human movement. It evaluates whether the generated motions exhibit

Table 5.4. *Text-To-Motion generation evaluation on KIT dataset (Rest of the metrics)*

Methods	**FID**	**MM Dist**	**Diversity**	**MModality**
Real	0.031(0.04)	2.788(0.12)	11.08(0.97)	–
Motion Diffuse	1.954(0.62)	2.958(0.05)	11.10(0.143)	0.730(0.13)
MDM	0.497(0.21)	9.191(0.22)	10.85(0.109)	1.907(0.214)
MLD	0.404(0.27)	3.204(0.27)	10.80(0.117)	2.192(0.71)

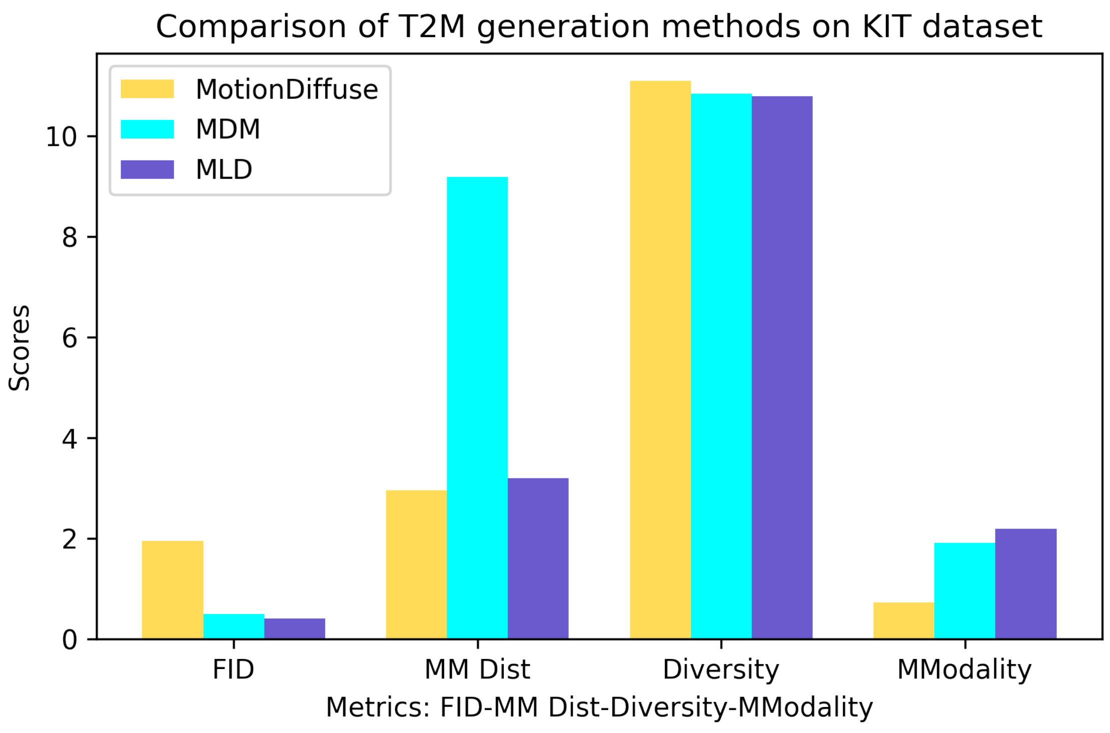


Figure 5.4. *Comparative results on KIT dataset in terms of FID, DIV, MM Dist, MModality*

multiple modes or styles, indicating diversity and richness in the generated dataset. A high MultiModality score suggests that the generative model is successful in capturing the variability and richness of human motion, producing diverse and realistic samples across different styles and variations.

Upon examining Figures 5.3 and 5.4, distinct trends emerge regarding MModality performance across datasets. In the KIT dataset, MLD emerges as the top performer in MModality, whereas MDM asserts its dominance in the HumanML3D dataset. These two methods exhibit superior performance in terms of MModality. Conversely, MotionDiffuse encounters challenges with MModality in both datasets, particularly evident in the KIT dataset where its performance appears comparatively weaker. This observation underscores the nuanced differences in performance across different evaluation criteria and datasets, highlighting the need for comprehensive analysis in human motion synthesis evaluation.

Finally, it's essential to address the crucial aspect of execution time, particularly regarding the computational resources required by each method. For a comprehensive overview, refer to Table 5.5, which outlines the computational efficiency of each method.

Notably, the evaluation was conducted on the HumanML3D dataset test set. A notable limitation becomes evident for classical diffusion models, namely MDM and MotionDiffuse, as MLD demonstrates a remarkable advantage in terms of speed. In fact, MLD achieves a speed that is two orders of magnitude faster than its counterparts, underscoring its efficiency and practical feasibility for real-world applications. This advantage can be attributed in part to the utilization of DDIM sampling in MLD's best model version. By leveraging this approach, MLD maintains a high level of motion generation quality while simultaneously minimizing inference time costs to an extremely low level. Conversely, MDM and MotionDiffuse offer flexibility in employing DDIM, allowing for a reduction in the number of noise steps taken during inference and, consequently, computational inference time. However, this trade-off comes at the expense of generation quality, as evidenced in Table 5.5. All evaluation experiments were executed on a NVIDIA **GeForce GTX 1660Ti**.

Table 5.5. *Comparison of the inference time costs on text-to-motion.*

Methods	**Inference time per Prompt (in secs)**
Motion Diffuse (DDPM)	29.4
Motion Diffuse (DDIM-100)	0.307
MDM (DDPM)	49.5
MDM (DDIM-100)	0.497
MLD	0.422

Before drawing conclusions from the quantitative evaluation within the text-to-motion generation paradigm, it's crucial to clarify the significance of the numbers in parentheses in Tables 5.1, 5.3, 5.2, and 5.4. These numbers represent the range of the 95% confidence interval, obtained by iteratively running the evaluation scripts 50 times. This approach ensures robustness and reliability in the assessment of model performance, accounting for potential variability in results due to stochastic factors or dataset characteristics.

5.2.2 Qualitative Evaluation

Before delving into the specifics of our qualitative evaluation, it is important to make certain clarifications. The textual prompts employed to perform a qualitative assessment and, consequently, a comparison were chosen using the following criteria:

- Prompts suggested in the papers for all three methods were utilized to highlight the advantages of the methods compared to the other methods, employing the same number of text prompts from each paper to guarantee equity. This approach was chosen for an additional reason: to observe how the other methods respond to text prompts where another method has excelled, thus evaluating their generalization capability.

- Prompts of our own imagination were employed to assess aspects such as orientation perception, style transfer, or the number of repetitions of an action needed to be executed during a generated motion.

- the evaluation will mostly revolve around the inconsistencies of the models as it is evident and undoubtable that all suggested methods comprise impressive scientific achievements. By building upon existing research and addressing the identified inconsistencies, we can further elevate the state-of-the-art in animation and enhance the overall quality of motion synthesis in various applications.

We have divided our evaluation into two distinct sections to provide a clear and organized structure: one focusing on single-action textual prompts and the other on textual descriptions implying multiple actions. For the complete results showcasing of our assessment, please refer to the following **link**.

