Developing Innovative Technologies for Creating Realistic Animation of Digital Characters for Real-time Environments

by

Junheng Fang

National Centre for Computer Animation

Faculty of Media & Communication

Bournemouth University

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of *Doctor of Philosophy*

December. 2024

Copyright Statement

Acknowledgements

First and foremost, I am profoundly thankful to my mentors, Prof. Lihua You, Prof. Jian Jun Zhang, and Dr. Ehtzaz Chaudhry, for their patience, motivation, and immense knowledge. Specifically, I appreciate Dr. Ehtzaz Chaudhry for his trust in me and his project support. I sincerely thank Prof. Jian Jun Zhang for his support throughout my journey. Jian Jun is a charismatic leader who always provides constructive guidance and help in both daily life and study. I am deeply grateful to Prof. Lihua You for his ongoing encouragement, advice, extensive academic support, and faith in me. He is an ideal mentor offering advice and encouragement with a correct and serious academic attitude. I can not achieve this without his valuable guidance and involvement.

Special thanks to Prof. Xiaokun Wang from the University of Science and Technology Beijing for his invaluable support and generous help. His insightful views, research attitude, and warm encouragement with a perfect blend of insight and humour motivated me at all stages.

My heartfelt thanks go to Dr. Nan Xiang, Dr. Yao Lyu, Dr. Ruibin Wang, Dr. Shuaiying Hou, Dr. Zhaohua Zheng and Hongyu Tao. Their selfless help before the deadline and their deep friendship have been a source of strength.

Thanks to Jiaxin Cai for being the rock in my life. Her unwavering encouragement and understanding helped me navigate through the challenging moments during my PhD study.

Thanks to my parents who have always supported my choices and believed me.

Last but not least, I have the deepest appreciation for Bournemouth University. Without its supportive academic environment, my doctoral research would not have been possible.

Abstract

The digital entertainment industry has seen exponential growth in recent years, of which the developments have been paralleled by the evolution of the Metaverse. The Metaverse represents a convergence of virtual and real worlds, where users interact through digital characters, necessitating advanced and authentic digital character animations. Traditional methods for creating 3D digital character animations involve complex, labour-intensive processes that are not conducive to real-time applications or large-scale production due to their high computational costs and extensive data requirements. The rapid advancement of virtual technology, big data, and artificial intelligence, especially in generative AI, offers promising solutions to these challenges. However, key research questions in this topic still remain, including how to create authentic 3D models with small data sizes, how to generate model animations with dynamic effects in real-time, and how to create the skeletal motion of digital humans efficiently.

This thesis develops innovative technologies for creating realistic and immersive animations of digital characters in real-time environments, focusing on three primary procedures: digital character modelling, skin deformation methods, and skeletal motion construction. The contributions we provided in this research are concisely listed as follows:

- A survey summarizing the recent developments and applications of position-based approaches is proposed. This survey includes the core idea of the position based dynamics (PBD) method, the advancements made inside the algorithm, the applications in other fields, and some guidance on the research directions for future work.

- An innovative modelling method is introduced, integrating the governing equation of elastic beam deformation and Newton's second law to reconstruct dynamic 3D models with high accuracy and reduced data volumes. This PDE-based approach offers a time-dependent solution for creating detailed deformable models, significantly improving upon other baseline surface reconstruction techniques.
- A novel method for facial blendshape generation is proposed, leveraging an ODE-based surface creation method and Newton's second law to produce natural facial animations. This method effectively reduces the data size while maintaining natural edge continuities and high efficiency in creating interpolated facial animations.
- A new neural network structure, the Video-to-Motion (VTM) framework, is developed to reconstruct skeletal motion from video sequences. By pre-learning motion priors and jointly training them with the model, this method ensures high-fidelity motion reconstruction with substantial computational efficiency.

To evaluate the efficacy of our proposed methods, various comprehensive experiments are conducted. These experiments demonstrate that the developed technologies can significantly benefit the process of creating realistic and immersive animations for digital characters in real-time interactive scenarios. We believe that the completely constructed digital character animation can alleviate the labour cost in the game and animation industries and can be integrated with other advanced game technologies to lead more users to enter the Metaverse, resulting in expanding the market and generating more wealth.

Contents

Copyright i

Acknowledgements ii

Abstract iii

List of Figures ix

List of Tables xiv

Declaration xv

1 Introduction 1

1.1 Background 1
1.2 Research Questions 5
1.3 Aims and Objects 6
1.4 Contributions 7
1.5 List of Publications 8
1.6 Outline of Thesis 9

2 Literature Review 11

2.1 Overview of 3D Modelling Methods 11
2.1.1 Explicit Representation Methods 12
2.1.1.1 Point Clouds 12
2.1.1.2 Voxels 14
2.1.1.3 Meshes 15
2.1.2 Implicit Representation Methods 19
2.1.2.1 Algebraic Surfaces 20
2.1.2.2 Constructive Solid Geometry 21
2.1.2.3 Signed Distance Function 21
2.1.2.4 Level Set 23

2.1.2.5 Neural Radiance Fields 23
2.2 Overview of Skin Deformation Methods 24
2.2.1 Geometric Skinning 25
2.2.2 Example-based Skinning 27
2.2.3 Physics-based Skinning 29
2.2.4 Position Based Dynamics 31
2.3 Overview of Motion Reconstruction Methods 32
2.3.1 Traditional Methods for 3D HPE 35
2.3.1.1 Generative Methods 35
2.3.1.2 Discriminative Methods 36
2.3.1.3 Template Matching Methods 36
2.3.2 Deep Learning-based Methods for 3D HPE 37
2.3.2.1 Direct Regression-based Methods 37
2.3.2.2 2D Information-based Methods 39

3 Review on State-of-the-art Improvements & Applications of PBD 42
3.1 Introduction 42
3.2 Basic Concept of PBD 44
3.2.1 Overview of PBD 44
3.2.2 Damping 45
3.2.3 Collision 45
3.2.4 Solver 47
3.3 Recent Improvements of PBD 47
3.3.1 Improvements in Convergence Problem 48
3.3.2 Improvements in Dependence Problem 48
3.3.2.1 Extended Position Based Dynamics . . . 49
3.3.2.2 Projective Dynamics 50
3.3.3 Improvements in Other Limitations 51
3.3.4 Improvements in Extensions 52
3.3.4.1 Cloth Simulation 52
3.3.4.2 Fluid Simulation 52
3.3.4.3 Rigid Body Simulation 53
3.3.5 Summary Table 55
3.4 Recent Applications of PBD 64
3.4.1 Deep Learning-related Application 64
3.4.2 Medicine Application 64
3.4.3 Architectural Application 66

3.4.4 Summary Table 66
3.5 Conclusion 70

4 PDE-based Dynamic Reconstruction Integrating PBD 71
4.1 Introduction 71
4.2 Deformation Simulation by PBD 74
4.3 PDE-based Surface Reconstruction 75
4.3.1 Mathematical Model 76
4.3.2 Closed-form Solutions to PDE 77
4.3.3 Extend PDE with Infinite Terms 79
4.3.4 Transformation Elimination 81
4.3.5 Approximation for Constants 83
4.4 Experiment 1: Dynamic Model Reconstruction 83
4.4.1 Summary 94
4.5 Experiment 2: Generation of In-between Keyframe Models 95
4.5.1 Summary 101
4.6 Conclusion 101

5 Skin deformation Method Based on Newton's Second Law for Generating Natural Human Facial Blendshapes 103
5.1 ODE-based Surface Creation 104
5.1.1 Mathematical Model 104
5.1.2 Closed-form Solution 105
5.2 Facial Blendshape based on Newton's Second Law 107
5.2.1 Algorithm 107
5.2.2 Experiment 109
5.2.2.1 Patch Segmentation 109
5.2.2.2 Surface Reconstruction 110
5.2.2.3 Facial Blendshape 112
5.3 Conclusion 114

6 Advanced Motion Prediction Method from Monocular Videos 115
6.1 Introduction 115
6.2 Process of VTM 118
6.2.1 Dataset Preparation 119
6.2.1.1 Dataset Selection 119
6.2.1.2 Data Preprocess 119

6.2.1.3 3D Motion Representation 119
6.2.1.4 2D Keypoints Representation 120
6.2.2 Pre-training Motion Priors 120
6.2.2.1 TPME 120
6.2.2.2 TPMD 121
6.2.2.3 RD 121
6.2.3 Learn Human Motion from Videos 121
6.2.3.1 TPVE 121
6.2.3.2 BRP 122
6.2.3.3 RTP 122
6.2.4 Training Loss 123
6.2.4.1 Loss for Motion Priors 123
6.2.4.2 Loss for Motion Reconstruction 123
6.2.4.3 Root Translation Reconstruction 124
6.3 Experiments 124
6.3.1 Implementation Details 124
6.3.2 Comparisons 125
6.3.2.1 Quantitative Comparisons 125
6.3.2.2 Qualitative Comparisons 127
6.3.3 Evaluations 129
6.3.3.1 Robustness to Unseen View Angles . . . 129
6.3.3.2 Robustness to in-the-wild videos 129
6.3.3.3 Evaluations on MoAE 131
6.3.3.4 Evaluations on Motion Priors 131
6.3.3.5 Evaluations on two-part design 133
6.3.3.6 Other Evaluations 133
6.3.4 Skeleton-driven Mesh Animation 135
6.4 Conclusion 136

7 Conclusion and Future Work **138**
7.1 Limitations and Future Work 139

Bibliography **142**

List of Figures

1.1 The inaugural flat virtual idol Lynn Minmay as the cover of the cartoon *Super Dimension Fortress Macross* (Macross Wiki 2024). 3
1.2 Example of three classic types of applications on digital human from SEMA Game Studio (2024): (a) *Necromancer* is a virtual idol; (b) *Ironman* is a reproduced movie character; (c) *Asian Woman* is a reconstructed human avatar. 3

2.1 (a) The point cloud model of "Stanford Bunny"; (b) "Stanford Bunny"; (c) The voxel model of "Stanford Bunny" . 13
2.2 The polygon mesh model of a dolphin. 16
2.3 Surfaces created by the PDE-based modelling technique (Fu et al. 2022). 20
2.4 Two simple examples of algebraic surfaces: the left one is a sphere, and the right is an annulus. 21
2.5 A complex object constructed from two simple primitives, a sphere and a cube, by Boolean operations (Renno and Papa 2015). 22
2.6 A 2D example of SDF. Each value of the grid stands for the nearest distance from the point to the surface (Zucker et al. 2010). 22
2.7 Apply algebraic operations on two SDF models to blend new and complex model shapes. 23
2.8 The main procedure of reconstructing 3D model from input images of different views by neural radiance fields (Mildenhall et al. 2021). 24
2.9 An example of geometric modelling (Singh and Kokkevis 2000). 25

2.10 The left is the reference pose, whilst the right is a typical example of "candy-wrapper" artefact, where $\mathbf{v}$ denotes the vertex attached to the joints $j_1, j_2, \ldots, j_n$ (n stands for the number of influencing joints); $C_j \in \mathbb{SE}(3)$ denotes the transformation matrix from the rest pose of joint j_n to its actual position (Kavan et al. 2008). 26
2.11 The neural model overview of DeePSD (Bertiche et al. 2021), which integrates neural network and machine learning to simulate deformation. 29
2.12 The process of applying underlying physics laws to model (Bian et al. 2019). 30
2.13 An example of a motion-capturing device, consisting of 17 inertial and magnetic sensor modules (Roetenberg et al. 2009). 33

3.1 Some examples of PBD simulated scenes. The left is a solid ball collided with a deformable soft cube; the middle is a scene with many twisted bars; the right shows the simulation of fluids (Bender et al. 2017). 43
3.2 Three different conditions of object collision. 46
3.3 The simulation results of a twisted rope demonstrate the ability of XPBD to generate detailed deformation results (Müller et al. 2020). 49
3.4 The computational time for each frame after $1, 10, 20$ iterations of simulating a hippopotamus model by PD and Newton's method from (**?**). It could be observed that, after 10 iterations, the simulation results retain as many details as the physics-based method, whilst PD only uses 1/10 time of the latter for simulating one frame. 50
3.5 The simulation result of a folded rod. The left is a real elastic rod, while the right is a simulated rod generated by PD (Soler et al. 2018). 51
3.6 The simulation result of rigid bodies. The top has 2000 elastic objects (Deul et al. 2016), while the bottom has 5000 rigid boxes (Weiss et al. 2017). 54
3.7 The simulation result of fluids interacting with a rigid body, produced by SPNets (Schenck and Fox 2018). . . . 65

3.8 The simulation result of soft tissue incisions in a virtual surgery (Berndt et al. 2017). 65
3.9 The simulation result of automatic city layout in solving an urban design task (Cao and Ji 2021). 66

4.1 A human face model and its reconstruction created by ODE-based surface creation method. 72
4.2 The deformation simulation results of a horse model, generated by PBD. From the top left, these models represent the deformed horse at frame 1,20,30,40,50,60,70,80,90,100, which proves the great simulation performance of PBD. . 74
4.3 The comparison between the original and reconstructed curves with 71 points at the 20^{th} and 100^{th} frames. (a) Original curves at the 20^{th} and 100^{th} frames. (b) PDE reconstructed curves at the 20^{th} and 100^{th} frames. (c) B-spline reconstructed curves at the 20^{th} and 100^{th} frames. (d) Bézier reconstructed curves at the 20^{th} and 100^{th} frames. (e) Original, PDE, B-spline, and Bézier curves at the 20^{th} frame. (f) Original, PDE, B-spline, and Bézier curves at the 100^{th} frame. 86
4.4 The comparison between the original and reconstructed curves with 71 points at the 37^{th} and 38^{th} frames. (a) Original curves at the 37^{th} and 38^{th} frames. (b) PDE reconstructed curves at the 37^{th} and 38^{th} frames. (c) B-spline reconstructed curves at the 37^{th} and 38^{th} frames. (d) Bézier reconstructed curves at the 37^{th} and 38^{th} frames. (e) Original, PDE, B-spline, and Bézier curves at the 37^{th} frame. (f) Original, PDE, B-spline, and Bézier curves at the 38^{th} frame. 90
4.5 The undeformed horse model and its extracted curves. (a) Polygon horse model, (b) extracted curves. 92
4.6 The comparison among the original deformation of the horse model generated by PBD and the reconstructed shapes created by our proposed method and B-spline method at frame $1, 20, 30, 40, 50, 60, 70, 80, 90, 100$. 93

4.7 The comparison among the original deformation of the armadillo model generated by PBD and the reconstructed shapes created by our proposed method and B-spline method at frame $1, 20, 30, 40, 50, 60, 70, 80, 90, 100$. 94
4.8 The models used for implementing our proposed method. From left to right are cube-rope, rings and dumpling. We take the cube-rope model as the example to explain our experiment and compare the errors and running time for all models with the PBD results and B-Spline. 96
4.9 Curve representations of the cube-rope models shown in Fig. 4.8. 96
4.10 The framework of our experiment. We use simulation results generated by PBD at frames 0,2,4,6,8 and 10. After extracting the curves from the original models, we get the curves to fit our mathematical model. Then, the curve information at frames 1,3,5,7, and 9 were computed by the trained mathematical model. The calculated curves were then used to reconstruct the mesh models, which are compared with the original PBD deformation models at the same frames. As could be seen from the details on the right parts, few differences could be found between the original model from PBD and the reconstructed model by our proposed method. 97
4.11 Visual comparison among the baseline curves highlighted in blue and reconstructed curves with the proposed algorithm highlighted in red and B-Spline curves combined with linear interpolation highlighted in pink. 98

5.1 ODE-based surface creation by two boundary and four control curves in (Bian et al. 2019). 105
5.2 Two face models with the same topology but different poses, achieved from (Sumner and Popović 2004). The left is a neutral model, while the right is a laugh model. . 109
5.3 Face patch segmentation. 110
5.4 Extracted wireframes of each face patch. The left is the six control curves of each patch, and the right is all curves generated by the ODE-based surface creation method using extracted control curves. 111

5.5 Comparison of the original and recreated models reveals that the original laughing model contains detailed features like nasolabial folds, which are effectively restored using the ODE-based surface recreation. The ODE sweeping technique preserves significant details by manipulating the control curves. 111
5.6 Comparison between 29 reconstructed keyframe models by our proposed method and LBS. We use the green and blue boxes to emphasize the shape alteration of the mouth and eyes. 113

6.1 The overview of our VTM framework. 118
6.2 The qualitative comparisons on motion reconstruction of our VTM to SOTA methods. 127
6.3 The reconstruction results of the same performer at the same frame from different unseen view angles. 129
6.4 The reconstruction results from in-the-wild videos. 130
6.5 Another reconstruction results from in-the-wild videos. . 130
6.6 Sample frame comparisons among videos, predicted skeletons, and mesh animations. 136

List of Tables

3.1 Contributions, advantages, and defeats of improved PBD algorithm. 56
3.2 Contributions, advantages, and defeats of extensions in XPBD & PD. 58
3.3 Contributions, advantages, and defeats of PBD improvements for different scenarios. 60
3.4 PBD applications in different fields. 68

4.1 The comparison among three methods of design variables, errors($\times 10^{-3}$), and CPU computing time for reconstruction of frame 20 and 100. 89
4.2 The comparison among three methods of design variables, errors($\times 10^{-4}$), and CPU computing time for reconstruction of frame 37 and 38. 91
4.3 Average errors and maximum errors of our method and B-Spline compared with PBD results. The three models with different complexity of curves are all better fitted with our proposed method with less error of both average error E_{An} and maximum error E_{Mn}. 100

5.1 The numerical comparison between the original models and the reconstructed models. 112

6.1 The quantitative comparisons on metrics of our VTM to SOTA methods. 128
6.2 The evaluation comparisons of MoAE and its variants on MPJPE, PA-MPJPE, and MRPE. 131
6.3 The evaluation comparisons of VTM and its variants on MPJPE, PA-MPJPE, and MRPE. 134

Declaration Statement

This thesis is submitted in fulfilment of the requirement for transfer from Master of Philosophy (MPhil) to the Doctor of Philosophy (PhD) at Bournemouth University.

Chapter 1

Introduction

1.1 Background

According to the "Global Games Market Report 2023" (Newzoo 2024) released by *Newzoo*, the number of global players reached 3,305million in 2023, marking a 4.3% increase from 3,168 million in 2022, with a projected rise to 3,675 million by 2026. In terms of revenue, the gaming industry's total revenue reached $184.0 billion in 2023, expected to reach $205.4 billion by 2026. Additionally, based on the "Global and China Animation Industry Report, 2019-2025," (Research China 2024) released by *Research In China*, the global animation industry had an output value of approximately $500 billion in 2023. In the current stage of the Metaverse, known as "the coexistence of virtual and real", users in these two vast commercial markets place more emphasis on the role of virtual avatars and require comprehensive, authentic, and natural animations of digital characters to enhance their experience in the virtual world.