Single-action setting

To begin with, the effectiveness of the methods under examination in responding to specific gestures and actions has been a subject of scrutiny. Among these methods, MDM displays limitations in recognizing certain upper limb gestures, such as waving (e.g. *'someone is waving to a person'*) while MotionDiffuse and MLD perform more satisfactorily in this regard. Notably, MDM stands out as the sole method adept at accurately distinguishing between left and right hands, a capability where MLD and MotionDiffuse falter, leading to erroneous identifications. However, when it comes to distinguishing left from right lower limbs, all methods demonstrate proficiency, with MDM maintaining superior performance by avoiding awkward movements often exhibited by MLD and MotionDiffuse, as shown in the output of text prompt *'a person kicks with their left/right leg'*.

An additional observation to note is the varying responsiveness of motion modeling techniques to directional shifts. While MotionDiffuse exhibits satisfactory performance in this regard, both MDM and MLD appear less receptive. For instance, when prompted with actions involving directional changes, such as *"a person paces from left to right"*, MotionDiffuse demonstrates a more accurate interpretation and execution of the intended movement.

Furthermore, discrepancies emerge in the ability of these methods to interpret the intended trajectory of motion. While MLD struggles with consistent responsiveness, often misinterpreting instructions such as walking in a specific direction (e.g. is asked to walk in counter-clockwise fashion but does so in clockwise fashion), MDM and Motion reliably perceive and execute the requested movement direction. Notably, MotionDiffuse excels in adjusting to varying gaiting speeds (see the outcome of prompt *'the person is walking out a medium speed'*), providing a more nuanced response compared to its counterparts, which exhibit shortcomings in this aspect. Similarly, MotionDiffuse demonstrates superior adaptability to queries containing the number of steps required during gaiting, as evident in the produced motion of *'a person walking moves forward taking 5 confident strides'*, whereas MLD and MDM tend to produce repetitive motions lacking in specificity.

In scenarios involving ground contact, such as crawling (*'a person is crawling on the floor'*), MLD exhibits noticeable deficiencies, manifesting in undesirable foot sliding issues, while MDM and MotionDiffuse exhibit more favorable responses. Interestingly, all methods showcase proficiency in generating realistic boxing or fighting sequences (e.g.

'the person is exchanging punches with his opponent'), indicating a shared competence in capturing dynamic movements within this context.

Furthermore, MLD's output occasionally resembles awkward or unclear dance movements, which can be deemed unsatisfactory for certain applications. In contrast, MDM performs more reliably in this aspect, demonstrating superior proficiency in interpreting and replicating dance-related actions. Finally, MLD tends to exhibit noticeable foot sliding artifacts when the text prompt suggests an increase in the speed of movement, such as motions entailing running.

Despite these comparative analyses of individual actions, the examination transitions to evaluating the models' efficacy in comprehending and seamlessly transitioning between sequences of actions as described in textual prompts. This shift prompts a deeper exploration into how each method conceptualizes sequential actions and executes smooth transitions, an essential aspect for deriving conclusive insights into their overall performance and applicability. Moving forward, it is crucial to transition from analyzing single-action prompts to evaluating how well the models handle textual descriptions implying a sequence of actions. Assessing their ability to conceptualize and implement smooth transitions between actions is paramount in determining their effectiveness in generating cohesive motion sequences.

Multi-action setting

In evaluating the performance of motion generation methods across in the multi-action setting, several key observations emerge. MLD and MotionDiffuse exhibit commendable responsiveness when actions involve sitting on objects, or even standing back up, whereas MDM appears limited to understanding sitting on the ground, as evidenced by its response to complex prompts such as *'a person walks up to a backwards chair and sits down on it with legs outstreched, then stands back up'*. Moreover, MotionDiffuse stands out as the sole method capable of executing motions involving object manipulation without introducing disruptive artifacts.

MLD demonstrates notable proficiency in faithfully reproducing sequences of actions with abrupt transitions or complex combinations, as proven by the outcome of the *('person jumps forward and turns left in mid air')* prompt, a task where MDM and MotionDiffuse falter not even forcing the avatar to turn after or during jumping. Furthermore, MLD excels in accurately rendering sequences comprising multiple short actions, a feat often marred by MDM's occasional total omission of actions, such as standing, and MotionDiffuse's loss of orientation, as illustrated by textual descriptions involving walking, sitting and returning to the starting point such as *'a person walks forward, turns, then sits, then stands and walks back'*.

While MotionDiffuse and MDM showcase superior orientation perception during motions involving shifts, especially when multi-directional shifts are queried(*'a man runs to the right, then runs to the left, then back to the middle'*), they occasionally deviate from the exact sequence outlined in the prompts. Similarly, all methods exhibit occasional inconsistencies in adhering to textual descriptions, with instances of actions being omitted or incorrectly executed, as happens in e.g. *'Looking around and then calling on a phone with right hand'*. MotionDiffuse presents serious shortcomings in differentiating between limbs, often executing actions with the wrong limb or both limbs simultaneously.

Additionally, in some instances, certain motion modeling techniques fail to accurately adhere to textual descriptions, raising concerns about their reliability in certain scenarios. Characteristic examples detected were the generated actions prompted as *'A person is pushed hard to the left and then recovers into a standing position'* and *'a person is running, then stops and bows'* which were not faithfully replicated by both MDM and MotionDiffuse, in the former case and MotionDiffuse and MLD in the latter. Moreover, there are cases where the focus of a method seems to prioritize one action over another, leading to the omission of certain implied actions. For instance, in the scenario where a person is described as crawling on the floor and then getting up on their feet, both MotionDiffuse and MDM may overlook one of the actions entirely, potentially compromising the overall fidelity of the animation.

Turning positive, MDM excels in capturing sports-related activities like golf or basketball, making it particularly relevant for animation systems. Conversely, MLD demonstrates superiority in handling sequences of multiple actions, especially those characterized by intricate textual prompts, showcasing its robustness in capturing nuanced action sequences.

It is worth acknowledging that the challenges faced by these motion modeling methods may not solely be attributed to their inherent limitations. Indeed, the duration of the animation clip provided could significantly impact the ability of the model to generate a coherent motion sequence aligned with the specific prompt. In many cases, the complexity of the action described in the prompt may exceed the constraints of the given time interval, making it difficult for the model to produce a seamless and accurate representation within those parameters. In such situations, the human in-the-loop may need to exercise intuition and make adjustments to ensure the final animation aligns more closely with the intended action.

In summary, while each motion synthesis technique exhibits strengths and weaknesses across different action scenarios, their performance nuances highlight the importance of considering specific task requirements and textual prompts when selecting the appropriate method for animation applications.

5.3 Trajectory Control

5.3.1 Quantitative Evaluation

As discussed in the corresponding chapter on text-to-motion generation assessment, the precision metric is crucial for evaluating the effectiveness of human motion generation models, as it considers not only accuracy but also the capacity to produce believable and contextually relevant motions, even when the top prediction isn't a perfect match. OmniControl demonstrates superior performance over the other two methods in Top-3 R-Precision, surpassing GMD by a narrow margin and PriorMDM by a wider one on the

Table 5.6. *Root trajectory control evaluation on the HumanML3D dataset (Quality metrics)*

	FID	**R-Precision (Top-3)**	**Diversity**	**Foot skating ratio**
Real Motions	0.002	0.797	9.503	0.000
PriorMDM	0.475	0.583	9.156	0.0897
GMD	0.576	0.665	9.206	0.1009
OmniControl	0.218	0.687	9.422	0.0547

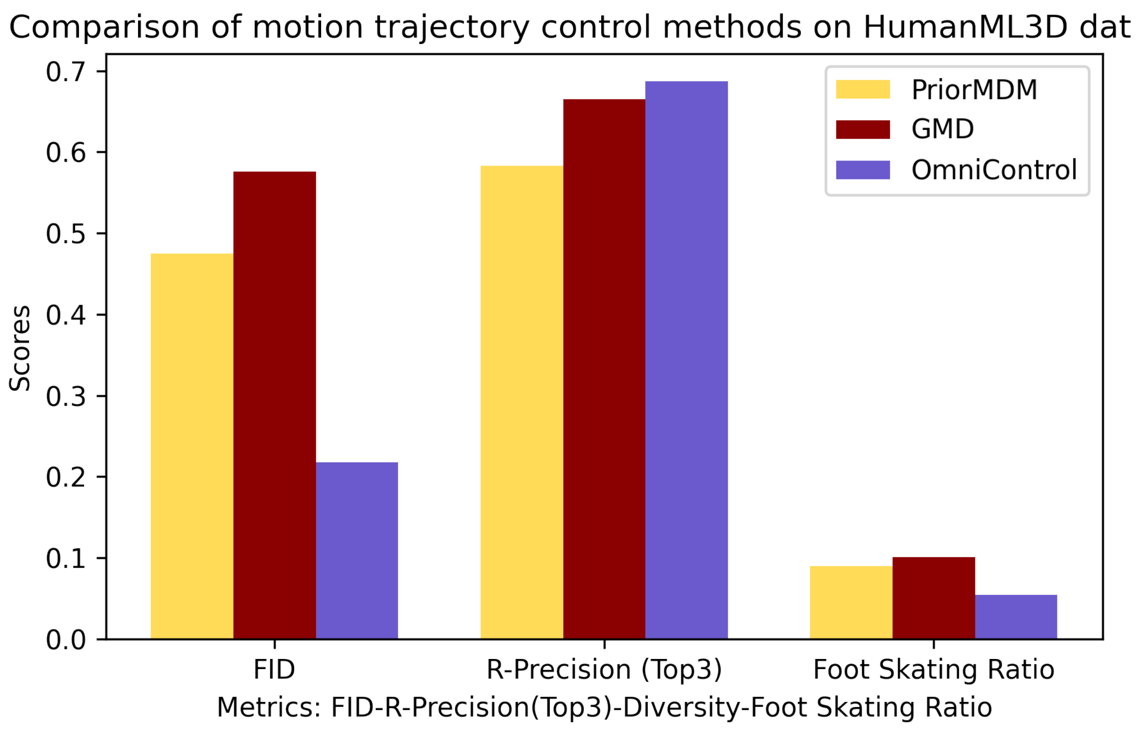


Figure 5.5. *Comparative results on HumanML3D dataset in terms of FID, R-Precision, Foot Skating Ratio*

HumanML3D dataset, as shown in figure 5.5. The situation differs slightly on the KIT dataset, where R-Precision values appear almost equal at first glance in figure 5.7, but a closer examination of table 5.7 reveals that the difference between OmniControl and the others is negligible, with GMD closely following. It's noteworthy that precision values for all methods are nearly halved on the KIT dataset, which is surprising, considering that in the former case, the values are comparable to those calculated on real motion, while in the latter, the difference is significant. Additionally, tables 5.1 and 5.3, corresponding to the HumanML3D and KIT datasets respectively, provide further insights, indicating that PriorMDM & OmniControl achieve precision metrics closely resembling those observed in real recorded motions, which is quite impressive. These results highlight their ability to generate motion sequences consistent with the intended actions described in the text prompts provided as guidance to the models. Particularly, the results from models trained on the HumanML3D dataset hold significant value as they demonstrate the models' ability to generate motion sequences aligned with textual descriptions while closely adhering to the prescribed trajectory.

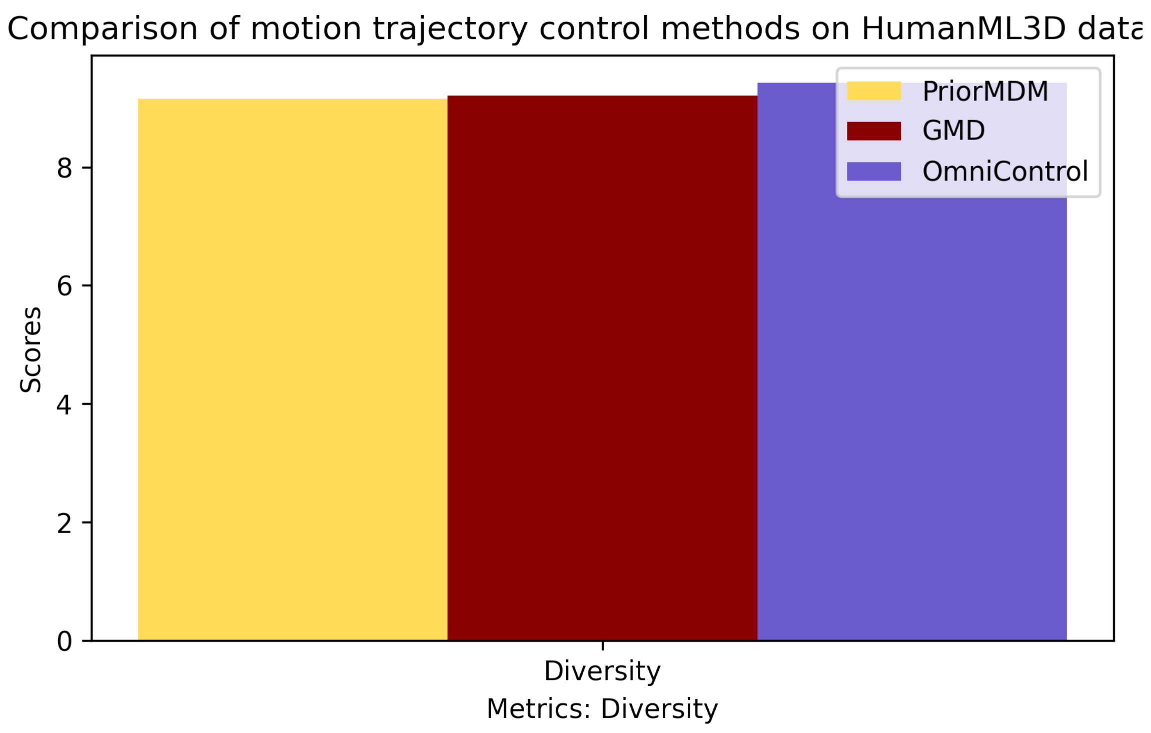


Figure 5.6. *Comparative results on HumanML3D dataset in terms of Diversity*

The Fréchet Inception Distance (FID) serves as a crucial metric in motion trajectory control, demonstrating a strong correlation with human perception of motion quality, namely on how human eye perceives the naturalness of the motion. FID provides a more nuanced assessment of generated samples beyond mere visual inspection. In the HumanML3D dataset, OmniControl outperforms the other methods significantly in terms of FID, with GMD showing a considerable gap and PriorMDM trailing slightly behind, as depicted in Figure 5.5. This trend persists in the KIT dataset, although in this case, PriorMDM follows closely behind, while GMD consistently falls short, as illustrated in Figure 5.7. Analysis of Tables 5.2 and 5.4 reveals that despite advancements in state-of-the-art methods, there remains a deficiency in replicating real motions in terms of FID. This observation aligns with the inherent demand of the FID metric for a level of fidelity that is challenging to achieve.