The concept of the "Metaverse" originates from *Snow Crash*, a science fiction published in 1992. In this novel, humans could enter digital space via virtual avatars and interact with each other. Fast forward 30 years of technological advancement in computer vision and graphics, the "Metaverse" has evolved from a mere idea in fiction to the forefront of internet digital economy development. From the development practices and implementation scenarios (Guo 2022), it can be observed that the Metaverse undergoes three stages: cloud gaming, digital twins, and the coexistence of virtual and real. Virtual avatars will greatly enhance the realism of various characters in the virtual world and provide each user with their own virtual idol, facilitating better social interaction within games. In the ultimate stage of the Metaverse, every real-world user can

create their own digital avatar in the virtual world and freely navigate across different Metaverses. Each individual in the real world engages in activities such as learning, working, investing, creating, and consuming, while their digital avatar in the virtual space participates in activities like creating, gaming, experiencing, trading, and investing. In this "co-existence of virtual and real" stage, real-world humans and their digital avatars will form new social relationships and emotional connections, establishing a novel "human society" where virtual and real coexist. To provide users with an immersive experience in the Metaverse, digital characters require realistic appearances, smooth body movements, and natural language expressions. Moreover, due to humans' inherent social attributes, interactive capability is crucial for digital characters in the Metaverse. Therefore, the large-scale generation of high-quality digital characters in real-time is essential for users to enter the Metaverse and other virtual worlds, as well as allowing users to explore broader digital spaces. In the realm of digital characters, the role of digital humans is particularly significant. They serve as representations of human beings in the Metaverse, acting as essential components of human society's reconfiguration within this digital space. Enhancing the realism and naturalness of digital humans can greatly enhance the immersive experience for Metaverse users.

The concept of "digital human" originates from the 1985 essay *Cyborg Manifesto* (Haraway 1985), in which Haraway defined "Cyborg" as a combination of inorganic machines and living organisms. In 1986, the U.S. National Library of Medicine (NLM) initiated the *Visible Human Project* (National Library of Medicine 1994), which aimed to achieve the 3D display of human anatomy. This was the first time the term "Digital Human" was introduced. Over time, the concept has gradually expanded, referring to virtual characters created using digital technology. These digital humans can be presented in 2D or 3D forms. Early virtual digital humans mainly consisted of flat cartoon idols [1] and lacked complex processes like 3D modelling, hence limiting their realism and depth. Nevertheless, with the rapid advancement in computer graphics technology, including 3D modelling and physical deformation simulation techniques,

[1]The inaugural virtual idol, Lynn Minmay, made her debut in the 1982 cartoon Super Dimension Fortress Macross, as shown in Fig. 1.1.

Figure 1.1: The inaugural flat virtual idol Lynn Minmay as the cover of the cartoon *Super Dimension Fortress Macross* (Macross Wiki 2024).

(a) Necromancer (b) Ironman (c) Asian Woman

Figure 1.2: Example of three classic types of applications on digital human from SEMA Game Studio (2024): (a) *Necromancer* is a virtual idol; (b) *Ironman* is a reproduced movie character; (c) *Asian Woman* is a reconstructed human avatar.

the trend in digital human research has shifted. Leading research institutions and commercial teams can now produce highly realistic virtual digital humans, which can be applied successfully in virtual idol generation, movie character production, and human avatar reconstruction, as shown in Fig. 1.2. Yet, the traditional process for constructing a complete animation of 3D digital humans involves complex procedures, including image collection, model creation, texture mapping, as well as motion capture and driving. This process requires specialized sensing devices and labour-intensive work, making it time-consuming, custom-made only, and expensive for storage and network transfer resources. It cannot meet the demands for fast-generating large-scale digital humans with good realism but with small data, hindering widespread adoption and application.

In recent years, there have been remarkable advancements in virtual technology, big data, artificial intelligence (AI), and other technologies, particularly in the domains of generative AI, like image generation and language interaction. These advancements have led to the evolution of digital human technology, which creates more realistic images, offers more scopes, and brings more commercial value. This has resulted in the term "generative digital characters", which refers to the use of generative AI techniques to create incredibly lifelike 3D digital characters. It is achieved by assimilating real data distribution in a data-driven manner, sampling the data distribution to create new representations, and finally rendering the data representation into the digital characters form. Generative AI has made the process of constructing 3D digital character models more efficient, resulting in improved realism and boundless potential for further development. However, methods integrating deep learning methods with 3D modelling still face the challenge of requiring extensive datasets for network training. The massive amount of data results in high computational demands, which in turn require powerful computer devices. Real-time 3D digital character construction is challenging due to these computational requirements. The focus of current research lies in reducing the computational workload without compromising quality.

Therefore, generating visually plausible digital characters in real-time environments has become the primary task in the game and animation industry. Existing digital character-related technologies involve each procedure in the process. In Chapter 2, we will investigate the three main

procedures: digital characters modelling (DCM), skin deformation methods (SDM), and skeletal motion construction. This research aims to explore the potential of innovative technologies in developing real-time, authentic, and immersive digital character animations to enhance user experiences in interactive environments. By using these technologies, artists can save time and resources in creating keyframe models with high similarities and avoid dealing with unnatural artefacts on skins. The detailed research questions and objectives will be presented in the following sections.

1.2 Research Questions

In light of the factors elucidated above, the research questions of this investigation can be delineated as follows:

- **Q1. How to create authentic 3D models with a small data size?** The geometric modelling technologies use low-level manual operations such as manipulations of surface vertices of polygon models, control points of B-Spline models, surface curves, etc., to create 3D models. These methods could achieve fast creation of low-poly models, whilst the details and realism could not be guaranteed. Moreover, mesh simplification methods are used to transfer high poly models into low poly models to fulfil the frames per second(fps) requirements in games and movies. To represent detailed and realistic 3D models with a small data size is a very challenging task.

- **Q2. How to create model animation with dynamic effects of movements and deformations in real-time environments?** An animation involves a series of keyframe models to form the mesh deformation and model movement. Individually creating a model at each frame during deformation is time-consuming and is difficult to maintain consistency. Purely geometric computer animation techniques like Linear Blend Skinning (LBS) (Magnenat et al. 1988), have high computational efficiency but poor realism, as they do not consider any underlying physical laws. Physics-based computer animation technologies follow the underlying physical laws of object movements and deformations, and can create realistic computer animation. However, the expensive computational

cost makes it hard to implement in real-time environments like interactive games. The method to generate realistic skin deformation is full of challenges.

- **Q3. How to create accurate skeletal motion of digital humans?** While skin deformation techniques are useful for creating natural deformation simulations of meshes, the motion realism of the entire model in real-time animation is still lacking. It is a significant issue to construct reasonable skeletal motion and utilize it to drive digital characters, creating animations with realistic movements. Existing motion creation methods all require significant human involvement. However, recent advancements in deep learning and neural networks have made it possible to tackle the challenging task of reconstructing skeletal motion from videos.

1.3 Aims and Objects

This research endeavours to create dynamic digital human animations in real time, ensuring authenticity and immersion to enrich user engagements within interactive environments. It also seeks to address the aforementioned research questions comprehensively. To achieve the goal, the following specific objectives must be attained:

- Review the state-of-the-art skin deformation techniques and investigate their performance in efficiency and authenticity. Choose the most proper method and write a survey about its development and progress. Gain an understanding of the current techniques and solutions, as well as identify their limitations to make sufficient preparations for the following work.

- Review current techniques for mathematical mapping and 3D modelling, and investigate the latest research on high-level 3D model creation methods that involve data reduction. Propose an innovative modelling method to preserve model details as much as possible with a small data volume. It is supposed to provide a controllable and realistic model, releasing the burden on storage and internet transmission.

- Propose a novel facial blendshape method for fast-generating interpolations between different facial poses, as facial expression changes apart from the other skin deformation of digital humans. It is supposed to save significant time and effort for artists who create models for adjacent frames, reducing the need for tedious rework.

- Review the following topics: skeletal motion creation, Human Pose Estimation(HPE), object detection, and generative digital human techniques. Propose a novel method to estimate motion from videos of a moving digital human. It is supposed to be effective and robust for accurate motion reconstruction from any video.

1.4 Contributions

This research contributes to the process of creating a complete animation of digital humans with realistic movements in real-time environments. Three primary stages are focused on: digital human modelling, skin deformation methods, and skeletal motion construction. Specifically, the main contributions can be summarized as:

- A survey is proposed that summarizes the developments and applications in position-based approaches since 2018. This survey covers the baseline algorithm of the original position based dynamics (PBD) method. It also reviews the advancements made in the algorithm, such as solvers and constraints, and introduces significant implementations of PBD, involving various fields and industries. Additionally, the survey provides some guidance on the research directions of position-based approaches for future work. (Chapter 3)

- An innovative modelling method that reconstructs dynamic 3D models with skin deformation obtained from PBD simulation is presented. It combines the object motion described by Newton's second law and the bending deformation of elastic beams controlled by the governing equation to develop a new mathematical model. This PDE-based mathematical model solves the closed-form solution by separating variables, which are applied to recreate deformed meshes at different frames. Compared with traditional surface reconstruction techniques, such as Bézier and B-spline, the proposed

method has shown its ability to reconstruct the deformed 3D models with high accuracy and a small data volume. (Chapter 4)

- A new facial skin deformation method is introduced. It integrates ODE sweeping surfaces and Newton's second law to represent face models at different poses, generating physics-based facial blendshapes. This skin deformation method inherits advantages of ODE sweeping surfaces, such as small data sizes and natural maintenance of edge continuities. The experiment results have proven that it can create more natural interpolated facial animation with high efficiency. (Chapter 5)

- A novel neural network, the Video-to-Motion Generator (VTM), is provided for constructing the skeletal motion of digital humans from video sequences. Moreover, a new method is proposed to manipulate 3D motion priors with distinct scales, which is achieved by aligning video and motion data on two-body partial potential feature manifolds. The experimental results have demonstrated that the proposed method can effectively replicate high-quality motion while maintaining a high degree of fidelity to the provided video. (Chapter 6)

1.5 List of Publications

- **P1 Fang, J.**, You. L, Chaudhry, E. and Zhang. J. J., 2023. State-of-the-art improvements and applications of position based dynamics. *Computer Animation and Virtual Worlds*, 34 (5), e2143.

- **P2 Fang, J.**, Chaudhry, E., Iglesias, A., Macey, J., You, L., and Zhang, J. J., 2022. Reconstructing dynamic 3d models with small data by integrating position-based dynamics and pde-based modelling. *Mathematics*, 10 (5), 821.

- **P3 Fang, J.**, Zhu, X., You, L. and Zhang, J. J., 2023. Efficient dynamic deformation simulation by integrating pde-based reconstruction and xpbd. *In 36th International Conference on Computer Animation and Social Agents*, IEEE.

- **P4 Fang, J.**, Bian, S., Macey, J., Iglesias, A., Ugail, H., Malyshev, A., Chaudhry, E., You, L. and Zhang, J. J., 2021. Efficient and

physics-based facial blendshapes based on ode sweeping surface and newton's second law. *In 25th International Conference Information Visualisation (IV)*, IEEE, 303-309.

1.6 Outline of Thesis

The following chapters of this thesis are structured as below:

- **Chapter 2.** This chapter thoroughly investigates the related techniques in the three primary procedures of digital character animation creation: modelling, skin deformation, and motion construction. These methods inspire the following works in this research.

- **Chapter 3.** This chapter first introduces how the basic PBD achieves the visually plausible deformation simulation results. Then the improvements for addressing the inherent limitations of PBD, and applications for integration with various industries since 2018 are thoroughly reviewed. This survey work was published in *Computer Animation and Virtual Worlds* (**P1**).

- **Chapter 4.** This chapter introduces a PDE-based modelling approach for fast and precise model reconstruction in dynamic scenarios. This method integrates the governing equation of elastic beams with Newton's second law, which enables it to be time-dependent. With it, detailed deformable models with small data sizes can be created for creating animation. This work was published in *Mathematics* (**P2**) and its related work was published in *in 36th International Conference on Computer Animation and Social Agents* (**P3**).

- **Chapter 5.** This chapter introduces a physics-based skin deformation approach for creating natural facial blendshapes by integrating an improved ODE-based surface creation method, which is derived from Newton's second law. This work was published in *In 25th International Conference Information Visualisation (IV)* (**P4**).

- **Chapter 6.** This chapter introduces a framework named as Video-to-Motion for predicting human motion from monocular videos. This framework first utilizes a two-part motion auto-encoder to learn motion priors and then leverages a two-part visual encoder

with the learned motion priors to reconstruct the skeletal motion from in-the-wild and unseen-view-angle videos.

- **Chapter 7.** This chapter comprises conclusions and future plans for this research.

Chapter 2

Literature Review

The aim of this research is to develop advanced animation technologies for generating realistic digital characters in real-time environments, which involves three main procedures: digital character modelling (DCM), skin deformation methods (SDM), and skeletal motion construction. To begin with, the current modelling method for 3D models will be reviewed in Section 2.1. Then, this chapter reviews the existing methods for skin deformation as well as their extensions in Section 2.2. Additionally, advanced methods for skeleton-based digital character representation and digital character pose estimation are surveyed in Section 2.3.

2.1 Overview of 3D Modelling Methods

Digital characters can be thought of as projections of real characters in the "Metaverse." They not only require realistic simulation of character appearance but also approximate character behaviours, movements, and facial expressions. Therefore, as the foundation of digital characters, 3D modelling methods are utilized to closely represent digital character skins and establish various physical attributes that correspond to character behaviour. In 3D modelling, there are generally two approaches: explicit and implicit methods. Explicit methods involve directly defining a set of elements that satisfy certain conditions. For example, point clouds contain the positions of points in three-dimensional space, while polygon meshes contain information about vertex positions and their connectivity. In industrial applications like gaming and film production, explicit representation models are primarily used, as traditional rendering pipelines can efficiently handle their display (especially those based on polygon meshes). However, the level of detail in explicit methods is limited by

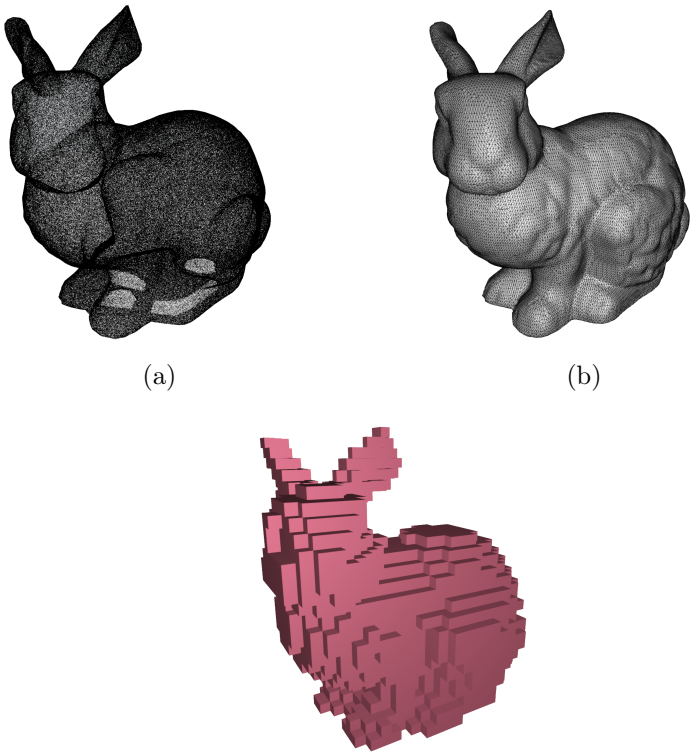
the model resolution. To generate highly realistic digital characters, a large number of elements are needed to approximate the model's details, leading to increased model complexity. On the other hand, implicit methods only require specifying some constraints on three-dimensional space, such as signed distance functions (SDF) or level set method. With the advancement of deep learning, more and more methods are utilizing neural networks to approximate implicit functions, such as deep signed distance functions(DeepSDF) (Park et al. 2019) and neural radiance fields(NeRF) (Mildenhall et al. 2021), resulting in finer geometric details of digital humans. Implicit methods, being a more flexible approach, allow digital human models to overcome spatial resolution limitations, making it a growing focus in digital character research. This section discusses methods for representing 3D models, starting with explicit representation methods, including geometric and physical methods, and then moving on to implicit representation methods.

2.1.1 Explicit Representation Methods

Explicit representation methods describe a 3D scene by assembling basic elements such as points, meshes, and voxels. These methods provide users with detailed and precise visualization of various environments and objects, as they can be directly defined and observed. In the following sections, 3D modelling methods based on these three basic primitives will be introduced separately.

2.1.1.1 Point Clouds

A point cloud is defined as a collection of numerous points in 3D space, which contains all the attribute information of geometries, including colour and normal. Otepka et al. (2013) indicated that the term "cloud" denotes the disordered characteristic of the group and its spatial consistency, featuring a vague edge. The point cloud dataset, obtained by scanning instruments, comprises points sampled from the surface of an object. As shown in Fig. 2.1, Fig. 2.1(a) is the point cloud model of the "Stanford Bunny", and Fig. 2.1(b) is the "Stanford Bunny" model.
Raw point cloud data can be acquired using various methods, such as LiDAR laser scanning, RGB-D cameras, and multi-sensor fusion. Among these, point cloud data obtained through multi-sensor fusion is the most comprehensive and reliable.



(a) (b)

(c)

Figure 2.1: (a) The point cloud model of "Stanford Bunny"; (b) "Stanford Bunny"; (c) The voxel model of "Stanford Bunny"

The reconstruction of a 3D model from the point cloud model is usually associated with different deep-learning techniques. Depending on the number of points in the point cloud dataset, it can be determined as sparse and dense. The sparse point cloud reconstruction involves taking multiple perspective images of the input scene or object. These images are processed to find keypoints using techniques like SIFT (Lowe 2004) or SURF (Bay et al. 2006), which are then matched up across images. After acquiring the initial camera parameters either from manually calibrating or from images, the camera poses and sparse point clouds can be calculated through Structure from Motion (SFM) (Ullman 1979). Then utilizing the Levenberg-Marquardt (LM) algorithm, more accurate 3D coordinates of points can be gained to reconstruct objects. However, SFM outputs can be influenced by factors like lighting and distance in images. To address this, David Lowe proposed the SIFT algorithm (Lowe

2004), which improves accuracy by finding scale and rotation invariant features.

The dense point cloud reconstruction involves taking the sparse point cloud obtained from the SFM (Structure from Motion) algorithm and using a series of global perspective images to create a depth map. This depth map assigns a depth value to each pixel, which is then projected into 3D space to form a dense point cloud. Calculating and generating the depth map is crucial and challenging in dense point cloud reconstruction. Bleyer et al. (2011) applied the stereo-matching algorithm PatchMatch on computing the depth map. Though this method yields good pixel values, it struggles with objects or scenes lacking surface texture due to partial information loss during computation. Huang et al. (2018) proposed DeepMVS, which comprises a network consisting of the patch match matching network, the intra-volume feature aggregation network, and the inter-volume feature aggregation network. This approach effectively addresses the issue of handling scenes with missing texture surfaces by generating a disparity map from the depth map. Nevertheless, it comes at the cost of slow processing speed. Considering the limitations of traditional point cloud reconstruction methods, such as scene loss and susceptibility to environmental factors, dense point cloud reconstruction algorithms based on deep learning have emerged. Yao et al. (2019) introduced R-MVSNet, which transforms the depth map onto different depth planes and combines the probability distribution of pixels on these planes to obtain an initial depth map. R-MVSNet offers fast processing speed and wide applicability, and no limitations with any input image.

2.1.1.2 Voxels

A voxel, short for "volume pixel element," is like a pixel but in three-dimensional space. It's a tiny element, most often a cube, that represents a point in the 3D world, much like how a pixel represents a point in a 2D image. Voxel-based 3D modelling methods divide 3D space into a series of adjacent but non-overlapping voxels, which represent the surface and internal shape of objects, as shown in Fig. 2.1(c). Common voxel-based modelling methods include 3D grid, Tetrahedral Network(TEN), Octree modelling, Triangular Prism (TP) modelling, and Constructive Solid Geometry (CSG). Compared to traditional 3D modelling methods,

voxel-based approaches provide unique advantages in robustness and reconstruction quality. However, they require higher storage consumption, and the computational complexity of extending 2D image convolutions to 3D images is significant.

Wu et al. (2015b) introduced a neural network based on voxel representation and deep learning, utilizing regression loss functions. This approach marks a significant breakthrough in 3D reconstruction using deep learning. Building upon it, Tatarchenko et al. (2017) proposed a method for segmenting 3D space, recursively dividing it into eight octant spaces, to predict higher-resolution shapes on voxel grids. Stutz and Geiger (2018) pioneered an unsupervised learning approach to 3D reconstruction. This method completes voxel models of objects without sacrificing accuracy. It relies on data training and linear optimization of deep neural networks (DNN). Wu et al. (2016) devised a 3D reconstruction method using Generative Adversarial Networks (GAN). By employing a generator G and a discriminator D to evaluate and train input images, this method generates random 3D models, ultimately yielding the most accurate model after iterations. The voxel models generated by this approach boast advantages such as high resolution and detailed structure.

Whilst voxel-based surface creation methods have their unique advantages in 3D modelling, they are limited by issues such as memory consumption and storage usage. In recent years, research on 3D reconstruction based on deep learning has shifted more towards neural networks that operate on point clouds and surface meshes. PointNet (Qi et al. 2017) has largely addressed the challenges associated with voxel-based surface representation in deep learning. Other notable methods utilizing voxel representation such as NeRF (Mildenhall et al. 2021) and 3D Gaussian splitting (Kerbl et al. 2023) play important roles in generative contents. However, most training and output results in 3D reconstruction based on deep learning now primarily rely on point clouds and surface meshes.

2.1.1.3 Meshes

By linking numerous vertices with edges, meshes can be formed (Botsch et al. 2010). They can then be further redefined by utilizing polygons, commonly triangles or quadrilaterals, to generate authentic depictions

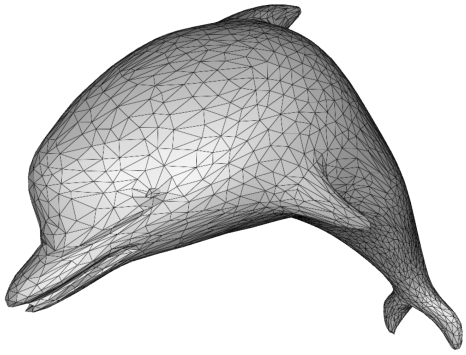


Figure 2.2: The polygon mesh model of a dolphin.

of objects (Shirman and Sequin 1987). Thus, 3D models consisting of surface meshes are polygon mesh models, like the dolphin model in Fig. 2.2. Polygon meshes consist of elements, including vertices, edges, faces, polygons, and surfaces. They offer a flexible and effective method for illustrating complex shapes and formations, as they can be easily modified and processed by computer algorithms, and can easily be visualized by business software. This type of model representation approach is essential for any pipeline involved in digital content creation (DCC), given its widespread adoption and interoperability.

Botsch et al. (2010) highlighted that the primary approaches for polygon mesh surface recreation are parametric representation methods. This kind of approach offers the benefit of the function $\mathbf{f} : \Omega \to \mathcal{S}$ reducing many 3D surface problems on the surface $\mathcal{S}$ to 2D problems in the parameter domain Ω. However, creating the parametric surface parametrization $\mathbf{f}$ can be quite complicated, as the parameter domain Ω must align with the topological and metric characteristics of the surface $\mathcal{S}$. Adjusting the shape of $\mathcal{S}$ may necessitate updating the parametrization to accurately reflect changes in the underlying geometry. Therefore, the challenges of parametric surfaces lie in their vulnerability to topological modification and spatial queries. The following part of this section will review various crucial techniques for parametric surface representation, including spline surfaces, subdivision surfaces, and Ordinary Differential Equations(ODEs)-based surfaces.

Hoschek and Lasser (1993), Prautzsch et al. (2002), Patrikalakis and Maekawa (2002) introduced surface creation techniques involving Bézier and B-spline. Bézier curves are essentially the result of linear interpolation as they are created by interpolating between pairs of control points

to form a smooth curve. Combining multiple Bézier curves allows to obtain a set of corresponding points on each curve at the same parameter position. By combining these newly sampled points and controlling them in a similar way to Bézier curves, a smooth surface can be formed by a set of smooth curves, which is defined as a Bézier surface. Due to the principles of Bézier surfaces, two adjacent Bézier surfaces naturally exhibit good G^0 continuity. This feature allows users to create a range of polygon models with smooth surfaces. For example, a polygon model of a broken blade was created utilizing Bézier surfaces by Li et al. (2010). However, Bézier surfaces possess a limitation: they cannot handle local adjustments. Altering control points locally impacts the entire shape of the model formed by all Bézier surfaces. Furthermore, the polynomial order of Bézier surfaces correlates with the number of vertices, which restricts the flexibility in manipulating the entire surface. A higher number of vertices results in a higher polynomial order, thereby weakening control over the curve's shape.

Therefore, to address the limitations of Bézier surfaces, B-spline surface creation methods were developed, allowing for local adjustments to the surface. Moreover, B-spline surfaces offer the advantage of decoupling the polynomial degree from the number of control points, and a lower degree can result in closer adherence of the B-spline curve to the control polyline. However, both Bézier and B-spline surfaces have defeats, such as the inability to accurately represent complex shapes like conics. Nevertheless, non-uniform rational B-splines (NURBS), as introduced in (Piegl and Tiller 1997), avoid these limitations. NURBS have become widely adopted in various software packages for geometric modelling and have become the standard surface representation in modern CAD systems due to their versatility, inherited advantages of B-spline surfaces, and alignment with international standards. As the name suggests, NURBS extends the basis functions of B-spline curves to rational functions and assigns a weight to each control point, making them non-uniform. Hence, NURBS can precisely define freeform surfaces without relying on solid models, while Bézier and B-spline surfaces are all particular cases of NURBS.

However, to ensure that adjacent patches of the model can all be smoothly connected, extra geometric constraints need to be considered

at all stages of surface processing. This significantly increases the complexity of NURBS surface reconstruction. Besides, NURBS functions rely on the heavy utilization of polynomials, whilst the solutions to ODEs and Partial Differential Equations (PDEs) may involve more intricate mathematical functions. Consequently, a single Bézier, B-spline, or NURBS patch may not match the capabilities of a single ODE or PDE patch in crafting complex shapes.

The process of subdivision approaches, as described in (Stam 1998, DeRose et al. 2023, Warren and Weimer 2001), initiates with a basic polygonal model. The polygonal faces of this model are then subdivided into smaller polygons through various approximation or interpolation techniques, resulting in a denser polygonal mesh of the model. Subdivision surfaces offer flexibility without being constrained by topological or geometric limitations, unlike spline surfaces. Their hierarchical structure enables the use of highly effective algorithms. Thus, Subdivision surfaces can simplify the creation of intricate shapes and enhance rendering efficiency. However, subdivision methods are limited to creating meshes with semi-regular subdivision connections, which means that the mesh triangulations result from repetitive uniform refinement of a coarse control mesh. Since arbitrary meshes do not meet this constraint, they must undergo remeshing to meet subdivision connectivity requirements in a preprocessing stage. However, this remeshing process involves resampling the surface, often leading to sampling artefacts and loss of data. Moreover, subdivision methods face challenges in achieving precision and the absence of an underlying parametrization.

Compared with the above geometric approaches, physics-based surfaces have a better capacity to create more realistic appearances as they consider the underlying physics of surface deformation. Nealen et al. (2006) reviewed different surface creation methods based on physics. These approaches encompass finite element method (FEM), finite difference method (FDM), and finite volume method (FVM). However, these common physics-based methods face challenges in numerical computation, preventing their implementation in real-time environments. In contrast, ODE/PDE-based surface creation approaches, which also adhere to the underlying physics principles in science and engineering, were introduced to address the issue of data size since the coordinates of the vertices on the surface obey certain ODEs/PDEs. For example, the

deformations of an elastic beam can be represented by a fourth-order ODE, according to (Timoshenko 1983). You et al. (2007) first proposed the novel surface creation method, ODE-based sweeping surfaces, which utilize minimal data to analytically represent a surface. This method allows quick computation with stable numeric. ODE-based surface blending was then provided in (You et al. 2014), which allows the creator to craft blending surfaces to be formed using the analytical solutions to ODEs. Its form is adjusted by the shape control parameters associated with the ODE. Though ODEs are simpler to solve compared with PDEs, PDE-based surface creation methods are more capable of representing extremely complex models.

Bloor and Wilson (1989) laid the foundation for PDE surface creation in computer graphics. Since then, various methods based on PDEs have emerged to address different modelling challenges, including surface design (Ugail et al. 1999), solid modelling (Ahmat et al. 2011), high-speed train head optimization (Wang et al. 2021b), and surface modelling (Bloor and Wilson 1990, Sheng et al. 2010). Wang et al. (2019) have proved that PDE-based modelling methods can create smooth surfaces and can generate surfaces by adjusting a small set of boundary conditions Wang et al. (2021c), simplifying the modelling process. Thus, by transforming geometric problems into boundary value PDE problems, the PDE surface creation method can produce natural, continuous, and smooth surfaces without the need for manual patch stitching, as shown in Fig. 2.3.

Whilst extensive research has focused on utilizing continuous PDE surfaces for modelling and exploring PDE surface-based reconstruction, limited attention has been given to dynamic PDE-based modelling. Thus, integrating PDE surface creation methods with advanced skin deformation methods to fulfil real-time 3D modelling for digital characters remains a challenging task. In Chapter 4, an advanced PDE-based modelling method that reconstructs dynamic 3D models, presented by this research, will be introduced.

2.1.2 Implicit Representation Methods

Implicit representation methods refer to mathematical representations that do not provide specific point information but instead describe the

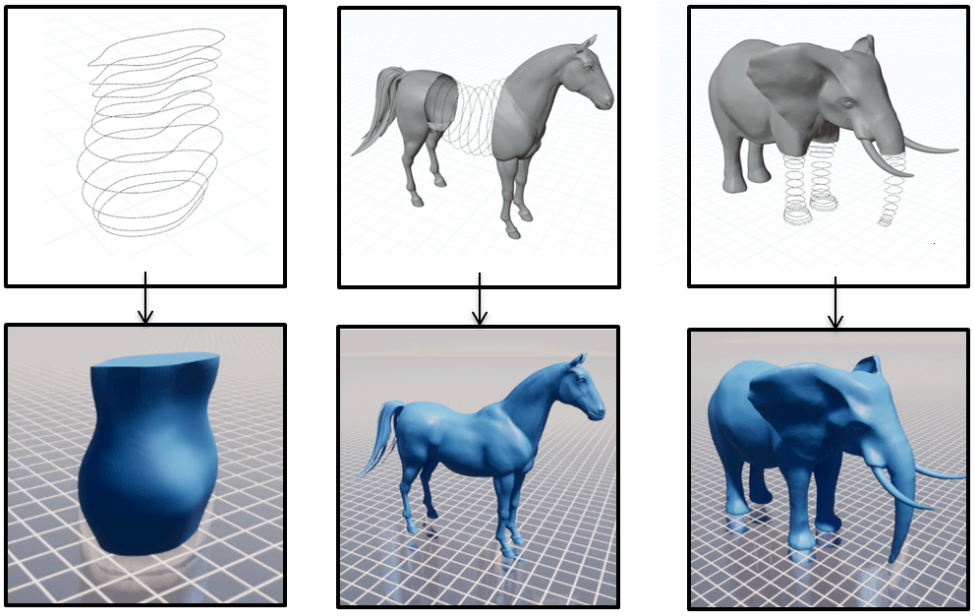
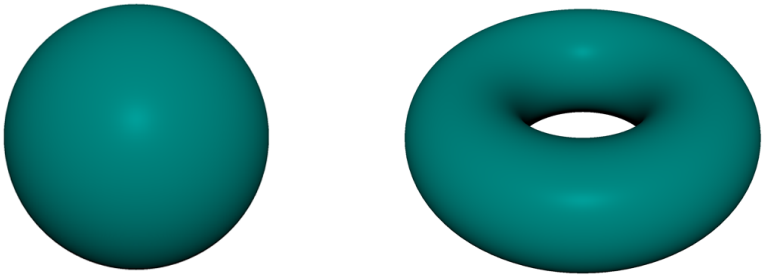


Figure 2.3: Surfaces created by the PDE-based modelling technique (Fu et al. 2022).

relationship satisfied by all points on the surface. With implicit equations, since no point information is given, sampling specific points on the surface becomes a challenging task. In fact, even the visualization of the surface shape directly from the equation can be difficult. However, using implicit surface equations makes it straightforward to determine a point's relationship with the object's surface, including whether it is inside the object, outside it, or exactly on the surface. Moreover, it enables easy determination of whether a ray intersects with the object. Various implicit representation methods have been developed, which will be reviewed in this section.

2.1.2.1 Algebraic Surfaces

Algebraic functions represent surfaces through algebraic expressions satisfied by every point on the surface. Bajaj (1988) noted that the common early methods of algebraic surface expressions were primarily characterized by 3D implicit curves, like Space Curves (Bose 1995) and Rational Curves (Abhyankar 1973). Some simple examples of algebraic surfaces can be seen in Fig. 2.4. However, it seems that surfaces represented solely by algebraic equations often exhibit regularity, which makes algebraic surfaces struggle to accurately represent more complex geometric shapes.



$x^2+y^2+z^2=1$ $(R-\sqrt{x^2+y^2})^2+z^2=r^2$

Figure 2.4: Two simple examples of algebraic surfaces: the left one is a sphere, and the right is an annulus.

2.1.2.2 Constructive Solid Geometry

Constructive Solid Geometry (CSG) essentially employs logical operators, such as Boolean operations, to combine different objects into complex surfaces or objects, as shown in Fig. 2.5. Hence it allows for the construction or representation of highly complex models or surfaces using simple primitives, such as cubes, cylinders, prism, pyramids, spheres, and cones. Its emergence has, to some extent, alleviated the challenge of representing complex objects using algebraic implicit surfaces. A comprehensive introduction to CSG technology was introduced in (Ghali 2008).

2.1.2.3 Signed Distance Function

Signed Distance Field (SDF) has been roughly introduced in (Oleynikova et al. 2016), which is fundamentally a grid-based representation method. The core idea of SDF involves partitioning space into a grid of points and then storing the distance from each point to the model. This effectively delineates a surface around the model, where points outside the model's surface have values greater than 0, points inside have values less than 0, and the value 0 denotes points on the surface. Fig. 2.6 shows an example in 2D space.

Algebraic operations can be performed on the Signed Distance Field (SDF) values of each point in space to blend them into different new models, as shown in Fig. 2.7. There's no need to recalculate the algebraic expressions of the object models' surfaces. SDF is widely used

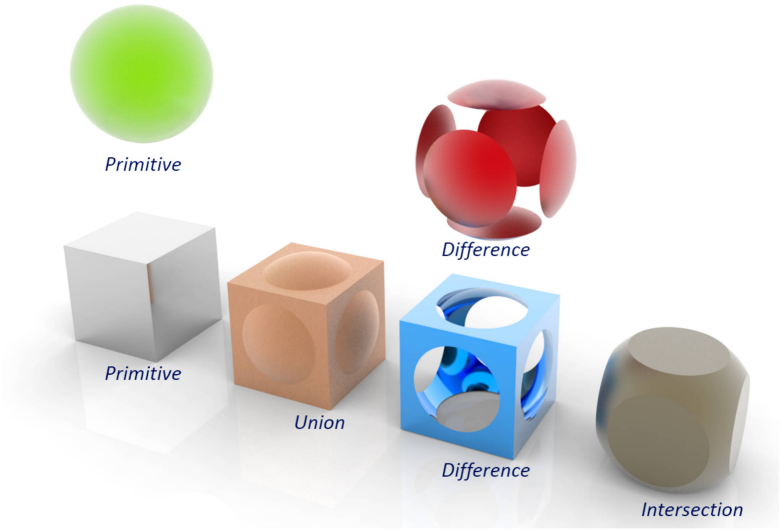


Figure 2.5: A complex object constructed from two simple primitives, a sphere and a cube, by Boolean operations (Renno and Papa 2015).

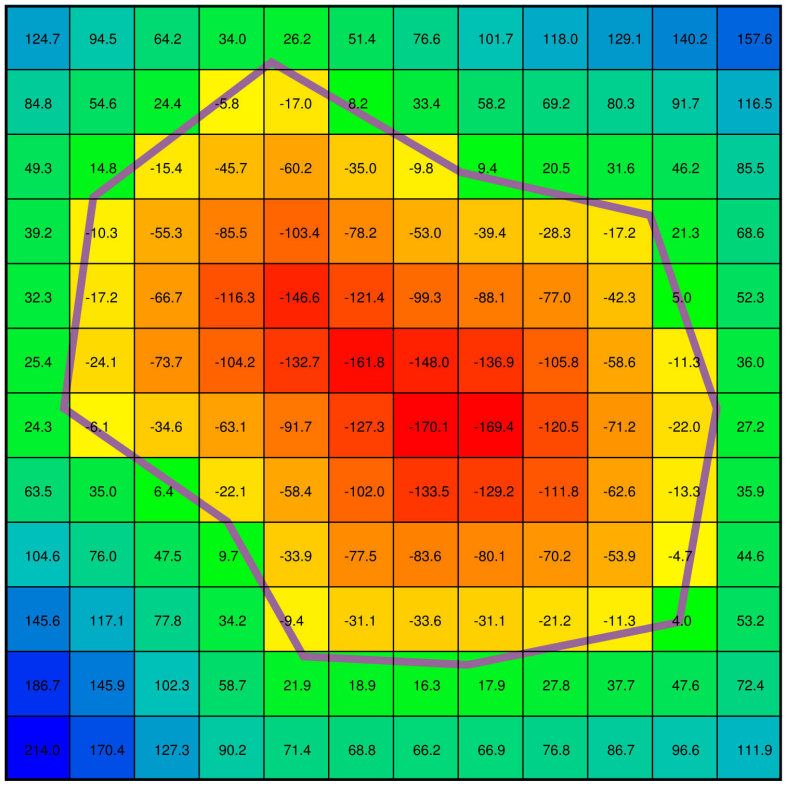


Figure 2.6: A 2D example of SDF. Each value of the grid stands for the nearest distance from the point to the surface (Zucker et al. 2010).

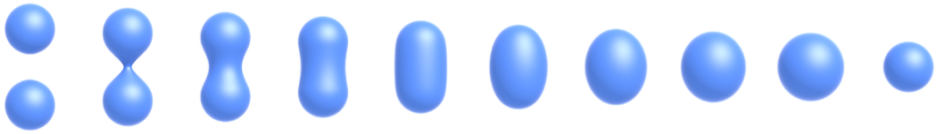


Figure 2.7: Apply algebraic operations on two SDF models to blend new and complex model shapes.

in algorithms for real-time representation of complex objects. Its simple representation aligns well with deep learning techniques, enabling rapid and accurate characterization of model features. For example, DeepSDF demonstrates superior performance in both shape representation and completion tasks by discretizing SDF into a regular grid for evaluation and measurement denoising (Park et al. 2019). It effectively handles the objectives of representing intricate structures and closed surfaces and delivering high-quality surface normals for shapes. Söderlund et al. (2022) integrated SDF grids into ray tracing methods to generate smoother and higher-quality images.

2.1.2.4 Level Set

Osher and Sethian (1988) first developed the level set method, which provides an implicit representation for evolving curves and surfaces. It is similar to the SDF approaches, functioning as a specialized form of SDF, which also identifies the locations where the function value is zero to define a surface. However, unlike SDF, which provides a strict mathematical definition for every point in space, the level set method approximates the function using a grid. Bilinear interpolation within these grid cells is used to determine the function value at any point, and all points where the function equals zero form the surface. This method has the advantage of more explicitly defining the shape of space curves compared to SDF. It has been widely used in medical imaging and physical simulations (Allaire et al. 2004, Fu et al. 2018, Li et al. 2007, Zhang et al. 2008).