Table 5.7. *Root trajectory control evaluation on the KIT dataset (Quality metrics)*

	FID	R-Precision (Top-3)	Diversity
Real Motions	0.031	0.779	11.08
PriorMDM	0.851	0.397	10.518
GMD	1.565	0.382	9.664
OmniControl	0.702	0.397	10.927

For a comprehensive understanding of result diversity, it's advisable to focus on Tables 5.6 and 5.7, as supported by the diagrams in Figures 5.6 and 5.8, which indicate comparable performance among the methods in this aspect. On the HumanML3D dataset,

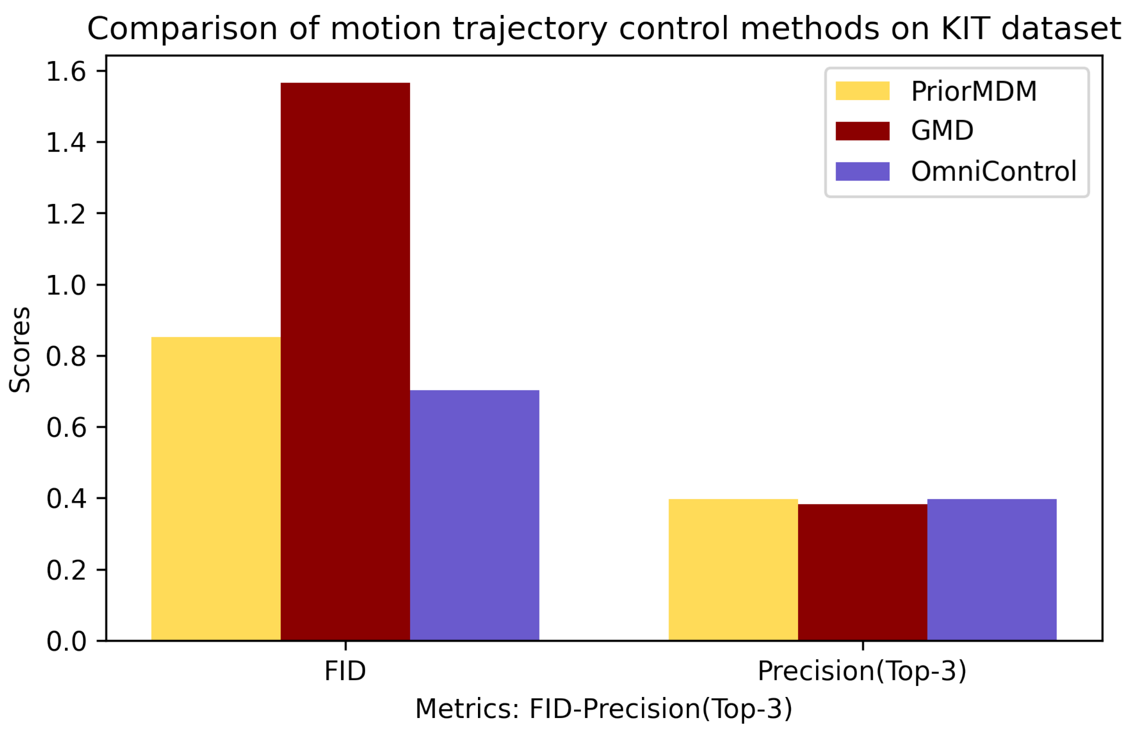


Figure 5.7. *Comparative results on KIT dataset in terms of FID, R-Precision, Foot Skating Ratio*

OmniControl emerges as superior, closely approximating the diversity of real motions, while GMD and PriorMDM lag significantly behind in their respective rankings. However, on the KIT dataset, PriorMDM secures second place, with GMD trailing in third. Nevertheless, all methods demonstrate convincing results overall.

In concluding our quantitative analysis regarding the quality of the generated movements, we focus on assessing the foot skating ratio, which serves as a metric for gauging the occurrence of foot sliding—an issue widely recognized as a significant challenge in human motion synthesis techniques. The evaluation of the foot skating ratio is exclusively conducted on the HumanML3D dataset due to the distinct feature structure present in the KIT dataset, which hinders the assessment of this particular aspect. To be succinct, it appears that OmniControl refrains from presenting the foot skating ratio metric, whereas PriorMDM exhibits a slightly inferior performance, with GMD demonstrating a marginally worse performance compared to PriorMDM, as illustrated in Figure 5.5.

Table 5.8. *Root trajectory control evaluation on the HumanML3D dataset (Accuracy metrics)*

	Trajectory Error	**Location error**	**Average Error**
Real Motions	0.00	0.00	0.00
PriorMDM	0.851	0.397	10.518
GMD	0.0931	0.0321	0.1439
OmniControl	0.0387	0.0096	0.0338

Moving forward, it is imperative to assess the methods based on their adherence to

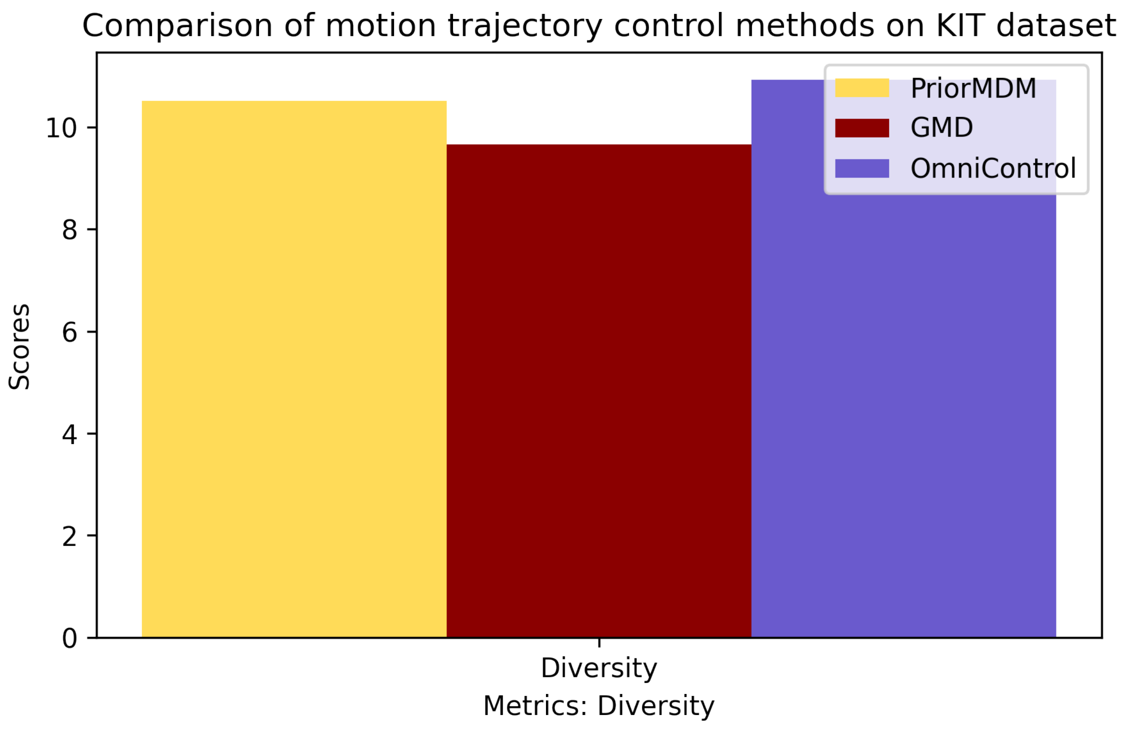


Figure 5.8. *Comparative results on KIT dataset in terms of Diversity*

the prescribed trajectory, a crucial factor for motion trajectory control. To this end, it is important to highlight what each of the three displayed metrics (*Trajectory error*, *Location error* and *Average error*) actually stand for:

- **Trajectory error** refers to the proportion of unsuccessful trajectories, which are characterized by any keyframe location error surpassing a predetermined threshold.
- **Location error**, on the other hand, represents the ratio of keyframe locations that fail to be reached within a specified distance threshold.
- **Average error** quantifies the average distance between the generated motion locations and the keyframe locations assessed at the keyframe motion steps.