2.1.2.5 Neural Radiance Fields

Neural Radiance Fields (NeRF) (Mildenhall et al. 2021) have emerged as a preferred method for representing scenes across various applications. NeRFs introduce an innovative approach to 3D scene representation,

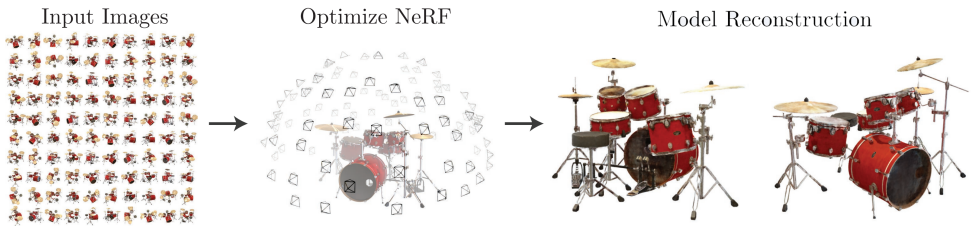


Figure 2.8: The main procedure of reconstructing 3D model from input images of different views by neural radiance fields (Mildenhall et al. 2021).

moving away from traditional point clouds and meshes to depict the scene as a continuous volume. The core idea of NeRF is to represent a 3D scene by querying an implicit neural network using a radiance field, which encodes the volume density and colour of every point from each camera viewpoint. This method provides a more flexible and adaptable means of capturing the complexities of 3D scenes, enabling advanced rendering and modelling techniques, as shown in Fig. 2.8. NeRFs have been further developed and optimized for better synthetic results (Barron et al. 2021, Verbin et al. 2022, Zhang et al. 2022, Insafutdinov et al. 2022, Yang et al. 2022), faster training and inferring (Nguyen-Phuoc et al. 2022, Yu et al. 2021, Garbin et al. 2021, Reiser et al. 2021, Liu et al. 2020a, Müller et al. 2022, Deng et al. 2022, Wei et al. 2021, Xu et al. 2022, Fridovich-Keil et al. 2022, Sun et al. 2022, Chen et al. 2022, Wang et al. 2021a), boundless and low-light scene (Martin-Brualla et al. 2021, Zhang et al. 2020, Niemeyer and Geiger 2021, Yang et al. 2021), and pose estimation (Yen-Chen et al. 2021, Wang et al. 2021d, Lin et al. 2021). Besides, NeRFs have been applied to many fields, including city reconstruction (Rematas et al. 2022, Turki et al. 2022, Tancik et al. 2022), and the processing of human faces and avatars (Park et al. 2021a b, Peng et al. 2021a).

2.2 Overview of Skin Deformation Methods

In the pipeline of generating realistic digital characters in real-time environments, after the construction of 3D models of digital characters, the next stage is to consider the deformation of model surfaces to enhance the experience of users in the "Metaverse" when models interact with the environment. However, Creating realistic and compelling skin deformations for digital characters is a multidisciplinary challenge, encompassing

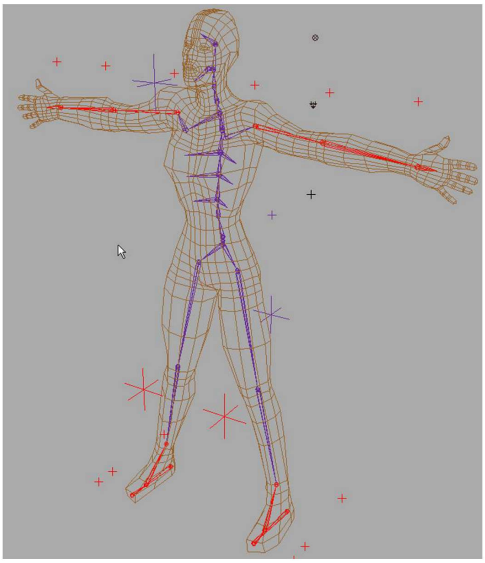


Figure 2.9: An example of geometric modelling (Singh and Kokkevis 2000).

three primary aspects: generating high-quality skin deformations, simulating skin contact in response to collisions, and producing secondary motion effects such as flesh jiggling during movement. This section will review existing techniques for simulating skin deformation, which can be roughly classified as geometric skinning, example-based skinning, and physics-based skinning approaches.

2.2.1 Geometric Skinning

Geometric skinning techniques define skeleton-to-skin binding directly through geometrical ways and relate the surface shape changes only with the movement of skeletons, as shown in Fig. 2.9. These geometric approaches to deforming articulated characters have demonstrated satisfactory results at interactive rates.

The traditional way to compute skin deformations is by linearly blending the corresponding bone transformations of the skin, called Linear Blend Skinning (LBS) (Magnenat et al. 1988). However, this simple linear blending approach causes artefacts when capturing complex deformations, including candy-wrapper effects (Fig. 2.10), collapsing elbow, and failure of secondary deformation (Lewis et al. 2023). These artefacts can be effectively eliminated by replacing linear blending with nonlinear blending, which converts affine rigid transformation matrices into pairs consisting of quaternion and translation.

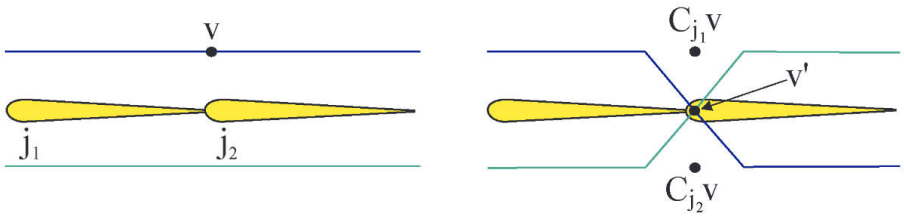


Figure 2.10: The left is the reference pose, whilst the right is a typical example of "candy-wrapper" artefact, where **v** denotes the vertex attached to the joints $j_1, j_2, \ldots, j_n$ (n stands for the number of influencing joints); $C_j \in \mathbb{SE}(3)$ denotes the transformation matrix from the rest pose of joint j_n to its actual position (Kavan et al. 2008).

Compared with the matrix counterparts, these pairs are easier to blend. Kavan and Žára (2005) employs a computationally intensive method, Singular Value Decomposition (SVD), which has limited practical impact due to their handling of the translational component of skinning transformations. Conversely, dual quaternion skinning (DQS) (Kavan et al. 2007) utilizes an approximate blending technique based on dual quaternions. Rigid transformations are consistently provided by DQS, which is nearly as fast as LBS. The underlying mathematics of DQS may be complex, whilst its implementation is relatively straightforward. Instead of matrices, it leverages the geometric algebra of quaternions to represent the rigid translations of the bones, which generalize regular quaternions to express both translation and rotation. Another straightforward modification of the LBS was introduced in (Jacobson and Sorkine 2011), incorporating an additional scalar weight function for each bone. This method allows for stretching and twisting without altering the existing skeleton rig or bone weights. Vaillant et al. (2013) introduced a nonlinear skinning method to manage skin contact and muscle bulge issues, though it fails to handle deep self-intersections. Then, an improved method was provided to address this, which implements new composition operators to local self-contact between implicit surfaces and facilitate blending effects. Another advanced algorithm, presented by Zhao et al. (2018), utilizes polycube to normalize the shape of meshes instead of arbitrary shapes.

Nonlinear methods have been effective in addressing the issues of LBS, such as candy-wrapper effects. However, they come with their own set of problems that need to be addressed, like joint-bulging artefact.

Thus, whilst geometric shape deformation techniques offer advantages in terms of simplicity, efficiency, and controllability, they have limitations in correcting various artefacts to simulate realistic deformation.

2.2.2 Example-based Skinning

Unlike geometric methods, example-based skinning techniques, also known as data-driven approaches, enable more intricate skinning effects, including muscle bulges and wrinkles, as well as mitigating the artefacts associated with linear skinning methods. These techniques utilize a series of sculpted example poses, interpolating them to achieve the desired deformation. Lewis et al. (2000) proposed pose space deformation (PSD), which is one of the earliest example-based methods. It interpolates correction vectors among the example data by leveraging a radial basis function. In PSD, the pose space is defined as a set of degrees of freedom for a character's model, varying between example poses, while a specific pose denotes specific values of these degrees of freedom. PSD is a general term for a class of methods where example poses are interpolated based on a character's pose.

Sloan et al. (2001) presented a more advanced extension of PSD, which example poses distributed in an abstract space to interpolate an articulated character. This abstract space comprises dimensions representing the 3D character's intrinsic properties, including age and gender, as well as dimensions describing the configuration, such as the degree of elbow bend. Additionally, weighted pose space deformation (WPSD) (Kurihara and Miyata 2004) was proposed to significantly reduce the example size. WPSD can process large-scale deformations with high efficiency, whilst it cannot simulate detailed deformations even with more computation compared with PSD. Thus, PSD is more applied to animation industries with pre-computing results, like the movie industry, but is not proper for real-time interactive systems, like Metaverse.

This high computational cost problem is then addressed by advanced research. For instance, Kry et al. (2002) proposed EigenSkin, a method akin to PSD, which interpolates the rigid transformations by pre-computing the optimized centre of rotation for each vertex. This approach significantly reduces memory usage and allows the GPUs to manage computations. Despite the simplicity of PSD-based methods, substantial efforts

are demanded from artists to manually create a wide range of example poses for training.

Mohr and Gleicher (2003) presented single-weight enveloping (SWE), which utilizes the example data approximated through fitting the deformation parameters, which is another type of approach for skinning. SWE estimates the single weight for each vertex associated with rigid character bones, allowing for the addition of extra bones. Conversely, multi-weight enveloping (MWE) (Wang and Phillips 2002) offers better approximations compared with SWE by operating on a linear framework that supports multiple weights of each vertex-bone. However, this comes at the expense of using 12 weights per vertex bone instead of just one in SWE. These two approaches enable the generation of a wider range of deformations from a smaller set of poses by introducing additional weight parameters, albeit at the cost of increased complexity in weight computation. To reduce the computational cost of applying parameter fitting approaches in a real-time interactive system, Wang et al. (2007) proposed a novel method, which treats the enveloping problem as learning a mapping between skeletal poses and their corresponding mesh.

With the development of machine learning in recent years, this technique has been integrated into example-based skinning techniques. In (Bailey et al. 2018), mesh deformations were divided into linear and non-linear components. The linear deformation is determined by the underlying skeleton transformations of the mesh, while the non-linear deformation is approximated by utilizing deep learning approaches. RigNet (Xu et al. 2020) utilized machine learning to learn example data in a dataset. After learning, it can predict a skeleton to fit the input 3D model and estimate surface skin weights. Fig. 2.11 shows the network model of DeePSD (Bertiche et al. 2021), which integrates a neural model into PSD.

Example-based methods excel at producing realistic mesh deformation when there are enough high-quality example shapes available. However, the flip side of this advantage is that example-based techniques require a large number of these high-quality example shapes, which will contain numerous human involvement.

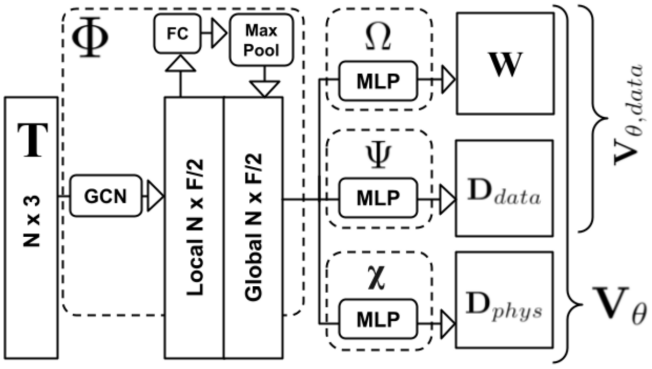


Figure 2.11: The neural model overview of DeePSD (Bertiche et al. 2021), which integrates neural network and machine learning to simulate deformation.

2.2.3 Physics-based Skinning

As geometric skinning approaches are not capable of accurately modelling dynamic skin deformations due to lacking underlying physical principles, animators traditionally configure deformation for each keyframe. This large amount of tedious manual work requires a high cost. Besides, example-based methods are also labour-intensive for artists, requiring enough high-quality example model data for training. Physics-based shape deformation methods tackle this problem as they integrate physics into the skinning process to significantly enhance the realism and credibility of character motions. They transfer this manual work to computers to compute the deformation, effectively saving labour costs. Following the foundational work of Terzopoulos et al. (1987) and Lasseter's animation principle "squash and stretch" (Lasseter 1998), physical simulation has become pivotal in the animated feature game and movie industry. Numerous physically-based methods have been developed to simulate the dynamic effects of the skin, reviewed in (Nealen et al. 2006).

Turner and Thalmann (1993) treated the fat layer of the character skin as a separate elastic surface, whilst the muscles are not modelled with deformable methods. Kavan and Sorkine (2012) skinned articulated shapes to simulate high-quality deformation results by optimizing skinning weights and incorporating joint-based deformers. Integrating FEM, Lee et al. (2009) proposed a sophisticated biomechanical model for simulating realistic flesh deformations. In (McAdams et al. 2011), a multigrid approach was presented to support hundreds of thousands of degrees of

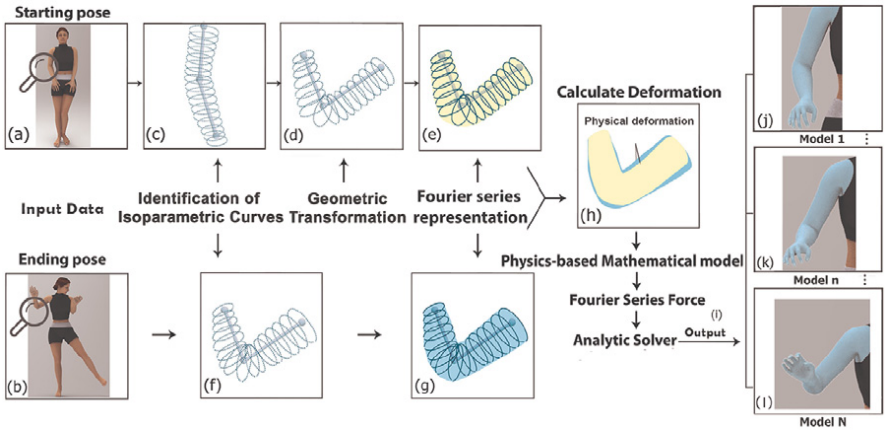


Figure 2.12: The process of applying underlying physics laws to model (Bian et al. 2019).

freedom (DOF), where the material can slide over modelled bones, providing more local details. Bender et al. (2013) introduced a multi-layer model to solve physical constraints, including collisions and local volume preservation, to simulate real skin effects of different tissues. Jacobson et al. (2012) combined the optimization of a nonlinear energy function with a deformable body to quickly simulate deformation with high efficiency. For achieving real-time performance, the computational cost of physics-based skin deformation methods has to be decreased. These works have addressed the requirements of interactive systems without losing accuracy or increasing implementation complexity (Teng et al. 2014, Li et al. 2014, Wang et al. 2015, Murai et al. 2017, Roussellet et al. 2018). In Chapter 5, a facial skin deformation method will be introduced by integrating Newton's second law into an ODE-based surface creation method, to generate natural facial expression change between specific poses.

Additional research has been conducted by integrating different skinning methods into physics-based approaches. Chaudhry et al. (2015) presented a new model of dynamic deformations integrating FDM and example-based methods to create realistic skin deformation with high efficiency. Xu and Barbič (2016) added PSD to physics-based methods to meet the real-time application. Based on that, Bian et al. (2019) combined example-based and physics-based skin deformation and LBS to develop an efficient deformation creation technique. The pipeline of this method is shown in Fig. 2.12, which applies a physics-based mathematical model to the model after geometric transformation.

With the involvement of underlying physical principles, physics-based methods are capable of generating authentic deformation results compared with geometric approaches, and avoiding numerous manual labours. However, as they transfer the labour from artists to computers, these methods rely heavily on computing power. The current hardware cannot deliver real-time performance for systems such as games and interactive animations due to these computationally intensive and complex methods. As a result, most developments focus on either enhancing the realism of geometric deformation methods or accelerating physics-based deformation methods.

2.2.4 Position Based Dynamics

The advent of Position Based Dynamics (PBD) (Müller et al. 2007) methods effectively balanced the trade-off between animation realism and computational efficiency, quickly becoming the most widely used deformation simulation method in the industry. They are fast, robust, and simple in simulating dynamic systems. These advantages allow them to quickly compute visually plausible deformation effects with minimal computational resources, effectively meeting real-time requirements.

The fundamental algorithm and some prior applications of PBD have been extensively reviewed in (Bender et al. 2014c 2017). PBD was first proposed in (Müller et al. 2007), with a non-physical system solving various constraints to generate visually authentic dynamic simulation. Whilst originally designed for interactive environments to simulate solid objects, position-based methods have been proven to be applicable to the simulation of fluids, articulated rigid bodies, and other scenarios.

However, due to their failure to comply with underlying physics laws, position-based approaches are only plausible in vision and suffer from issues such as the inability to converge to a specific solution. To expedite convergence, a multi-grid-based strategy (Georgii and Westermann 2006) was employed by Müller in Hierarchical Position Based Dynamics (HPBD) Müller (2008) to handle general non-linear constraints, enhancing the suitability of the simulation process for interactive applications like computer games. To model more complex physical phenomena, Bender et al. (2014b) proposed a continuum-based formulation and treated strain energy as a constraint function for the PBD solver. Additionally, previous PBD methods have long been plagued by a persistent issue:

constraints can become arbitrarily stiff depending on iteration count and time step. To tackle this problem, Macklin et al. (2016) introduced the XPBD method, which utilizes a new constraint formulation and the Lagrange multiplier to solve constraints in a time step and iteration count-independent manner.

Since the previous works of PBD before 2018 have been thoroughly introduced in (Bender et al. 2014c 2017), Chapter 3 will provide a comprehensive review of the algorithm, as well as the advancements and applications of position-based approaches after 2018.

2.3 Overview of Motion Reconstruction Methods

In addition to modelling and skin deformation, a crucial aspect of real-time digital character generation is motion construction, as the Metaverse relies on realistic and comprehensive motion animations to provide users with an immersive experience. Initially, motion construction involved artists manually modelling motion frame by frame. By assigning movements to the skeletons to drive their corresponding skins, complete digital character animations can be created. However, this manual process is labour-intensive and tedious, and hence the early animation creation requires high labour costs to ensure the relative authenticity. To reduce human effort and cost, various methods, such as motion capture technology (Menolotto et al. 2020), have been developed. Fig. 2.13 is an example of the motion-capturing device. More recently, with the development and proliferation of deep learning, motion construction has evolved to the point where it can be reconstructed from 2D images or videos. Compared to other digital characters, creating motion for digital humans is the most complex and widely applicable task. Therefore, most research is focused on Human Pose Estimation (HPE), while this section will also focus on it and give a comprehensive overview of this challenging task.

Based on the dimensionality of human poses, the human pose reconstruction task can be divided into 2D HPE and 3D HPE. Numerous datasets such as FLIC (Sapp and Taskar 2013), MPII (Andriluka et al. 2014), and MSCOCO (Lin et al. 2014) have emerged, supporting various

Figure 2.13: An example of a motion-capturing device, consisting of 17 inertial and magnetic sensor modules (Roetenberg et al. 2009).

algorithm frameworks for single-person and multi-person pose estimation, greatly enhancing the performance of 2D HPE. The goal of 2D HPE is to locate and identify 2D keypoints of humans and connect these keypoints in a sequence to form the projected skeleton on a 2D plane. On the other hand, 3D HPE aims to predict the 3D coordinates and angles of human joints. The lack of annotated 3D pose datasets has led many research methods to build upon 2D pose estimation techniques. Consequently, advancements in 2D pose estimation have laid a solid foundation for 3D human pose estimation, which holds immense potential for future research. In practical applications, 3D HPE offers more precise representations of human poses compared to 2D HPE, as it incorporates depth information. This increased precision makes 3D pose estimation more valuable and applicable across a wider range of fields. However, 3D HPE is also more challenging due to issues such as occlusion, the inherent depth ambiguity in mapping from single-view 2D to 3D, and the lack of large-scale outdoor datasets. Given the significance of 3D human pose estimation, this section primarily focuses on summarizing the research progress in this area.

Existing research in 3D HPE methods can be broadly categorized into traditional methods and deep learning methods. Before the widespread

adoption of deep learning, 3D human pose annotation datasets and high-performance GPUs were not readily available. Researchers primarily relied on traditional computer vision and machine learning techniques to estimate 3D human poses. The key distinction between traditional and deep learning-based 3D pose estimation methods is the use of multi-layer neural networks in the latter. This difference in modelling approaches leads to significant variations in estimation accuracy and computational complexity. Modelling is a crucial aspect of 3D HPE, aiming to represent keypoints and features extracted from input data. Given the complexity of real-world environments, selecting appropriate and effective image features to simplify the modelling process is essential.

Traditional methods often use human body models to describe and infer poses by extracting image features. These methods demand high accuracy in feature representation and spatial relationships of keypoints. High-level features such as Scale Invariant Feature Transform (SIFT) (Lowe 2004) and Histogram of Oriented Gradients (HOG) (Dalal and Triggs 2005) are commonly used due to their robust representation capabilities and ability to compress feature space dimensions effectively. However, these traditional features, despite their time efficiency, are manually designed and have significant limitations. They may lose critical image details and suffer from occlusion and inherent geometric ambiguities, restricting their applicability. Moreover, traditional methods impose certain requirements on the collected image or video data, making them susceptible to factors like collection costs, occlusion, lighting, and environmental conditions, regardless of using monocular or multi-view cameras.

In contrast, deep learning models offer strong feature representation capabilities and automate feature extraction from input data, eliminating the need for manual feature design. They leverage convolutional neural networks (CNN) (LeCun et al. 1998) to train on image data, directly obtaining effective representation methods. CNN extracts rich semantic features from images, providing higher accuracy and robustness compared to manually crafted features. The representational power of these networks grows exponentially with increased network depth, enhancing the precision and robustness of pose estimation in complex environments. Despite significant progress in deep learning for human pose estimation,

challenges such as occlusion, insufficient training data, and depth ambiguity remain difficult to overcome.

2.3.1 Traditional Methods for 3D HPE

Traditional methods for 3D HPE can be broadly categorized into three types: generative methods, discriminative methods, and template matching methods. Template matching methods can be seen as a hybrid of the generative and discriminative approaches.

2.3.1.1 Generative Methods

In traditional methods, feature extraction and the Pictorial Structure Model (PSM) (Fischler and Elschlager 1973) play crucial roles in pose estimation. PSM views the human body as a collection of joint structures (Burenius et al. 2013), with spatial constraints between these joints, which are particularly advantageous for joint estimation of 3D poses.

When applying generative methods to 3D HPE, the primary task is to build a parametric human model (Zhang et al. 2021). By adjusting the model's parameters, different poses can be generated. Thus, HPE using generative methods can be framed as an optimization problem, where the goal is to minimize the difference between the generated model image and the actual image. The core of generative-based HPE lies in constructing the human model, selecting optimization functions and target functions, and performing searches in high-dimensional pose spaces to achieve accurate results. Methods for searching the pose parameter space of constructed human models include Iterative Closest Point (ICP) (Ganapathi et al. 2012), Gaussian Mixture Models (GMM) (Ye and Yang 2014), and Markov Chain Monte Carlo (MCMC) sampling (Brau and Jiang 2016). These methods, although accurate, are computationally intensive and complex, which hinders real-time performance. Moreover, the initialization parameters of the human model significantly affect the ability to find the optimal pose. Good initialization methods can reduce the time required for spatial searches of human parameters, thereby improving real-time performance. Conversely, poor initialization can extend the search time, greatly impacting the algorithm's efficiency and real-time applicability.

2.3.1.2 Discriminative Methods

To address the limitations of generative methods, researchers have developed discriminative methods for HPE. These methods offer several advantages, including not requiring pre-generated human pose models or initialization, and having faster computational speeds. Discriminative methods regard pose estimation as a regression problem. Initially, they use classification algorithms to identify body parts and mark the estimated human joint points through a dotting method. Clustering algorithms then determine the centres of these points, and a pre-trained regressor precisely estimates the positions of each key point. However, discriminative methods struggle with robustness when faced with insufficient samples or occlusions. Additionally, the quality of training data significantly impacts the accuracy of the final estimation, especially for discriminative methods that rely on labelled data.

Discriminative algorithms extract features from images and learn the mapping from the feature space to the pose space. Given the strong correlation between skeletal joints and joint positions, some methods have incorporated this dependency into their models. For instance, Ionescu et al. (2011) proposed a discriminative monocular 3D human pose reconstruction method based on latent segmentation inputs. This model can infer human poses from monocular images captured in complex environments. Shotton et al. (2011) trained a regression forest to cluster input depth images by body parts and used a mean-shift algorithm to estimate joint positions. Chang and Nam (2013) employed a random classification forest to identify which body part each visible pixel belongs to and used a random regression forest to estimate all human joint points. Building on this, Park et al. (2017)] introduced a random validation forest to eliminate interference caused by self-occlusion. Ramakrishna et al. (2014) used multiple hierarchical multiclassifier cascades to estimate joint positions, addressing occlusion to some extent. However, these predictive models still face challenges such as high model complexity, excessive parameters, and computational intensity.

2.3.1.3 Template Matching Methods

Template matching based on geometric priors is a key approach for skeleton keypoint detection. This method represents the human body by

keypoints, limb structures, and their spatial relationships. Effective template matching techniques can simulate a wide range of poses and improve pose detection accuracy. Historically, two main strategies have been used. The hybrid methods combine generative and discriminative approaches, as demonstrated by (Ganapathi et al. 2012), who used pretrained discriminative models to estimate body parts and initialized generative processes affected by fast motion or occlusion. Another strategy involved data-driven template matching for initial pose estimation, followed by generative methods for precise adjustment (Baak et al. 2013, Ye et al. 2011). However, these methods also face challenges, including the high cost of building and maintaining pose template libraries, the trade-off between template variety and search efficiency, and the difficulty in extracting highly discriminative features from raw data. As a result, little further research has been conducted using these methods.

2.3.2 Deep Learning-based Methods for 3D HPE

Deep learning methods have gained prominence in 3D human pose estimation due to their superior feature extraction capabilities compared to traditional manually designed feature methods. Unlike conventional approaches that require pre-defined feature extraction, deep learning leverages neural networks to automatically obtain high-level semantic features. Initially, CNN is used to extract image features, which are then processed to determine the positions of skeletal keypoints. This self-learning feature representation in deep learning outperforms traditional methods that rely on prior knowledge. Additionally, the transfer learning capabilities of deep learning allow models trained on large datasets to be effectively applied to smaller datasets. Consequently, deep learning-based 3D human pose estimation has become the mainstream research approach. This approach can be categorized into two main types: direct regression-based methods, and 3D pose estimation based on 2D information.

2.3.2.1 Direct Regression-based Methods

Direct regression-based HPE, also known as end-to-end HPE, uses a large network to process all data. Leveraging the ability of deep neural networks to fit complex functions, this method typically does not require additional algorithms or intermediate data. Thus, it directly predicts 3D

pose coordinates from a single image using a regression-based network structure. The primary advantage of this approach is its end-to-end training and inference, simplifying the application process. However, it demands a sophisticated network architecture and rigorous data preprocessing.

Li and Chan (2015) pioneered in using deep learning for 3D human pose estimation by training a network to directly regress 3D joint positions from images. Their method utilized a multi-task training framework with tasks divided into joint detection and regression. These tasks shared early feature layers, where the detection task classified whether a local window contained a specific joint, and the regression task calculated the relative position of joints to the root joint. The training method was unique as it first trained a separate visual task for object detection, and then used the CNN layers from the feature extraction part as the initial model for 3D HPE, discarding the object detection head and focusing on the regression task to achieve the final estimation results. Similarly, Park et al. (2016) proposed a network structure with an additional supervision branch for 2D pose estimation. This approach used 2D pose estimation results concatenated with image features to estimate 3D poses, incorporating relative position information from multiple joints, not just the root joint, for more accurate 3D poses. Tekin et al. (2016) pre-trained an unsupervised autoencoder to learn the mapping from 3D poses to a high-dimensional latent space, encoding structural dependencies between joints to strengthen pose constraints. They then used a shallow network to learn high-dimensional pose representations. By utilizing multi-step outputs of the encoder, rather than a single fixed-length vector, they preserved more information.

Heatmap regression retains more information from the image, and using heatmaps of skeletal keypoints is a mainstream method in 2D human pose estimation that can also be applied to 3D pose estimation. Tekin et al. (2017) and Zhou et al. (2019a) utilized 2D heatmaps as intermediate representations for estimating 3D poses instead of 2D joint coordinates. Pavlakos et al. (2017) extended the use of skeletal keypoint heatmaps and the Stacked Hourglass Network (SHN) (Newell et al. 2016) from 2D pose estimation to 3D. Considering the wide range of Z-axis depth, they proposed a coarse-to-fine structure for gradual regression. For each joint, each stage generates heatmaps with different channel numbers to

progressively refine the Z-axis resolution, thus forming 3D heatmaps. By calculating the confidence of each point, the 3D joint positions can be inferred in a discrete 3D space, improving the accuracy through iterative refinement from coarse to fine estimates. However, this method has limitations. When obtaining joint coordinates, taking the maximum value position from the heatmap and converting it back to the original image space introduces quantization errors, leading to significant deviations in the final coordinates. Additionally, the non-differentiable nature of the max operation prevents end-to-end training and optimization of the model. To address these issues, Zhou et al. (2016) shifted their approach by detecting body parts instead of directly estimating joint positions. They embedded kinematic models directly into the deep neural network to estimate general joint movements, thus enhancing the robustness and accuracy of 3D human pose estimation.

2.3.2.2 2D Information-based Methods

To address the limitations of direct regression methods in model optimization and practical application, researchers have explored 3D human pose estimation based on 2D information. These approaches effectively mitigate the issue of mismatched labelled data quantity and network size encountered in direct regression methods. 3D pose estimation based on 2D information generally involves two stages: first, acquiring 2D information, and then predicting 3D pose coordinates from the 2D pose. There are two main implementation strategies: one integrates the training of both 2D and 3D pose networks, while the other uses a pre-trained 2D pose network and inputs the resulting 2D pose into a 3D pose estimation network for dimensionality enhancement. The latter, also known as 3D pose estimation based on 2D skeleton sequences, reduces the overall task complexity. This method allows the network to more easily learn the 2D-to-3D mapping and benefits from the maturity of 2D pose estimation techniques. Additionally, it facilitates the incorporation of reprojection for semi-supervised learning, making it a more mainstream approach.

Joint training of 2D and 3D pose networks represents an alternative to directly regressing 3D coordinates from images. This method uses the 2D information obtained from the network as an intermediate representation for further predicting 3D coordinates. Chen and Ramanan (2017) introduced a method based on 2D pose estimation and pose matching.

They retrieve and compare the 2D pose obtained from images with the 2D projections of poses from a large 3D human pose library, ultimately outputting the best-matching 3D pose. This method has the advantage of avoiding the need for complex human structure constraints, but it requires a large amount of data to ensure high accuracy. Moreno-Noguer (2017) inferred 3D human poses through distance matrix regression. This approach encodes the pairwise distances of 2D and 3D body joints into two Euclidean Distance Matrices (EDM). EDM is invariant to rotation and translation within the image plane and maintains scale invariance with normalization. Wang et al. (2018) utilized a pairwise ranking convolutional neural network to predict the depth order of human joints. They then employed a coarse-to-fine pose estimator to perform 3D pose regression using the 2D joints and depth ranking matrix. Li and Lee (2019) treated 3D human pose estimation as an inverse problem with multiple feasible solutions. They first generated multiple 3D pose hypotheses and then used a ranking network to select the best 3D pose based on its 2D projection. This method allows for the refinement of 3D pose estimation by considering various potential poses and selecting the most accurate one.

A prominent method based on 2D skeletal sequences for 3D pose estimation was proposed by Martinez et al. (2017). This approach starts with 2D human pose estimates and uses a simple, shallow neural network to regress these 2D poses into 3D human poses, achieving high accuracy. This straightforward, fast, and lightweight baseline effectively maps 2D poses to 3D poses and demonstrates that most errors in 3D pose estimation stem from inaccuracies in 2D pose estimation and the mapping from 2D to 3D joints. Pavllo et al. (2019) showed that 3D poses in videos can be accurately predicted using a dilated temporal convolutional model based on 2D joints. They introduced back-projection and a semi-supervised training method with unlabelled video data, providing superior accuracy, simplicity, and efficiency compared to recurrent neural network (RNN) methods (Hossain and Little 2018, Lee et al. 2018). This approach also outperformed previous methods in scenarios with limited labelled data. However, it assumes temporal independence of prediction errors, which may not hold when occlusions occur. Chen et al. (2020) adopted an iterative strategy to directly match 2D inputs from multiple cameras with 3D poses, iteratively updating the 3D pose. Nevertheless,

since this method is linear in time complexity, its runtime increases significantly with the number of cameras. Remelli et al. (2020) proposed a lightweight solution by encoding each view's image into a unified latent representation, separating feature mapping from the camera's viewpoint. They used a learned camera projection operator to produce accurate per-view 2D detections, which were then elevated to 3D using GPU-accelerated direct linear transformation. To enhance the generalization of multi-view feature fusion, Xie et al. (2020) introduced a pre-trained multi-view fusion model, MetaFuse, which effectively adapts to new camera settings with minimal labelled data by learning from many cameras through a meta-learning framework, maximizing adaptability to various camera poses.

Overall, joint training of 2D and 3D pose networks requires complex structures and ample training data. In contrast, 3D pose estimation based on 2D skeletal sequences benefits from mature 2D pose estimation methods, offering simple, lightweight network structures, and fast training speeds. This makes it a mainstream approach in current 3D HPE research and in Chapter 6, this research will introduce a novel deep learning-based method, which estimates high-quality 3D skeletal motion from 2D inputs.

To sum up, this chapter has covered various related works. With the insights gained and goals outlined, the following chapters will delve into my endeavours towards creating advanced animation technologies for generating realistic digital characters in real-time environments.

Chapter 3

Review on State-of-the-art Improvements & Applications of PBD

3.1 Introduction

As discussed in Chapter 1 and 2, dynamic simulation of 3D models is crucial for creating immersive experiences in the metaverse, making it a hot research topic in computer animation. Numerous works have advanced simulation technologies, with reviews by Bender et al. (2014a) and Gibson and Mirtich (1997) covering these developments. However, achieving real-time realistic animations remains a critical performance challenge. According to the above content, this issue can be addressed through shape deformation methods, which fall into three main categories: geometric deformation, example-based deformation, and physics-based deformation methods. Geometric methods modify 3D model shapes by manipulating underlying control structures in different ways. For instance, Linear Blend Skinning (LBS) (Magnenat et al. 1988) blends the influence of multiple joints on each vertex based on assigned weights, Free-form deformation (FFDs) (Sederberg and Parry 1986) embed the 3D model within a lattice structure, and joint-based methods (Kavan and Žára 2005, Yang et al. 2006) apply transformations to a hierarchical structure of joints or bones. These methods work directly on positions and do not involve any physical surface deformation, often resulting in animations that appear unnatural. Example-based methods mitigate the artefacts of geometric methods to generate intricate deformation effects by learning from well-sculpted example poses, such as PSD (Lewis et al. 2000), SWE (Mohr and Gleicher 2003) and RigNet (Xu et al. 2020).

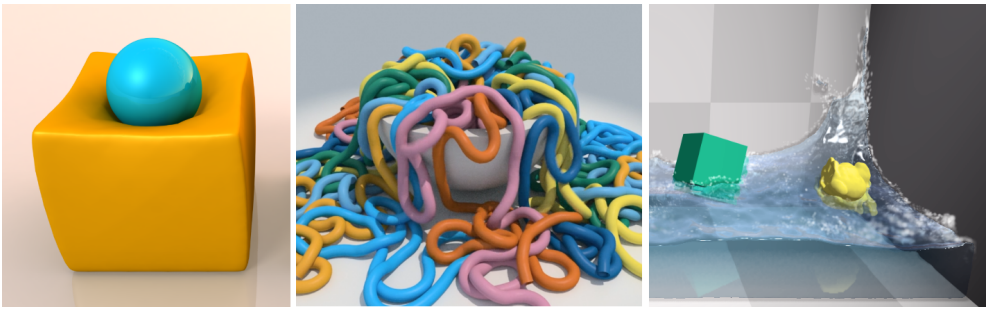


Figure 3.1: Some examples of PBD simulated scenes. The left is a solid ball collided with a deformable soft cube; the middle is a scene with many twisted bars; the right shows the simulation of fluids (Bender et al. 2017).

These methods require expensive human involvement in creating examples of realistic mesh deformation. On the other hand, physical methods rely on physical laws like Newton's second law to calculate forces and accelerations, updating corresponding velocities through time integration. Techniques in this category include impulse-based methods (Mirtich 1996), mass-spring systems (Chadwick et al. 1989), and FEM (Capell et al. 2005). Whilst physical methods can produce more realistic animations, they have drawbacks such as overshooting issues and high computational costs due to the need for velocity and acceleration layers and extensive numerical calculations. Consequently, they also fall short of the desired performance standards in real time.

Recently, position based dynamics (PBD) methods have become a focal point in interactive systems due to their effective balance between realism and efficiency. Compared to other approaches, PBD excels in speed, robustness, and simplicity, making it highly popular for simulating dynamic systems in real time. First introduced by Müller et al. (2007), PBD inherits the advantages of geometric methods by directly handling positions and omitting velocity and acceleration layers. These approaches focus on solving quasi-static problems, enabling great performance in real-time simulations and offering visually plausible deformations and good controllability. Some examples of simulation results produced by PBD frameworks are shown in Fig. 3.1.

The core algorithm and various applications of PBD have been comprehensively detailed in the paper (Bender et al. 2014c 2017). Originally, PBD methods were introduced for simulating interactive environments with solid objects. However, it was later demonstrated that these meth-

ods could also be effectively applied to simulating fluids, articulated rigid bodies, and other scenarios. In this chapter, recent advancements and applications of PBD in different scenarios will be reviewed. Previous works related to PBD up to 2018 have been comprehensively covered in (Bender et al. 2014c 2017). For clarity and readability, we will briefly mention part of these earlier works and focus more on the latest developments since 2018.

3.2 Basic Concept of PBD

The whole pipeline in simulating the deformation of a particle-based system by the original PBD is introduced in this section, according to (Müller et al. 2007).