The predefined threshold value is set to $50cm$. The error between real motions and the prescribed trajectory would be zero, as intuition suggests.

As illustrated in Figures 5.9 and 5.10, OmniControl exhibits remarkable superiority over the other two methods across all error metrics defining the performance of pelvis trajectory control, also known as root trajectory control, on both the HumanML3D and KIT datasets. However, it is noteworthy that despite OmniControl's consistent superiority, PriorMDM and GMD assume varying roles across datasets. Specifically, on the HumanML3D dataset, GMD demonstrates a noteworthy performance, approaching the level of OmniControl, while PriorMDM significantly lags behind the other two methods. Conversely, on the KIT dataset, PriorMDM attempts to mirror the performance of OmniControl, albeit not as closely, while GMD falls notably short of expectations.

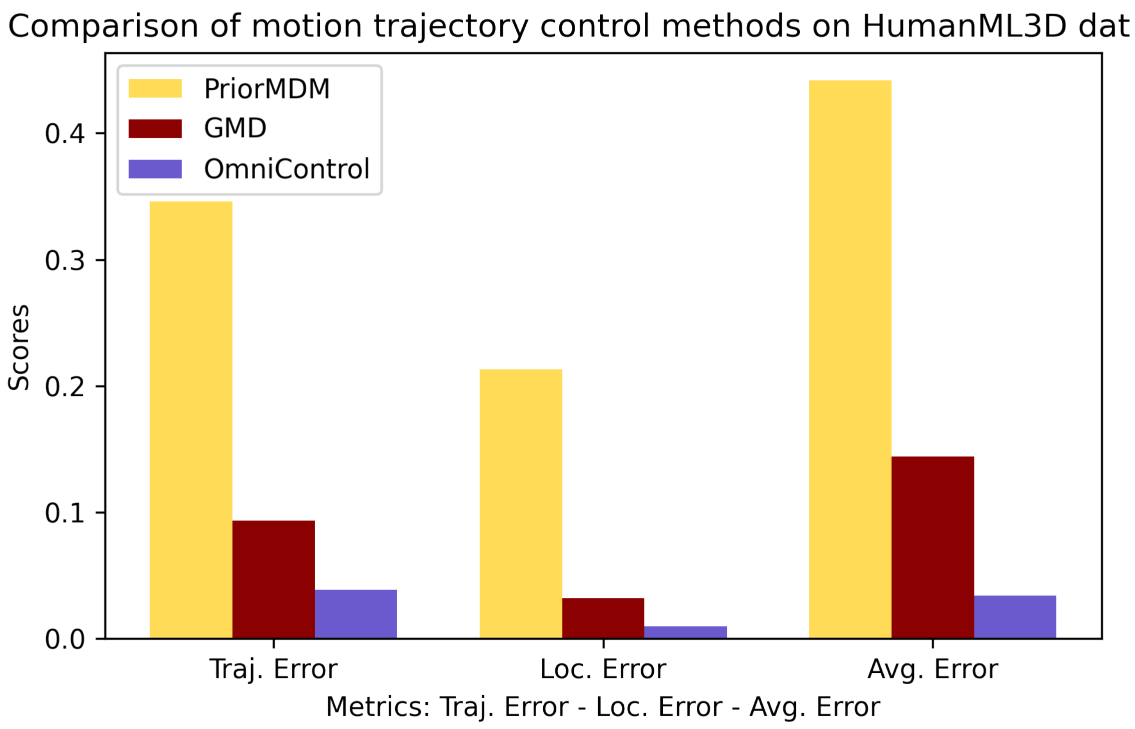


Figure 5.9. *Comparative results on HumanML3D dataset in terms of FID, DIV, MM Dist, MModality*

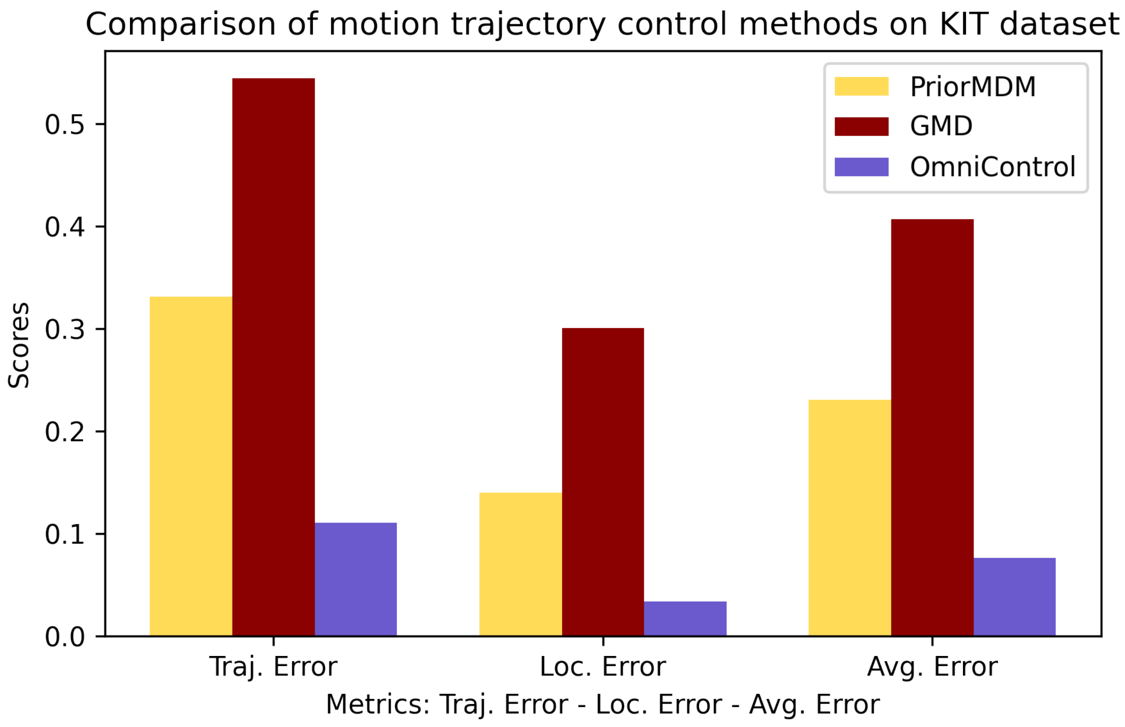


Figure 5.10. *Comparative results on HumanML3D dataset in terms of FID, DIV, MM Dist, MModality*

In our perspective, the metric that warrants the most consideration for conducting an effective assessment is the average error. This metric essentially quantifies the average deviation from the prescribed trajectory. While the other two metrics primarily focus on identifying instances of particularly poor outcomes, which may not fully characterize the model's overall performance, they are not to be underestimated in their importance as they aid us gain a comprehensive understanding of each model's performance and their ability to maintain trajectory fidelity consistently. Taking into account all metrics, with particular emphasis on the Average error, OmniControl emerges as the leading contender.