3.2.1 Overview of PBD

A deformable object can be defined by a set of N vertices and M constraints in a particle-based system. The i^{th} ($t \in [1, 2, \ldots, N]$) vertex contains these attributes: mass m_i, position $\mathbf{x}_i$ ($\mathbf{x} = (x, y, z)$) and velocity $\mathbf{v}_i$ ($\mathbf{v} = (v_x, v_y, v_z)$). Applying the set of M constraints to alter the position and velocity attributes of the N vertices at the next timestep, a group of functions C_j ($j \in [1, 2, \ldots, M]$) can be computed. Thus, the position $\mathbf{x}_i^1$ and velocity $\mathbf{v}_i^1$ of the i^{th} vertex at the next timestep $t + \Delta t$ with its initial position $\mathbf{x}_i^0$ and velocity $\mathbf{v}_i^0$ at timestep t is calculated as follows:

In line (2), the current attributes of each vertex are used to initialize the state variables. Line (3) applies external forces, which cannot be considered positional constraints like gravity, to the system via a symplectic Euler integration step. Line (4) includes an optional damping step to improve simulation performance, while line (5) computes the initial position $\mathbf{p}_i$, which is used only as predictions. Line (6) generates M_{coll} nonpermanent collision constraints, which are distinct from the fixed constraints C_j ($j = 1, 2, \ldots, M$). Lines (7)–(10) use a solver to correct the predicted position $\mathbf{p}_i$. Line (11)-(12) recompute updated velocities and positions. Line (13) applies friction and restitution coefficients to modify the velocities of colliding vertices. Finally, the recomputed velocity and position are assigned to $\mathbf{x}_i^1$ and $\mathbf{v}_i^1$ at the next timestep $t + \Delta t$. Further explanations of the pipeline will be introduced in the following subsections.

Algorithm 1 Calculate position and velocity at a timestep Δt

Input: $\mathbf{x}_i^0$, $\mathbf{v}_i^0$, m_i, $C_j(j \in [1, 2, \ldots, M])$, iteration number IN, certain constraint order

Output: $\mathbf{x}_i^1$, $\mathbf{v}_i^1$

1: **for** each $i \in [1, N]$ **do**
2: initialize $\mathbf{x}_i = \mathbf{x}_i^0$, $\mathbf{v}_i = \mathbf{v}_i^0$, $w_i = 1/m_i$
3: $\mathbf{v}_i = \mathbf{v}_i + \Delta t w_i \boldsymbol{f}_{external}(\mathbf{x}_i)$
4: dampVelocities($\mathbf{v}_i$)
5: $\mathbf{p}_i = \mathbf{x}_i + \Delta t \mathbf{v}_i$
6: $C_k(k \in [1, 2, \ldots, M_{coll}])$ = generateCollisionConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
7: **while** iteration index $s < IN$ **do**
8: $\mathbf{p}_i$ = projectConstraints($C_1, C_2, \ldots, C_M, \ldots, C_{M+M_{coll}}, \mathbf{p}_i$)
9: $s++$
10: **end while**
11: $\mathbf{v}_i = (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
12: $\mathbf{x}_i = \mathbf{p}_i$
13: $\mathbf{v}_i$ = velocityUpdate($\mathbf{v}_i$)
14: $\mathbf{x}_i^1 = \mathbf{x}_i$, $\mathbf{v}_i^1 = \mathbf{v}_i$
15: **end for**
16: **return** Outputs

3.2.2 Damping

In (Müller et al. 2007), it has been proven that integrating a damping term $\mathbf{C}\dot{\mathbf{X}}$ into Newton's second law can improve the quality of PBD simulation, where $\mathbf{C}$ is a matrix for computing the damping and $\dot{\mathbf{X}}$ denotes a velocity vector. This term is used in line (4) of Algorithm 1 for reducing temporal oscillations of the point positions, which enhances the simulation stability. As point damping and spring damping can improve the realism of the simulation results by conserving momentum and point stability, they are most often used in applications.

3.2.3 Collision

Line (6) of Algorithm 1 generates extra temporary collision constraints to correct the calculation of the positions of each vertex. There are primarily two groups of constraints, depending on whether dynamic or static objects are considered.

When a dynamic object collides with a static object, there will also be two different conditions considering whether the ray $\mathbf{x}_i \rightarrow \mathbf{p}_i$ crosses or is inside any objects.

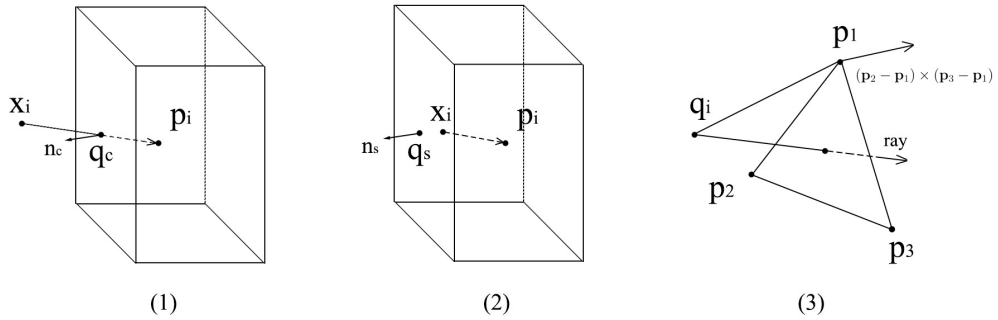


Figure 3.2: Three different conditions of object collision.

If the ray enters an object, like Fig. 3.2(1), the collision will be handled as continuous, with an *inequality* constraint $C(\mathbf{p}_i)$ and stiffness $k = 1$ added to the constraint list, computed as:

$$C(\mathbf{p}_i) \geqslant (\mathbf{p}_i - \mathbf{q}_c) \cdot \mathbf{n}_c \tag{3.1}$$

where $\mathbf{q}_c$ denotes the intersection point, and $\mathbf{n}_c$ is the normal of the collided object surface at $\mathbf{q}_c$.

If the ray is completely in the internal space of an object, like Fig. 3.2(2), the collision will be handled as static, with the *inequality* constraint $C(\mathbf{p}_i)$ computed as:

$$C(\mathbf{p}_i) \geqslant (\mathbf{p}_i - \mathbf{q}_s) \cdot \mathbf{n}_s \tag{3.2}$$

where $\mathbf{q}_s$ denotes the closest point of the collided object surface to $\mathbf{p}_i$, and $\mathbf{n}_s$ is still the normal of the collided object surface at $\mathbf{q}_c$.

When two dynamic objects collide, the computation becomes a bit more complex. Consider the collision as a point $\mathbf{q_i}$ of one object moves through a triangle, which consists of the closest three points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ of the other object to $\mathbf{q_i}$, like Fig. 3.2(3). The collision constraint can be computed as:

$$C(\mathbf{q}_i, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) \geqslant \pm(\mathbf{q}_i - \mathbf{p}_1) \cdot [(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)] \tag{3.3}$$

where $\pm$ is set to make sure that $\mathbf{q}_i$ is on the correct side of the triangle. This constraint preserves linear and angular momentum as it is independent of the rigid objects.

The generation of the collision constraints is done outside of the solver loop, which significantly reduces the running time of the simulation compared with physics-based methods.

3.2.4 Solver

Lines (7)-(9) of Algorithm 1 utilize a solver to correct the predicted position $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N$ of the N vertices for satisfying the $M + M_{coll}$ constraints, which can be presented as nonlinear equations. The solver contains a nonlinear Gauss-Seidel iteration to separately calculate each constraint one by one.

A correction $\Delta\mathbf{p}$ can be added to linearize the nonlinear constraints through:

$$C(\mathbf{p} + \Delta\mathbf{p}) = C(\mathbf{p}) + \nabla C(\mathbf{p}) \cdot \Delta\mathbf{p} + O(|\Delta\mathbf{p}|^2) \succ 0 \tag{3.4}$$

where the symbol $\succ$ means either $=$ or $\geqslant$.

Considering a second-order approximation, Eq. 3.4 can be written as:

$$C(\mathbf{p} + \Delta\mathbf{p}) \sim C(\mathbf{p}) + \nabla C(\mathbf{p}) \cdot \Delta\mathbf{p} \succ 0 \tag{3.5}$$

It can be figured out that the above system is undetermined as Eq. 3.5, which is not an equation. Thus, for solving the undetermined problem, the correction $\Delta\mathbf{p}$ should be restricted to be in the same direction with ∇C to preserve the linear and angular momentum, formulated as:

$$\Delta\mathbf{p} = \lambda \nabla C(\mathbf{p}) \tag{3.6}$$

where the scalar λ is a Lagrange multiplier, which can be determined by substituting Eq. 3.6 into Eq. 3.5 as:

$$\lambda = \frac{C(\mathbf{p})}{\sum_j w_j |\nabla_{\mathbf{p}_j} C(\mathbf{p})|^2} \tag{3.7}$$

Then the correction of each individual vertex $\mathbf{p}_i$ can be formulated as:

$$\Delta\mathbf{p}_i = -\lambda w_i \nabla_{\mathbf{p}_i} C(\mathbf{p}) \tag{3.8}$$

Lastly, the stiffness parameter $k \in [0, 1]$ will be added to the solver to control the strength of the constraints.

3.3 Recent Improvements of PBD

However, after surveying the recent works, it can be found that the original PBD is limited due to these three main defeats:

1. The inability to effectively achieve a convergent solution during iterations.

2. The dependence of results on stiffness parameters k, timestep Δt, and iteration count n_{iter}.

3. The influence of the processing order of different constraints in the solver.

In this section, the latest advancements in PBD will be reviewed, including advanced algorithms designed to address these primary limitations and recent developments extending PBD to other scenarios such as fluids and cloth.

3.3.1 Improvements in Convergence Problem

As discussed in Section 3.2, the iterative solver is the most critical step in PBD simulations. It helps the system achieve better corrections in each iteration, resulting in more realistic deformation outcomes. Nevertheless, PBD uses a nonlinear Gauss-Seidel solver for iterating projections, which causes slow propagation of corrections and makes it challenging to achieve a convergent solution. To enhance the solver's convergence efficiency, several methods have been proposed, such as Hierarchical Position-Based Dynamics (HPBD) (Müller 2008) and the second-order accurate multistep method (BDF2) (English and Bridson 2008). HPBD defines a multigrid-based mesh to propagate error corrections more swiftly, although its tearing algorithm requires further improvement. By incorporating BDF2 into PBD, which utilizes information from previous time steps, projection convergence is accelerated. However, this method may not be competitive for stretchy materials. Additionally, other methods like the Long Range Attachments (LRA) (Kim et al. 2012) method have been proposed to further expedite error propagation. Inextensible character clothing can well-suit LRA, whilst unattached environmental cloth, such as flying paper, does not benefit from it.

3.3.2 Improvements in Dependence Problem

To eliminate the dependence of simulation results on stiffness parameters k, timestep Δt, and the number of iterations n_{iter}, various methods have been proposed. Among these, Extended Position-Based Dynamics

Figure 3.3: The simulation results of a twisted rope demonstrate the ability of XPBD to generate detailed deformation results (Müller et al. 2020).

(XPBD) and Projective Dynamics (PD) have gained significant attention. These two methods not only address the dependence issue but also elevate position-based approaches into standalone research topics.

3.3.2.1 Extended Position Based Dynamics

XPBD was first introduced by Macklin et al. (2016), of which the primary function is to address the dependency issues inherent in PBD. This is achieved by associating a compliance $\alpha = \frac{1}{k}$ with each constraint. This simple modification to the PBD solver allows for a rigid solution regardless of the timestep. Additionally, XPBD allows the generation of accurate constraint force predictions for force-related effects by providing consistent solutions. Building on XPBD, by dividing each timestep into n isometric substeps and performing XPBD iterations at each substep, Macklin et al. (2019b) spent a lower computational cost to notably enhance achievable stiffness. Müller et al. (2020) improved XPBD to accurately resolve small spatiotemporal details in rigid body simulations, of which the ability to simulate twisted ropes was demonstrated in Fig. 3.3. Furthermore, Romeo et al. (2018) and Romeo et al. (2020) modified XPBD by adjusting the distance constraints between mesh vertices to define muscle dynamics simulations. Liu et al. (2022b) incorporated additional geometric constraints into XPBD to achieve differential parameter identification and shape control of linear objects for real-to-sim robotic manipulation.

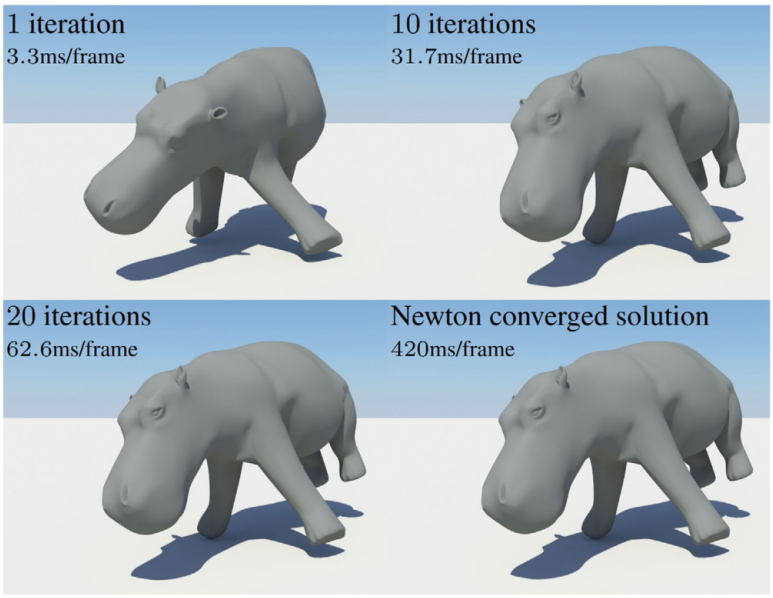


Figure 3.4: The computational time for each frame after $1, 10, 20$ iterations of simulating a hippopotamus model by PD and Newton's method from (**?**). It could be observed that, after 10 iterations, the simulation results retain as many details as the physics-based method, whilst PD only uses $1/10$ time of the latter for simulating one frame.

3.3.2.2 Projective Dynamics

Projective Dynamics (PD) is another extension of PBD, as introduced in (Bouaziz et al. 2014). PD incorporates additional constraints into the iteration solver to improve its robustness when handling non-uniform meshes with varying resolutions. This improvement allows PD to implicitly process interactions between objects, reducing the correlation between stiffness and the number of iterations. Moreover, as illustrated in Fig. 3.4, PD converges faster than Newton's method, thus significantly decreasing computation time.

Bouaziz et al. (2014) demonstrated that, due to the robustness and simplicity of PD, various materials could be simulated, such as cloth, shells, and solids. In (Soler et al. 2018), PD was further applied to simulate Cosserat rods. To accurately model the twisting and bending deformations of rods, it is essential to conserve angular momentum in addition to linear momentum. Therefore, the system is changed to include $\mathbf{q}_n, \mathbf{v}_n, \omega_n$, where ω_n represents the angular velocity of the n^{th} vertex at timestep t_n. Fig. 3.5 shows the high quality of visual results achieved. Furthermore, Solar et al. compared the convergence times of PD and

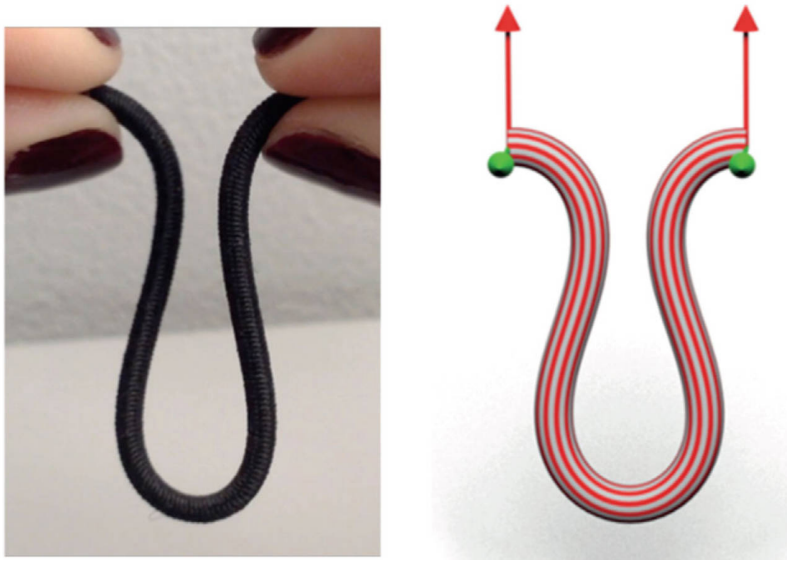


Figure 3.5: The simulation result of a folded rod. The left is a real elastic rod, while the right is a simulated rod generated by PD (Soler et al. 2018).

PBD and concluded that PD converges to a mesh-independent solution even faster than PBD.

PD can also be used to simulate deformable characters, consisting of rigid parts (bones) and deformable parts (flesh), with articulated skeletons, demonstrated in (Li et al. 2019). After integrating additional rigid-body constraints and joint constraints into the global step of the PD solver, they group and reorder n_f flesh vertices and n_b bone vertices using affine constraints. This method generates stable and efficient joint simulations with minimal joint error, achieving a quality comparable to state-of-the-art rigid body simulators.

3.3.3 Improvements in Other Limitations

In the Guass-Seidel-type solver of PBD, the constraints are handled independently one after another, hence the convergence order and final results are affected by the solving order of constraints. Only a few advanced methods have been proposed so far in addressing the order-dependent effects caused by the constraint processing order issues within PBD. For instance, Gu et al. (2017) introduced a sorting method to handle the fundamental constraint order, enhancing both realism and efficiency in cloth simulation.

In spite of this, the PBD solver has many other limitations, such as the lack of momentum constraints, which ties material stiffness to the timestep size and the number of iterations. XPBD has addressed this problem, while Dahl and Bargteil (2019) proposed a straightforward method to conserve angular momentum loss. Additionally, Bender et al. (2014c) explained that the Gauss-Seidel version of PBD cannot be effectively parallelized due to the constraint averaging issue. This limitation makes the number of iterations dependent on the number of constraints. Addressing this dependency problem is another crucial topic that warrants attention.

3.3.4 Improvements in Extensions

As previously mentioned, simulating particle-based dynamic objects was the goal that PBD was developed for, though it has since been extended to other scenarios, including cloth, fluid, and rigid body simulations.

3.3.4.1 Cloth Simulation

The basic stretch constraints involve distance constraints between adjacent vertices (Jakobsen 2001). Besides them, cloth simulation also requires handling bending constraints between adjacent triangles. For inextensible materials, since the edge lengths remain equal, adding an isometric bending constraint has been demonstrated particularly useful for simulating cloth with realistic folds and wrinkles (Bender et al. 2014b). Mohammed et al. (2020) applied extra position constraints on the PBD solver to maintain constant densities. This approach effectively generates various wind effects on cloth, addressing issues of dynamic layer preservation and wrinkles. Other scenarios may also impose various constraints to accommodate their unique attributes, including self-collision, cloth-balloon interaction, and strain energy constraints.

3.3.4.2 Fluid Simulation

Fluids were also initially modelled as particle systems by associating specific constraints to ensure the minimum distance between particles, known as Smoothed Particle Hydrodynamics (SPH) (Monaghan 1992). However, SPH has limitations, such as failing to achieve hydrostatic equilibrium at rest. To address this, Macklin and Müller (2013) integrated position-based concepts with SPH, introducing position based

fluids (PBF), which added density constraints to the solver. Shao et al. (2017) incorporated PBD position constraints into SPH to stabilize fluid-solid interactions. Their work demonstrated that PBD notably smoothed the vorticity of the fluid particle system, enhancing the visual fidelity of SPH simulations.

Building on PBF, Köster and Krüger (2016) proposed a method to significantly improve performance in specific scenarios by adaptively changing the position of each particle in the simulation system using fine-grained level-of-detail (LOD) information. Geyer (2022) introduced another modification to PBF, allowing fluid particles to adapt to the size of fluid shapes, thereby reducing particle count and computational cost. This approach resolved interaction issues between fluid particles of different sizes within PBF.

3.3.4.3 Rigid Body Simulation

The fundamental PBD method is versatile, extending beyond particle-based systems to rigid body simulations by incorporating joint and contact constraints, as shown in Fig. 3.6. The traditional PBD methods apply Newton's second law to particles, though they can only suit particle-based systems. This is because a particle has three spatial degrees of freedom (DOF), while a rigid body has three additional rotational DOFs to represent its orientation. For simulating rigid bodies, the constraints must be extended to the Newton-Euler equations to include rotational dynamics, treating a rigid body as a group of infinitely many particles.

Frâncu and Moldoveanu (2017) proposed a novel PBD formulation, which integrates friction and contact into the solver for rigid and elastic bodies using nonlinear convex optimization. Weiss et al. (2017) introduced PBD as a discrete algorithm for multi-agent crowd simulations, developing a set of position constraints and integrating them into the PBD solver. This provided a robust, stable, and easily implementable numerical framework for real-time crowd simulations. Sharma et al. (2020) enhanced multi-agent simulation by adding extra constraints using separating planes in the PBD solver to achieve flexible collision avoidance. Macklin et al. (2019a) presented a framework integrating an off-the-shelf linear solver into PBD to enhance rigid and deformable contact handling by a non-smooth Newton iteration method. This method showed great performance in robotic manipulation scenarios. In (Liu et al. 2022a),

Figure 3.6: The simulation result of rigid bodies. The top has 2000 elastic objects (Deul et al. 2016), while the bottom has 5000 rigid boxes (Weiss et al. 2017).

Liu et al. integrated optimal robot design, model-based motion control, and system identification into PBD to develop a differentiable framework, improving design efficiency and estimation accuracy. Moreover, Pan and Manocha (2018) introduced Position-Based Articulated Dynamics (PBAD), which reformulates articulated body dynamics simulation into an energy minimization problem using only position variables, enabling fully implicit integration. This approach improves efficiency at large timesteps compared to traditional methods but could not completely avoid numerical dissipation, lacking the precision of integrators in (Liu et al. 2022a).

For deformable object simulations based on position-based concepts, real-time performance is a critical standard met by most methods mentioned. However, there exist visual artefacts such as numerical damping and "explosions" when simulating with large timesteps and numerical approximations in PBD solvers. To correct these artefacts, Dinev et al. (2018) proposed a post-processing energy projection method with energy projection, generating visually plausible and stable motions in interactive systems.

3.3.5 Summary Table

The content of all the above improvement articles will be recorded in the following tables for easy inquiry and understanding. The contributions, advantages, and defeats of the mentioned improved methods of PBD are summarized in Table 3.1. The extensions of XPBD and PD are listed in Table 3.2. The advancements of PBD for different scenarios are summarized in Table 3.3.

Table 3.1: Contributions, advantages, and defeats of improved PBD algorithm.

Reference	**Method**	**Contribution**	**Advantages**	**Defeats**
Müller (2008)	HPBD	Alter the PBD solver by a nonlinear multigrid-based algorithm.	Faster convergence by accelerating error correction	(1)Visual artefacts with low iteration count;(2)solvers hard to parallelize;(3)without considering enough bending constraints.
English and Bridson (2008)	BDF2	Improve accuracy of position projection by a BDF2-based second order.	(1)Accelerate projection;(2)decrease damping.	(1)Bad for stretchy materials;(2)lacking smoothness;(3)perturbations caused by time splitting.
Kim et al. (2012)	LRA	Add global inextensibility.	(1)Accelerate error propagation to suit inextensible clothing;(2)make cloth stretching natural	(1)Bad for unattached environmental cloth;(2)unnatural in high resolution.
Macklin et al. (2016)	XPBD	(1)Add more physical constraints;(2)decouple the dependence between timestep, iteration count and stiffness.	(1)Address the dependence issues;(2)improve simulation results with tiny modification.	(1)Not enough accuracy;(2)high convergence cost.

Table 3.1: Contributions, advantages, and defeats of improved PBD algorithm (continued).

Reference	**Method**	**Contribution**	**Advantages**	**Defeats**
Bouaziz et al. (2014)	PD	Connect to physical systems with a local/global step to integrate energy potentials.	(1)Decouple the dependence to stiffness;(2)fast convergence;(3)high accuracy.	(1)Implicit damping problem;(2)bad for fluids;(3)not consider hard materials.
Gu et al. (2017)	Order sorting	Constraint adjustment sequence.	Enhance realism with efficient convergence.	Triangulations affect the results.
Dahl and Bargteil (2019)	Momentum compensation	Optimize correction to conserve global momentum.	(1)Negligible computational cost;(2)improve accuracy.	Few cases require angular momentum conservation.

Table 3.2: Contributions, advantages, and defeats of extensions in XPBD & PD.

Reference	**Based Method**	**Contribution**	**Advantages**	**Defeats**
Macklin et al. (2019b)	XPBD	Utilize timestep subdivision.	(1)Higher stiffness and less stretching;(2)use an implicit integrator to decrease constraint error;(3)use an explicit integrator to stabilize simulation.	(1)Increase computational cost;(2)bad for decreasing velocity error;(3)iteration order depends on residual;(4)expensive iteration cost.
Müller et al. (2020)	XPBD	Accurately remain small details for rigid body simulation.	(1)Increase accuracy;(2)easy manipulation by mass ratio and frequent rotation.	(1)Bad for high-frequency vibrations;(2)require powerful computing devices;(3)not robust for complex scenarios.
Romeo et al. (2018 2020)	XPBD	Modify distance constraint to simulate contract of muscles and fascia.	(1)High controllability;(2)efficient simulation of muscle movement.	Limited application scenarios.
Liu et al. (2022b)	XPBD	(1)Add extra geometric constraints;(2)solving constraints in a differentiable way;(3)define rope-like object issue.	(1)Robust and precise simulation results;(2)work in real-time environments.	Coupling problem between collision processing and deformable rigid bodies.

Table 3.2: Contributions, advantages, and defeats of extensions in XPBD & PD(continued).

Reference	**Based Method**	**Contribution**	**Advantages**	**Defeats**
Soler et al. (2018)	PD	(1)Integrate angular velocities;(2)Add extra constraints;(3)present potential weights for rods.	(1)Accurate rod simulation;(2)numerically robust;(3)fast convergence.	(1)Limited application scenarios;(2)More computational cost.
Li et al. (2019)	PD	(1)Articulated soft character simulation;(2)formulate all vertices;(3)accurate enforcement of joint constraints.	Enhance performance.	Extra bone rotation caused by insufficient joint limits.

Table 3.3: Contributions, advantages, and defeats of PBD improvements for different scenarios.

Scenatio	**Reference**	**Contribution**	**Advantages**	**Defeats**
Cloth Simulation	Bender et al. (2014b)	Add constraints representing continuum mechanics	Capacity of generating complicated physical effects, including lateral contraction, anisotropy and isotropy.	(1)More visually authentic;(2)inherit dependence problem;(3)bad convergence result.
	Mohammed et al. (2020)	(1)Create cloth wind effects;(2)address problems including wrinkle preservation and cloth overlay.	Enhance convergence and compressibility;(2)Good results in real-time environments	Bad for solid object simulation caused by insufficient physical constraints.
Fluid Simulation	Macklin and Müller (2013)	Integrate PBD solver with density constraint for fluid simulation	(1)High computation efficiency;(2)generalize PBD to fluid simulation.	(1) Slow convergence problems caused by large number particles stacking;(2)parameters are dependent.
	Shao et al. (2017)	(1)Add position constraint to alleviate penetration problem;(2)Add vorticity constraint for diffusion stabilization.	(1)More visually authentic;(2)stabilize solid boundary processing;(3)Enhance efficiency.	Lacking experiments with different phenomena.

Table 3.3: Contributions, advantages, and defeats of PBD improvements for different scenarios(continued).

Scenatio	Reference	Contribution	Advantages	Defeats
	Köster and Krüger (2016)	Modify solver with proper iteration count via fine-grained LOD.	(1)Enhance PBF performance;(2)stably conserve average density when particle positions modify.	(1)Not enough test for adaption models;(2)insufficient constraints for authentic simulation.
	Geyer (2022)	Improving the fluid shape control of PBF	(1)Low computational cost;(2)enhance performance.	(1)Bad for large-scale particle systems;(2)Low adaption standard.
Rigid Body Simulation	Deul et al. (2016)	Add joint and contact constraints.	(1)Easy Control;(2)good performance for complex scenarios with large-scale rigid bodies;(3)present coupling of rigid and deformable objects.	Insufficient components, including motors.
	Frâncu and Moldoveanu (2017)	Simulate rigid and deformable models simultaneously via adding friction and contact constraint.	(1)Authentic and stable simulation;(2)accelerate computation.	(1)Cause some visual artefacts;(2)insufficient friction smoothing.

Table 3.3: Contributions, advantages, and defeats of PBD improvements for different scenarios(continued).

Scenatio	Reference	Contribution	Advantages	Defeats
	Weiss et al. (2017)	Add extra constraints to simulate crowds.	(1)Enhance crowd simulation in efficiency and stability;(2)make behaviour more flexible and emergent.	(1)Not authentic enough;(2)not support parameter tuning.
	Pan and Manocha (2018)	Provide a time integrator into the PBD solver.	(1)stable;(2)use large timestep to accelerate convergence.	(1)Insufficient accuracy;(2)Numerical dissipation.
	Dinev et al. (2018)	Present a post-processing method for energy projection.	Improve realism, stability and computational efficiency	(1)Insufficient numerical accuracy;(2)Oscillation issues.
	Macklin et al. (2019a)	(1)Smooth friction;(2)provide a preconditioner and a compliance formulation for simulating hyper-elastic materials.	Enhance convergence and robustness.	(1)Not capable of processing energy-preserving integrator or elastic collision;(2)only suits particle-based system;(3)poor performance.
	Sharma et al. (2020)	Multi-agent simulation by separation planes.	(1)Moderate agent collision;(2)easy control;(3)high compatibility.	Insufficient application fields.

Table 3.3: Contributions, advantages, and defeats of PBD improvements for different scenarios(continued).

Scenatio	**Reference**	**Contribution**	**Advantages**	**Defeats**
	Liu et al. (2022a)	Provide a differentiable framework to simulating rigid body	High compatibility, accuracy and efficiency.	Insufficient comparison.

3.4 Recent Applications of PBD

In recent years, the applications of PBD have extended to several other fields. In this section, the latest applications in specific areas, including deep learning, medical simulations, and architecture, will be reviewed.

3.4.1 Deep Learning-related Application

With the rapid development of deep learning (DL) techniques in this decade, it has become a hot topic in computer graphics, which has been applied to various fields such as 3D representation, image transmission, and autonomous vehicles, achieving significant progress. The integration of PBD and DL has been proven effective by many researchers. For example, graph networks (GN) have been used in PBD as an acceleration method for simulating rod dynamics to estimate constraint projections (Shao 2022). Compared to the original PBD, the method integrated GN enhances runtime performance. Beyond improving PBD performance, the high-quality simulation data generated by PBD has been used as input for data-driven methods to approximate physical forces effectively (Holden et al. 2019). Yang et al. (2020) learned and predicted physical rules for controlling challenging scenarios by embedding neural networks in the projection constraint step. Kim et al. (2022) introduced an Anisotropic Constraint Boundary Convolutional Neural Network (AnisoCBConvNet), which uses PBD as a dynamic solver to generate surface data for deep learning. Except for PBD, PBF (Macklin and Müller 2013) has also been implemented in Smooth Particle Networks (SPNets) (Schenck and Fox 2018) to compute rigid body and liquid interactions, as shown in Fig. 3.7.

3.4.2 Medicine Application

In addition to its applications in cloth, deformable objects, and fluid simulations, PBD has also been employed in surgical simulations since PBD avoids overshooting issues and is easier to handle collision constraints compared with physics-based methods. Pan et al. (2015) simulated hybrid soft tissue by an interactive dissection method, which applies constraints conserving energy and volume to PBD, enhancing the visual realism of simulation results.

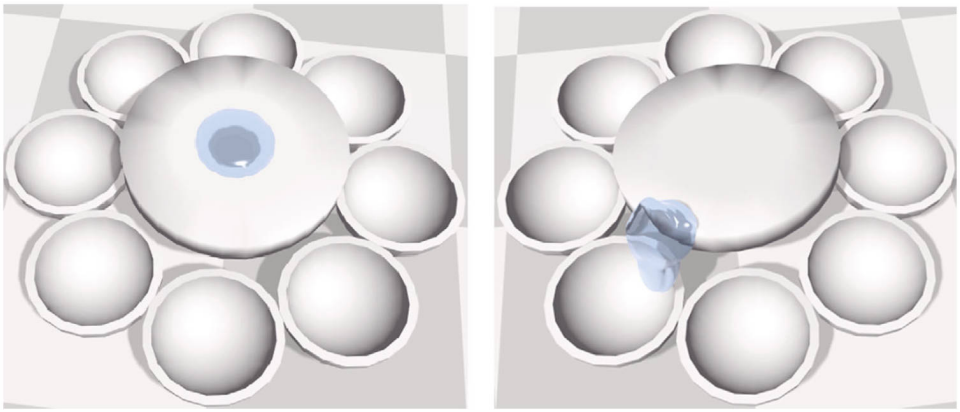


Figure 3.7: The simulation result of fluids interacting with a rigid body, produced by SPNets (Schenck and Fox 2018).

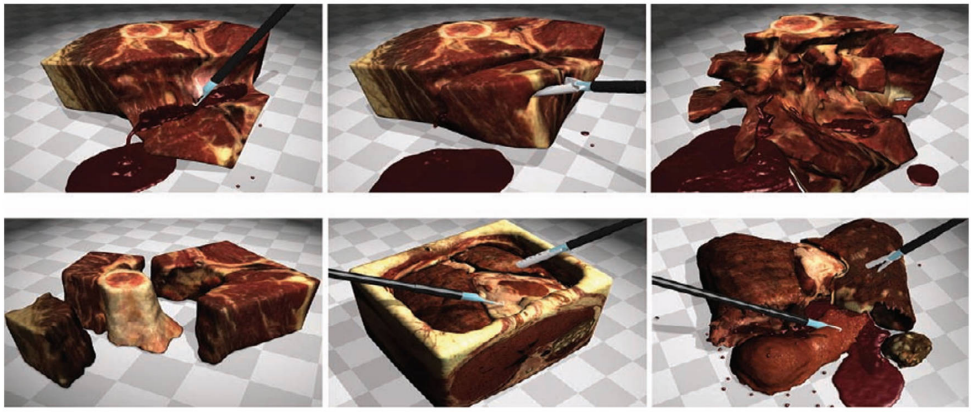


Figure 3.8: The simulation result of soft tissue incisions in a virtual surgery (Berndt et al. 2017).

Following this, Berndt et al. (2017) leveraged PBD's capabilities to model all dynamic objects in surgery, simulating soft tissue incisions (as shown in Fig. 3.8), bones, and body fluids by associating different constraints for corresponding materials. Their method demonstrates faster simulation speeds and better scalability when simulating hybrid object scenarios compared to FEM-based methods proposed by Wu et al. (2015a). Walczak et al. (2019 2020 2022) simulated mitral and aortic valves using a simple material model, utilizing PBD for interactive parametrization to simulate missing information.

Moreover, Han et al. (2020) simulate tissue deformation within a 2D surgical framework with PBD. Xu et al. (2018) proposed a method incorporating mass-spring-damper (MSD) constraints into PBD for simulating soft tissue deformation during laparoscopic cholecystectomy, addressing

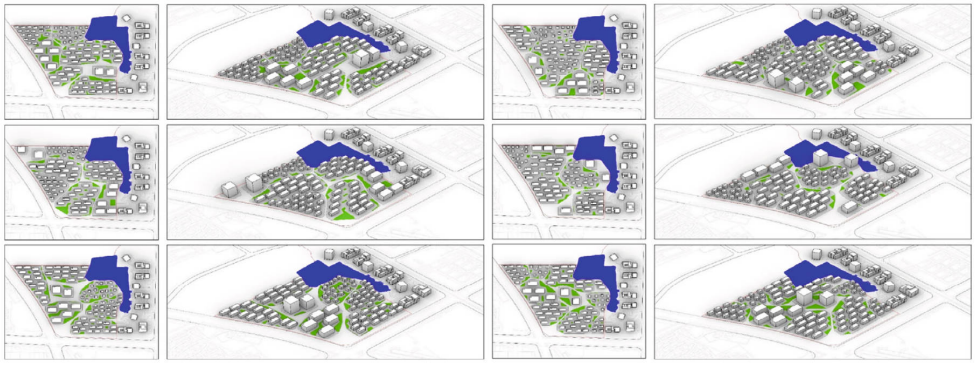


Figure 3.9: The simulation result of automatic city layout in solving an urban design task (Cao and Ji 2021).

the dependency problem between deformation effects and iterative variables. Additionally, PBD has been used in virtual surgery to simulate soft tissues (Tagliabue et al. 2020, Liu et al. 2020b), demonstrating its versatility and effectiveness in medical simulations.

3.4.3 Architectural Application

PBD has also been applied to architectural layout tasks, such as urban design, where architectural layout standards can be viewed as layout constraints. Cao and Ji (2021) modelled the city layout problem as the positioning and orientation of buildings $B = b_1, b_2, \ldots, b_n$. In real layout design task, the building entities are often defined by a series of virtual models, which are called agent models (AM), to explain their properties, including shape, shadow and field. For each building $b_i \in B$ $(i = 1, 2, \ldots, n)$, additional attributes include position p_i, orientation θ_i, associated shape mesh, a set of agent models, and inverse mass $w_i = \frac{1}{m_i}$, where m_i is the building mass. A priority stiffness k is specially set for control the order of constraints solving. Due to their inherent properties, different agent models will also affect the correction process. Thus, two extra steps were added to the original PBD solver: updating the priority stiffness k and agent models AM before generating collision constraints. As shown in Fig. 3.9, with this improved PBD method, the city layout can be automatically generated.

3.4.4 Summary Table

The content of all the above application articles will be recorded in the following table for easy inquiry and understanding, where the different

applications of PBD in various fields are listed in Table 3.4.

Table 3.4: PBD applications in different fields.

Field	Reference	Integration with PBD
Deep learning	Schenck and Fox (2018)	Embed PBF in SPNets to simulate the interaction between fluid and solid.
	Holden et al. (2019)	Utilize PBD simulation results as inputs of networks.
	Yang et al. (2020)	Utilize neural networks for correct projection constraints in PBD.
	Shao (2022)	Speed up deformable rod simulation by integrating GN into PBD.
	Kim et al. (2022)	Utilize PBD simulation results as inputs of networks.
Medical	Pan et al. (2015)	Integrate volume-conserving and energy constraints into PBD to enhance visual performance.
	Wu et al. (2015a)	Compare the surgery simulation performance of FEM and PBD.
	Berndt et al. (2017)	Tissue simulation via PBD.
	Xu et al. (2018)	Add MSD constraints to PBD.
	Walczak et al. (2019 2020 2022)	Aortic valve and mitral simulation via PBD.
	Han et al. (2020)	Tissue deformation in 2D surgical framework via PBD.
	Tagliabue et al. (2020)	Soft tissue simulation via PBD.

Table 3.4: PBD applications in different fields(continued).

Field	**Reference**	**Integration with PBD**
Architectural	Cao and Ji (2021)	Large-scale city layout via PBD.