Table 5.9. *Root trajectory control evaluation on the KIT dataset (Accuracy metrics)*

	Trajectory Error	**Location error**	**Average Error**
Real Motions	0.00	0.00	0.00
PriorMDM	0.851	0.397	10.518
GMD	0.5443	0.3003	0.407
OmniControl	0.1105	0.0337	0.0759

To conclude our quantitative analysis thoroughly, it's essential to delve into the intricate interplay between computational resources and performance, a dynamic that greatly influences the holistic evaluation of each method. In this context, it's imperative to consider not only the efficacy of the methods but also the computational burden they impose. This allows for a nuanced understanding of the trade-off between performance and resource consumption, which is pivotal for making informed decisions.

A standout performer in this regard is PriorMDM, which exhibits a remarkable efficiency by requiring significantly less computational time compared to its counterparts. In fact, it outperforms both GMD and OmniControl by a substantial margin, demanding only a fraction of the computational resources needed by the other methods - GMD needs 3 times the computational of PriorMDM while OmniControl lasts even more. This disparity in computational load can be attributed to the foundational architecture of PriorMDM, which serves as the backbone for the subsequent methods. By leveraging this established framework and integrating additional modules judiciously, OmniControl and GMD extend the capabilities of PriorMDM. Predictably, this augmentation inevitably comes at the cost of increased computational overhead.

Table 5.10. *Comparison of the inference time costs on motion trajectory control.*

Methods	**Inference time per trajectory session (in secs)**
PriorMDM	76.7
OmniControl	231.3
GMD	211.6

This observation underscores the importance of considering not just raw performance metrics but also the computational efficiency of each method. Striking the right balance between performance and resource utilization is paramount, ensuring that the chosen method meets the requirements of the task at hand while maximizing efficiency and minimizing computational overhead.

5.3.2 Qualitative Evaluation

The qualitative analysis conducted to assess the effectiveness of three distinct methods—PriorMDM, GMD, and OmniControl—seeks to capture the nuanced visual experience of users when employing these methods. This evaluation particularly emphasizes the utilization of hand-crafted trajectories, contrasting with trajectories drawn from our available datasets, which were utilized in the quantitative evaluation showcased in the preceding section. Our approach was guided by the belief that a diverse range of prescribed motions should be explored, encompassing extreme scenarios such as sharp turns, in addition to conventional trajectories characterized by smooth curves. The evaluation revolves around two primary dimensions:

- **Accuracy of Generated Motion:**This dimension investigates whether the motion derived solely from a pelvis trajectory aligns with the prescribed signal and to what degree.
- **Aesthetic Coherence and Naturalness:**This aspect examines how cohesive and visually appealing the generated motion appears, with a focus on its naturalness while attempting to stay faithful to the input control signal.

Consequently, we have structured our evaluation into two distinct sections: one dedicated to assessing the fidelity to the input signal and the other to evaluating the human-like quality of the produced motion through visual inspection. For the complete results showcasing of our assessment, please refer to the following **link**.

In addressing the first evaluation element, it becomes apparent that the outputs from PriorMDM exhibit a remarkable alignment with the prescribed trajectories compared to the other two methods, despite quantitative metrics from trajectory error analysis—especially the *average error*, which, as previously elucidated, holds significant representational value in this context—placing it third on the HumanML3D dataset and second to OmniControl on the KIT dataset. However, this outcome does not come as a surprise to us, as the evaluation metrics are computed across all joints due to the fact evaluation motions were sampled from the datasets, whereas we provide just the pelviS (x, y, z) coordinates for our experiments. Upon delving into the operational structure of PriorMDM, it becomes evident that in root trajectory control mode, PriorMDM entirely leaves out of the noising process the input features related to the root (equivalently to the pelvis), ensuring their integrity in the output regardless of external influences. This characteristic is consistent across all generated motions. Conversely, OmniControl and GMD employ mechanisms such as spatial guidance and realism guidance, which partially influence the prescribed motion to varying degrees based on the given trajectory. Diverse trajectories allow for differing levels of conformity to the control signal while still yielding a coherent output motion. This inherent flexibility occasionally leads GMD and OmniControl to diverge significantly from the given trajectories, as observed with **GMD in the oval trajectory and OmniControl in the T trajectory**. Nevertheless, it's worth noting that motions involving sharp turns, such as the small-sized T trajectory used for evaluation, were expected to pose challenges due to their inherent nature. In summary, if we were to establish a ranking based on adherence to the input signal, PriorMDM emerges as the frontrunner, followed by OmniControl at a relatively close distance, and GMD at a greater distance.

Transitioning to the second and final element of our evaluation, it's essential to underscore that the conclusions drawn here naturally follow from the preceding evaluation element. This assertion is underpinned by the inherent constraints observed in human motion joint kinematics. Essentially, the more rigorously we enforce the generated motion to adhere to the input trajectory, the greater the likelihood of introducing artifacts and inconsistencies, especially when faced with challenging prescribed trajectories—such as the T trajectory mentioned earlier. It becomes evident from the outcomes that all methods encounter such issues on certain occasions. PriorMDM, which, as dissected previously, prioritizes adherence to the motion path dictated by the trajectory to the utmost degree, raises concerns regarding the coherence of the motion while striving to faithfully follow the control signal. In scenarios like the T trajectory and the zig-zag trajectory, it exhibits motion inconsistencies during abrupt directional shifts, resulting in unnatural movements that may perturb the observer's eye. Similarly, OmniControl is compelled to execute a jarring 180-degree turn in the outcome of the right turn trajectory due to the excessively sharp prescribed directional shift, along with significant foot sliding observed in the T trajectory outcome. OmniControl, structurally oriented towards producing natural movements while allowing for more leniency in conforming to the prescribed trajectory compared to PriorMDM—as corroborated by fidelity metrics reported earlier—achieves a balance between producing natural motion in the zig-zag trajectory while loosely adhering to the sharp turns dictated by the prescribed trajectory. Regarding GMD, it demonstrates remarkable proficiency in generating natural motions. For instance, in the outcome of the right turn, it delivers the most visually appealing shift compared to other methods. However, it occasionally exhibits notable inconsistencies in appropriately adjusting the velocities included in the feature vector, appearing to struggle with following large-scale trajectories such as the oval curve. This deficiency manifests in significant foot sliding issues, as confirmed by the reported foot skating ratio metric. Additionally, it is noteworthy that GMD brings a novel approach as it performs denoising to the latent space encoded features, rather than the input features themselves, which seems to be a promising approach.

In summary, all three methods exhibit significant potential in the realm of human motion trajectory control. The choice between them ultimately hinges on the specific needs of the user. For scenarios where strict adherence to the input trajectory is paramount, particularly in computer animation systems aiming for efficiency, PriorMDM stands out as the preferred option. On the other hand, if the priority is to generate natural motions with a high level of fidelity, users may opt for OmniControl. Although GMD may fall slightly short compared to its counterparts, it shines in certain specialized scenarios. Thus, we find ourselves faced with a nuanced trade-off encompassing three key factors: adherence to the prescribed trajectory, the naturalness of the output motion, and the computational resources required. This complexity contrasts with the binary trade-off typically discussed in text-to-motion generation. Looking towards future endeavors, it is evident that novel methods should aim for a hybrid approach. These methods should strike a balance between respecting the trajectory and enhancing fidelity, taking into account the inherent flexibility of each individual trajectory. This approach would effectively mitigate trajectory-related errors and eliminate visually displeasing motion artifacts, paving the way for more sophisticated and refined motion control techniques.

Chapter 6

Conclusion

Diffusion models have become the leading type of generative models for human motion synthesis addressing a plethora of application, especially in fields like computer animation. Recent research has focused on refining the quality of motion generated samples through the iterative sampling procedure which is the cornerstone of their operational philosophy. This study investigates the performance of cutting-edge methods in two primary tasks: *text-to-motion generation* & *motion trajectory control*, particularly focusing on the pelvis joint. After providing a thorough explanation of diffusion models and supplemental material such as discussing motion representation techniques and available datasets, we delve into the core issues of these tasks. We offer a detailed overview of the top-performing methods and discuss their contributions to recent advancements in the field. Furthermore, we conduct comparative analyses of the evaluation settings to highlight the strengths and weaknesses of each method. Despite the impressive results achieved by all methods, there are still significant challenges that require further investigation. Through our comparative evaluation, we emphasize the importance of considering individual needs and weighing the pros and cons of each method, especially in terms of performance versus computational costs trade-off. We hope this survey sets the stage for future research directions in human motion synthesis, inspiring new breakthroughs in the field.

Bibliography

[1] Jonathan Ho, Ajay Jain και Pieter Abbeel. *Denoising Diffusion Probabilistic Models*, 2020.

[2] Gabriel Mongaras. *Diffusion models, DDPMs, DDIMs and classifier-Free Guidance*. Available online, 2023.

[3] Prafulla Dhariwal και Alex Nichol. *Diffusion Models Beat GANs on Image Synthesis*, 2021.

[4] researchers at the Perceiving Systems Department MPI for Intelligent Systems. *SMPL made simple FAQs*. Available online, -.

[5] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll και Michael J. Black. *SMPL: A Skinned Multi-Person Linear Model*. *ACM Trans. Graph.*, 34(6), 2015.

[6] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger και Ilya Sutskever. *Learning Transferable Visual Models From Natural Language Supervision*, 2021.

[7] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or και Amit H. Bermano. *Human Motion Diffusion Model*, 2022.

[8] Yonatan Shafir, Guy Tevet, Roy Kapon και Amit H. Bermano. *Human Motion Diffusion as a Generative Prior*, 2023.

[9] Yiming Xie, Varun Jampani, Lei Zhong, Deqing Sun και Huaizu Jiang. *Omnicontrol: Control any joint at any time for human motion generation*. *arXiv preprint arXiv:2310.08580*, 2023.

[10] Korrawe Karunratanakul, Konpat Preechakul, Supasorn Suwajanakorn και Siyu Tang. *Guided Motion Diffusion for Controllable Human Motion Synthesis*, 2023.

[11] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang και Ziwei Liu. *MotionDiffuse: Text-Driven Human Motion Generation with Diffusion Model*, 2022.

[12] Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, Jingyi Yu και Gang Yu. *Executing your Commands via Motion Diffusion in Latent Space*, 2023.

[13] Sarah Jayne Blakemore και Jean Decety. *From the perception of action to the understanding of intention*. *Nature Reviews Neuroscience*, 2001.

[14] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li και Li Cheng. *Generating Diverse and Natural 3D Human Motions From Text*. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, σελίδες 5152–5161, 2022.

[15] Simon Alexanderson, Rajmund Nagy, Jonas Beskow και Gustav Eje Henter. *Listen, Denoise, Action! Audio-Driven Motion Synthesis with Diffusion Models*. *ACM Transactions on Graphics*, 42(4):1–20, 2023.

[16] Yusuke Nishimura, Yutaka Nakamura και Hiroshi Ishiguro. *Long-Term Motion Generation for Interactive Humanoid Robots Using GAN with Convolutional Network*. *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '20, σελίδα 375–377, New York, NY, USA, 2020. Association for Computing Machinery.

[17] Gianpaolo Gulletta, Wolfram Erlhagen και Estela Bicho. *Human-Like Arm Motion Generation: A Review*. *Robotics*, 9(4), 2020.

[18] Taras Kucherenko, Dai Hasegawa, Gustav Eje Henter, Naoshi Kaneko και Hedvig Kjellström. *Analyzing Input and Output Representations for Speech-Driven Gesture Generation*. *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents*. ACM, 2019.

[19] Yann LeCun, Yoshua Bengio και Geoffrey Hinton. *Deep learning*. *nature*, 521(7553):436–444, 2015.

[20] Y. Bengio, Réjean Ducharme και Pascal Vincent. *A Neural Probabilistic Language Model*. τόμος 3, σελίδες 932–938, 2000.

[21] Diederik P Kingma και Max Welling. *Auto-Encoding Variational Bayes*, 2022.

[22] Danilo Jimenez Rezende και Shakir Mohamed. *Variational Inference with Normalizing Flows*, 2016.

[23] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville και Yoshua Bengio. *Generative Adversarial Networks*, 2014.

[24] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever και Dario Amodei. *Language Models are Few-Shot Learners*, 2020.

[25] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike και Ryan Lowe. *Training language models to follow instructions with human feedback*, 2022.

[26] Tero Karras, Samuli Laine και Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*, 2019.

[27] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen και Timo Aila. *Alias-Free Generative Adversarial Networks*, 2021.

[28] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi και David J. Fleet. *Video Diffusion Models*, 2022.

[29] Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung Woo Ha και Jinwoo Shin. *Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks*, 2022.

[30] Ben Poole, Ajay Jain, Jonathan T. Barron και Ben Mildenhall. *DreamFusion: Text-to-3D using 2D Diffusion*, 2022.

[31] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic και Sanja Fidler. *GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images*, 2022.

[32] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas και Michael J. Black. *Expressive Body Capture: 3D Hands, Face, and Body from a Single Image*, 2019.

[33] Javier Romero, Dimitrios Tzionas και Michael J. Black. *Embodied hands*. *ACM Transactions on Graphics*, 36(6):1–17, 2017.

[34] Muhammed Kocabas, Nikos Athanasiou και Michael J. Black. *VIBE: Video Inference for Human Body Pose and Shape Estimation*, 2020.

[35] Dario Pavllo, Christoph Feichtenhofer, David Grangier και Michael Auli. *3D human pose estimation in video with temporal convolutions and semi-supervised training*, 2019.

[36] Julieta Martinez, Rayat Hossain, Javier Romero και James J. Little. *A simple yet effective baseline for 3d human pose estimation*, 2017.

[37] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll και Michael J. Black. *AMASS: Archive of Motion Capture as Surface Shapes*, 2019.

[38] Zhixuan Yu, Jae Shin Yoon, In Kyu Lee, Prashanth Venkatesh, Jaesik Park, Jihun Yu και Hyun Soo Park. *HUMBI: A Large Multiview Dataset of Human Body Expressions*, 2020.

[39] Matthias Plappert, Christian Mandery και Tamim Asfour. *The KIT motion-language dataset*. *Big data*, 4(4):236–252, 2016.

[40] Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong και Li Cheng. *Action2Motion: Conditioned Generation of 3D Human Motions*. *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20. ACM, 2020.

[41] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll και Michael J. Black. *SMPL: A Skinned Multi-Person Linear Model*. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, 2015.

[42] Diederik Kingma, Tim Salimans, Ben Poole και Jonathan Ho. *Variational diffusion models*. *Advances in neural information processing systems*, 34:21696–21707, 2021.

[43] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan και Surya Ganguli. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*, 2015.

[44] Michael Gutmann και Aapo Hyvärinen. *Noise-contrastive estimation: A new estimation principle for unnormalized statistical models*. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, σελίδες 297–304. JMLR Workshop and Conference Proceedings, 2010.

[45] Diederik P Kingma, Max Welling και others. *An introduction to variational autoencoders*. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[46] Kai Lai Chung. *Markov chains*. *Springer-Verlag, New York*, 1967.

[47] Adaloglou Nikolaos Karagiannakos, Sergios. *Diffusion models: toward state-of-the-art image generation*. *https://theaisummer.com/*, 2022.

[48] Jiaming Song, Chenlin Meng και Stefano Ermon. *Denoising Diffusion Implicit Models*, 2022.

[49] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever και Mark Chen. *GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models*, 2022.

[50] Jonathan Ho και Tim Salimans. *Classifier-Free Diffusion Guidance*. *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[51] Alex Nichol και Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*, 2021.

[52] Claudio Pizzolato, Monica Reggiani, Luca Modenese και David G Lloyd. *Real-time inverse kinematics and inverse dynamics for lower limb applications using OpenSim*. *Computer methods in biomechanics and biomedical engineering*, 20(4):436–445, 2017.

[53] Jeffrey A Reinbolt, Ajay Seth και Scott L Delp. *Simulation of human movement: applications using OpenSim*. *Procedia Iutam*, 2:186–198, 2011.

[54] Alexandra Pfister, Alexandre M West, Shaw Bronner και Jack Adam Noah. *Comparative abilities of Microsoft Kinect and Vicon 3D motion capture for gait analysis*. *Journal of medical engineering & technology*, 38(5):274–280, 2014.

[55] Scott L Delp, Frank C Anderson, Allison S Arnold, Peter Loan, Ayman Habib, Chand T John, Eran Guendelman και Darryl G Thelen. *OpenSim: open-source software to create and analyze dynamic simulations of movement*. *IEEE transactions on biomedical engineering*, 54(11):1940–1950, 2007.

[56] Hervé Abdi. *The eigen-decomposition: Eigenvalues and eigenvectors*. *Encyclopedia of measurement and statistics*, σελίδες 304–308, 2007.

[57] Daniel Roetenberg, Henk Luinge, Per Slycke και others. *Xsens MVN: Full 6DOF human motion tracking using miniature inertial sensors*. *Xsens Motion Technologies BV, Tech. Rep*, 1:1–7, 2009.

[58] Ralph Gross και Jianbo Shi. *The CMU motion of body (MoBo) database*. 27, 2001.

[59] Catalin Ionescu, Dragos Papava, Vlad Olaru και Cristian Sminchisescu. *Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014.

[60] Saeed Ghorbani, Kimia Mahdaviani, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm και Nikolaus F. Troje. *MoVi: A large multi-purpose human motion and video dataset*. *PLOS ONE*, 16(6):e0253157, 2021.

[61] Abhinanda R. Punnakkal, Arjun Chandrasekaran, Nikos Athanasiou, Alejandra Quiros-Ramirez και Michael J. Black. *BABEL: Bodies, Action and Behavior with English Labels*. *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, σελίδες 722–731, 2021.

[62] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li και Li Cheng. *Generating Diverse and Natural 3D Human Motions from Text*. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, σελίδες 5142–5151, 2022.

[63] Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong και Li Cheng. *Action2Motion: Conditioned Generation of 3D Human Motions*. *Proceedings of the 28th ACM International Conference on Multimedia*, MM ’20. ACM, 2020.

[64] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll και Michael J. Black. *AMASS: Archive of Motion Capture as Surface Shapes*, 2019.

[65] Sebastian Starke, He Zhang, Taku Komura και Jun Saito. *Neural State Machine for Character-Scene Interactions*. *ACM Transactions on Graphics*, 38, 2019.

[66] D. Hirshberg, M. Loper, E. Rachlin και M.J. Black. *Coregistration: Simultaneous alignment and modeling of articulated 3D shape*. *European Conf. on Computer Vision (ECCV)*, LNCS 7577, Part IV, σελίδες 242–255. Springer-Verlag, 2012.

[67] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Joseph Rodgers και James Davis. *SCAPE: shape completion and animation of people*. *ACM Transactions on Graphics (TOG)*, τόμος 24, σελίδες 408–416. ACM, 2005.

[68] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Joseph Rodgers και James Davis. *BlendSCAPE: Blending shape, color, and texture for realistic human facial animation*. *ACM Transactions on Graphics (TOG)*, 27(3):Article 53, 2008.

[69] Matthew M. Loper και Michael J. Black. *OpenDR: An Approximate Differentiable Renderer*. *Computer Vision - ECCV 2014*David Fleet, Tomas Pajdla, Bernt Schiele και Tinne Tuytelaars, επιμελητές, σελίδες 154–169, Cham, 2014. Springer International Publishing.

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser και Illia Polosukhin. *Attention Is All You Need*, 2023.

[71] Alec Radford και Karthik Narasimhan. *Improving Language Understanding by Generative Pre-Training*. 2018.

[72] Jacob Devlin, Ming Wei Chang, Kenton Lee και Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2019.

[73] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei και Ilya Sutskever. *Language Models are Unsupervised Multitask Learners*. 2019.

[74] Rico Sennrich, Barry Haddow και Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units*, 2016.

[75] Mathis Petrovich, Michael J. Black και Gül Varol. *TEMOS: Generating diverse human motions from textual descriptions*, 2022.

[76] Yinglin Duan, Tianyang Shi, Zhengxia Zou, Yenan Lin, Zhehui Qian, Bohan Zhang και Yi Yuan. *Single-Shot Motion Completion with Transformer*, 2021.

[77] Lvmin Zhang, Anyi Rao και Maneesh Agrawala. *Adding Conditional Control to Text-to-Image Diffusion Models*, 2023.

[78] Ilya Loshchilov και Frank Hutter. *Decoupled Weight Decay Regularization*, 2019.

[79] Jonathan Ho και Tim Salimans. *Classifier-Free Diffusion Guidance*, 2022.

[80] Olaf Ronneberger, Philipp Fischer και Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*, 2015.

[81] Guy Tevet, Brian Gordon, Amir Hertz, Amit H. Bermano και Daniel Cohen-Or. *MotionCLIP: Exposing Human Motion Generation to CLIP Space*, 2022.

[82] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi και Hongsheng Li. *Efficient Attention: Attention with Linear Complexities*, 2024.

[83] Mathis Petrovich, Michael J. Black και Gül Varol. *Action-Conditioned 3D Human Motion Synthesis with Transformer VAE*, 2021.

List of Abbreviations

AR	Augmented Reality
VR	Virtual Reality
VAE	Variational Autoencoders
GAN	Generative Adversarial Network
DDPM	Denoising Diffusion Probabilistic Models
SMPL	Skinned Multi-Person Linear
DDIM	Denoising Diffusion Implicit Model
FID	Frechet Inception Distance
IS	Inception Score
NLL	Negative Log-Likelihood
ELBO	Evidence Lower Bound
DOF	Degree of Freedom
LBS	Linear Blend Skinning
PCA	Principal Component Analysis
BPE	Byte-Pair Encoding
MDM	Motion Diffusion Model
GMD	Guided Motion Diffusion
MLD	Motion Latent Diffusion
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error