3.5 Conclusion

In this chapter, the latest developments and applications of position-based methods are reviewed. By introducing the core algorithm of the original PBD for particle-based systems, the significance of its main three steps, which correct the vertex attributes, is explained, including damping, collision, and solver. Next, improvements in position-based methods since 2018 are examined, highlighting significant extensions such as Projective Dynamics (PD) and Extended Position-Based Dynamics (XPBD). The simulations of various materials are introduced, along with modifications made to accommodate different scenarios, including rigid bodies, fluids, and cloth. Then, the recent applications of PBD in medical simulations, integration with deep learning, and architectural layout planning are reviewed.

From these works, we can observe that PBD often excels in applications requiring high real-time performance. Since PBD omits the velocity and acceleration layers by directly correcting position, it is computationally efficient while still producing visually plausible simulation results. Since the primary target of this research is to create authentic digital character animations in real-time environments, in Chapter 4, we will consider PBD simulation results as our ground truth in surface reconstruction techniques, emphasizing the two advantages of this research: real-time performance and realism.

Chapter 4

PDE-based Dynamic Reconstruction Integrating PBD

4.1 Introduction

As mentioned in Chapter 1, despite the challenges posed by the COVID-19 pandemic, the global gaming market has remained vibrant in recent years, leading to intense competition for player engagement. To enhance the gaming experience, advanced technologies such as voice control and facial recognition have been introduced. However, the quality of digital character animations continues to be a critical factor influencing players' perceptions of interactive games. Researchers are constantly seeking techniques to create realistic animations of digital characters in real-time environments, including modelling, skin deformation, and motion construction methods.

Game models are typically created for interactive systems, necessitating that they are both small and capable of fulfilling real-time rendering. To achieve this, mesh simplification techniques like (Daniels et al. 2008, Ebke et al. 2014) have been developed to reduce design parameters in high-resolution models or to create low-polygon models from scratch. However, low-resolution 3D models often fall short of providing the natural and precise animations required for interactive games. To balance computational efficiency with model detail, techniques such as motion prediction and anti-aliasing have been developed. Besides, level-of-detail (LOD) is indispensable in modern open-world games, as it significantly reduces computational resources by decreasing model detail for objects outside the player's immediate view. While LOD alleviates the burden

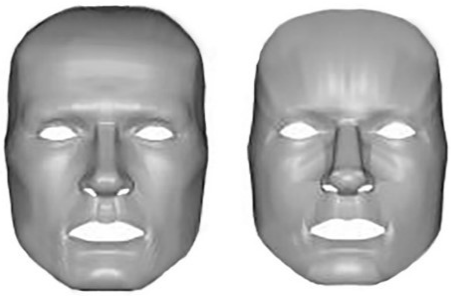


Figure 4.1: A human face model and its reconstruction created by ODE-based surface creation method.

of more detailed models, the reconstruction of simplified meshes remains a challenging issue.

Section 2.1.1.3 of Chapter 2 introduced ODE-based Sweeping Surface, which significantly reduces data size and enhances the controllability of surface shapes without losing realistic details. Fig. 4.1 is a comparison example of ODE-based surfaces and ground truth. The left is a human face model, consisting of 15,378 polygons and 8,221 vertices. The right is reconstructed with 4,236 polygons and 4,764 vertices from the left ground truth model via an ODE-based surface creation approach, introduced in Chapter 5. It can be found that the ODE-based reconstruction model owns much fewer surfaces and vertices compared with the original one, with most visual details preserved, like nasolabial folds and cheekbone. This demonstrates the potential of ODE-based modelling approaches in creating realistic models in real-time environments.

Similar to ODE-based modelling introduced in Chapter 5, PDE-based modelling creates 3D models by finding solutions of vector-valued PDEs with boundary constraints, whilst it has the ability to represent more complex surfaces. According to (Sheng et al. 2010), the approximation by NURBS reduces an original femur polygon mesh of 3.2 MB to 0.55 MB, whilst an analytical PDE approximation can reduce it to 0.26 MB. This illustrates that design variables can be significantly reduced by PDE-based modelling compared to patch-based and polygonal modelling techniques.

PDE-based modelling that disregards the effects of acceleration and velocity is known as static PDE-based modelling, which is only considered in existing ODE/PDE-based modelling technologies. For dynamic objects, PDE-based modelling considers the effects of acceleration and velocity by integrating their underlying physical principles as constraints. These PDE-based methods for dealing with dynamic scenarios are called

dynamic PDE-based modelling. Although extensive research has been conducted on static PDE-based modelling, there has been limited study on dynamic PDE-based modelling. To date, we have not encountered any work combining dynamic PDEs with deformation simulation to reconstruct dynamic 3D models. Dynamic PDE-based modelling not only has the advantage of reducing the number of design variables, but the 3D models reconstructed via dynamic PDE-based modelling approaches also possess time dependency. By setting time variables involved in the mathematical expressions to define dynamic 3D models, some keyframes in deformation simulations can be replaced with corresponding keyframes generated by dynamic PDE modelling. This can circumvent part of calculations and thus enhance the efficiency of real-time simulations.

According to the above motivations, we will introduce a new surface creation method based on a PDE mathematical model, which has the following contributions:

- Developing a PDE-based dynamic reconstruction method to recreate deformation models from a series of keyframes in high-quality performance, including high efficiency, good accuracy and small data sizes.

- Integrating the governing equation of elastic beams into Newton's second law to describe the modelling of dynamic objects. This forms a new mathematical model, which makes our method physics-based for creating natural shape changes.

- Applying the separation of variables technique for fast computing the approximation, as this physics-based method converts the modelling issue to an approximation issue.

- Integrating our method with advanced position-based deformation methods for high-quality and fast reconstruction of deformable models in real-time simulation.

- Adding time variable as a parameter into the mathematical model to describe the consistency among keyframe models in the same series, which allows interpolating of nonexistent keyframes and omitting in-between keyframes during storage to reduce the data size of digital character animations.

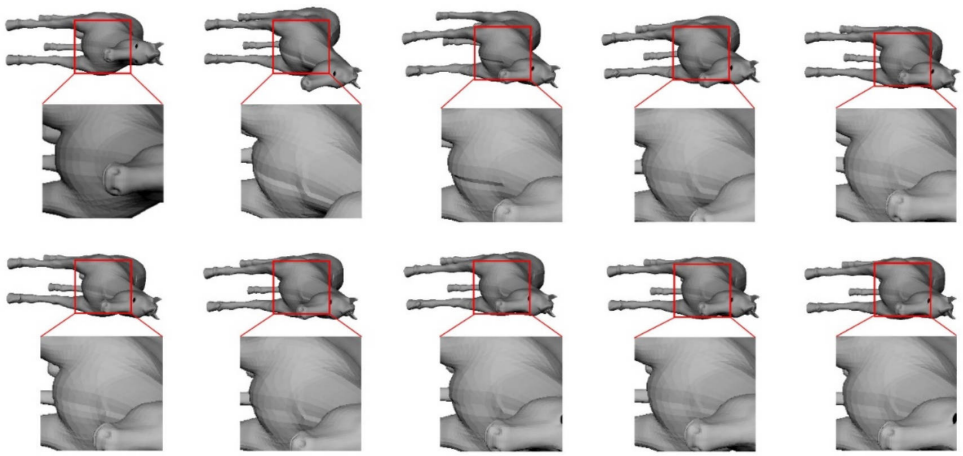


Figure 4.2: The deformation simulation results of a horse model, generated by PBD. From the top left, these models represent the deformed horse at frame 1,20,30,40,50,60,70,80,90,100, which proves the great simulation performance of PBD.

- Combining all closed-form solutions into series with infinite terms and complicated functions to enhance the capability of complex dynamic reconstructions.

4.2 Deformation Simulation by PBD

Since our method aims to create surfaces with deformation, the first step is to acquire the reference deformable keyframe models. We choose PBD as our simulation ground truth for dynamic deformable models due to its simplicity and capacity for fast computation. The code package for simulation is based on GitHub *PositionBasedDynamics*[1], which is designed for visually plausible simulation of deformable solids, fluids, and rigid bodies. The deformation results are generated on an Intel 6700 CPU with a clock rate of 3.4GHz, and will be utilized in Section 4.4 and 4.5. Some keyframe models generated from the PBD simulation of a horse model are shown in Fig. 4.2, where the neck can be observed stretching and torsion. The utilization of the package demonstrated the capacity for broad applicability and high-quality performance of PBD.

[1] `https://github.com/InteractiveComputerGraphics/PositionBasedDynamics`

4.3 PDE-based Surface Reconstruction

In modern interactive systems like games, the realism of deformation simulation is often enhanced by integrating physics-based methods, as they can provide accurate 3D geometries of bones and muscles for digital characters. Newton's second law has been applied by most approaches to develop dynamic simulations for computer animations due to its foundation and importance. Although Newton's second law primarily applies to particles (Bender et al. 2015), it has been used to describe shape deformation in 3D models. As introduced in Chapter 3, this includes simulating soft tissue in virtual surgery, deforming 3D facial models, and describing the dynamic deformation of cloth. In this research, we also introduce Newton's second law to develop a physics-based approach for presenting deformable surfaces.

Newton's second law has been proven to be applied to particle systems in (Bender et al. 2015). The points in the system move independently when simulating their movements by Newton's second law. However, this contradicts the fact that the points defining a curve cannot move independently. For describing curve deformations, we consider the curve deformation as beam deformation. By integrating the governing equation of an elastic beam with Newton's second law, the points independently moving problem of a curve can be addressed.

Nevertheless, precise models are necessary for achieving high realism in physics-based modelling. Processing these models involves extensive numerical computations, resulting in inefficiency. To balance realism and efficiency, interactive systems like computer animation and video games often use simplified physics models. For instance, in finite element simulations of anatomy and physics-based facial animation, soft tissues are simplified to isotropic and linear elastic materials, despite their anisotropic, heterogeneous, and nonlinear elastic properties (Gladilin et al. 2004). Similarly, we simplify the deformations of 3D models to be isotropic and linear elastic, using Young's modulus to describe their mechanical properties.

Building on this foundation, we combined Newton's second law of motion with the governing equation for elastic beam bending deformation to establish a new mathematical model. After deriving a closed-form solution for this model and integrating it with PBD, we developed a

dynamic reconstruction method specifically designed for reconstructing dynamic 3D models with small data sizes for real-time environments.

4.3.1 Mathematical Model

Building upon (Barrielle et al. 2016), Newton's second law can be applied to shape changes, depicting the dependence between the product of the acceleration and mass of an object and its external forces:

$$ma = f \tag{4.1}$$

The governing equation of elastic beams for bending deformation is :

$$EI\frac{d^4w}{dx^4} = f \tag{4.2}$$

where $w(x)$ denotes the beam deflection in the z direction at position x, and EI is the flexural rigidity.

Integrating Eq. 4.1 with Eq. 4.2, we gain:

$$ma = EI\frac{d^4w}{dx^4} \tag{4.3}$$

Since a vertex w has three components x, y, z of coordinates in 3D space, Eq. 4.3 can be separated into three parts:

$$\begin{aligned} ma_x &= EI\frac{d^4x}{du^4} \\ ma_y &= EI\frac{d^4y}{du^4} \\ ma_z &= EI\frac{d^4z}{du^4} \end{aligned} \tag{4.4}$$

where u indicates the beam's parametric direction for converting the deflection $w(x)$ to coordinate $w = (x, y, z)$.

Letting $D = EI$, Eq. 4.4 can be combined to:

$$ma_w = D\frac{d^4w}{du^4} \tag{4.5}$$

The acceleration a_w describes the second derivative of displacement w with respect to the time t:

$$a_w = \frac{d^2w}{dt^2} \tag{4.6}$$

Combining Eq. 4.4 and Eq. 4.5, the PDE mathematical model for dynamic reconstruction is gained:

$$m\frac{\partial^2w}{\partial t^2} = D\frac{\partial^4w}{\partial u^4} \tag{4.7}$$

4.3.2 Closed-form Solutions to PDE

By utilizing the separation of variables technique, we can gain the solution to Eq. 4.7 with the following form:

$$w(u,t) = w_1(u) \cdot w_2(t) \tag{4.8}$$

Rewritten Eq. 4.8 by substituting the fourth partial derivative of w with respect to the parametric variable u and the second partial derivative of w with respect to the time variable t as:

$$m\frac{w_2''(t)}{w_2(t)} = D\frac{w_1''''(u)}{w_1(u)} = b_w \tag{4.9}$$

where b_w is a non-zero constant.

Then the partial differential equation 4.9 can be separated into two ordinary differential equations:

$$w_1''''(u) - b_w w_1(u)/D = 0 \tag{4.10}$$

and

$$w_2''(t) - b_w w_2(t)/m = 0 \tag{4.11}$$

First solving Eq. 4.10 by introducing $w_1(u) = e^{ru}$, we can get:

$$r^4 e^{ru} - b_w e^{ru}/D = 0$$

and deleting e^{ru}, a simplified characteristic equation is gained:

$$Dr^4 - b_w = 0 \tag{4.12}$$

of which roots are:

If $b_w/D > 0$,

$$r_1 = D_{w1}, \quad r_2 = -D_{w1}, \quad r_3 = iD_{w1}, \quad r_4 = -iD_{w1}$$

By instituting the roots into $w_1(u) = e^{ru}$, the closed-form solutions to Eq. 4.10 is:

$$\begin{aligned} w_1(u) = {} & c_{w1}^* e^{D_{w1}u} + c_{w2}^* e^{-D_{w1}u} \\ & + c_{w3}^* cos D_{w1}u + c_{w4}^* sin D_{w1}u \end{aligned} \tag{4.13}$$

If $b_w/D < 0$, the roots are:

$$r_{1,2} = \sqrt{2}(1 \pm i)D_{w1}/2, \qquad r_{3,4} = -\sqrt{2}(1 \pm i)D_{w1}/2 \tag{4.14}$$

By instituting the roots into $w_1(u) = e^{ru}$, the closed-form solutions to Eq. 4.10 is:

$$\begin{aligned} w_1(u) = & c^*_{w1} e^{\xi D_{w1} u} cos\xi D_{w1} u + c^*_{w2} e^{\xi D_{w1} u} sin\xi D_{w1} u \\ & + c^*_{w3} e^{-\xi D_{w1} u} cos\xi D_{w1} u + c^*_{w4} e^{-\xi D_{w1} u} sin\xi D_{w1} u \end{aligned} \tag{4.15}$$

where $D_{w1} = \sqrt[4]{|b_w/D|}$, $\xi = \sqrt{2}/2$, and $c^*_{wi} (i = 1, 2, 3, 4)$ are unknown constants.

Then solving Eq. 4.11 by introducing $w_2(t) = e^{rt}$, we can get:

$$r^2 e^{rt} - b_w e^{rt}/m = 0$$

and deleting e^{rt}, a simplified characteristic equation is gained:

$$mr^2 - b_w = 0 \tag{4.16}$$

of which roots are:

If $b_w/m > 0$,

$$r_1 = m_{w1}, \quad r_2 = -m_{w1}$$

By instituting the roots into $w_2(t) = e^{rt}$, the closed-form solutions to Eq. 4.11 is:

$$w_2(t) = c^*_{w5} e^{m_{w1} t} + c^*_{w6} e^{-m_{w1} t} \tag{4.17}$$

If $b_w/m < 0$, the roots are:

$$r_3 = i m_{w1}, \quad r_4 = -i m_{w1}$$

By instituting the roots into $w_2(t) = e^{rt}$, the closed-form solutions to Eq. 4.11 is:

$$w_2(t) = c^*_{w5} cos m_{w1} t + c^*_{w6} sin m_{w1} t \tag{4.18}$$

where $m_{w1} = \sqrt{|b_w/m|}$, and $c^*_{wi} (i = 5, 6)$ are unknown constants.

Integrating the closed-form solutions $w_1(u)$ to Eq. 4.10 and the closed-form solutions $w_2(t)$ to Eq. 4.11 into Eq. 4.8, the closed-form solutions to the proposed mathematical model can be calculated as:

If $b_w/m > 0, b_w/D > 0$,

$$\begin{aligned} w(u,t) = & c_{w1} e^{m_{w1} t} e^{D_{w1} u} + c_{w2} e^{m_{w1} t} e^{-D_{w1} u} + c_{w3} e^{m_{w1} t} \cos D_{w1} u \\ & + c_{w4} e^{m_{w1} t} \sin D_{w1} u + c_{w5} e^{-m_{w1} t} e^{D_{w1} u} + c_{w6} e^{-m_{w1} t} e^{-D_{w1} u} \\ & + c_{w7} e^{-m_{w1} t} \cos D_{w1} u + c_{w8} e^{-m_{w1} t} \sin D_{w1} u \end{aligned} \tag{4.19}$$

If $b_w/m > 0, b_w/D < 0$,

$$\begin{aligned} w(u,t) =& c_{w1}e^{m_{w1}t}e^{\xi D_{w1}u}\cos\xi D_{w1}u + c_{w2}e^{m_{w1}t}e^{\xi D_{w1}u}\sin\xi D_{w1}u \\ &+c_{w3}e^{m_{w1}t}e^{-\xi D_{w1}u}\cos\xi D_{w1}u + c_{w4}e^{m_{w1}t}e^{-\xi D_{w1}u}\sin\xi D_{w1}u \\ &+c_{w5}e^{-m_{w1}t}e^{\xi D_{w1}u}\cos\xi D_{w1}u + c_{w6}e^{-m_{w1}t}e^{\xi D_{w1}u}\sin\xi D_{w1}u \\ &+c_{w7}e^{-m_{w1}t}e^{-\xi D_{w1}u}\cos\xi D_{w1}u + c_{w8}e^{-m_{w1}t}e^{-\xi D_{w1}u}\sin\xi D_{w1}u \end{aligned} \tag{4.20}$$

If $b_w/m < 0, b_w/D > 0$,

$$\begin{aligned} w(u,t) =& c_{w1}\cos(m_{w1}t)e^{D_{w1}u} + c_{w2}\cos(m_{w1}t)e^{-D_{w1}u} \\ &+c_{w4}\cos(m_{w1}t)\cos D_{w1}u + c_{w3}\cos(m_{w1}t)\sin D_{w1}u \\ &+c_{w5}\sin(m_{w1}t)e^{D_{w1}u} + c_{w6}\sin(m_{w1}t)e^{-D_{w1}u} \\ &+c_{w7}\sin(m_{w1}t)\cos D_{w1}u + c_{w8}\sin(m_{w1}t)\sin D_{w1}u \end{aligned} \tag{4.21}$$

If $b_w/m < 0, b_w/D < 0$,

$$\begin{aligned} w(u,t) =& c_{w1}\cos(m_{w1}t)e^{\xi D_{w1}u}\cos\xi D_{w1}u + c_{w2}\cos(m_{w1}t)e^{\xi D_{w1}u}\sin\xi D_{w1}u \\ &+c_{w3}\cos(m_{w1}t)e^{-\xi D_{w1}u}\cos\xi D_{w1}u + c_{w4}\cos(m_{w1}t)e^{-\xi D_{w1}u}\sin\xi D_{w1}u \\ &+c_{w5}\sin(m_{w1}t)e^{\xi D_{w1}u}\cos\xi D_{w1}u + c_{w6}\sin(m_{w1}t)e^{\xi D_{w1}u}\sin\xi D_{w1}u \\ &+c_{w7}\sin(m_{w1}t)e^{-\xi D_{w1}u}\cos\xi D_{w1}u + c_{w8}\sin(m_{w1}t)e^{-\xi D_{w1}u}\sin\xi D_{w1}u \end{aligned} \tag{4.22}$$

where $c_{wi}(i = 1, 2, \ldots, 8)$ are unknown constants.

4.3.3 Extend PDE with Infinite Terms

Putting together the different combinations from Eq. 4.19 to Eq. 4.22, we can use the following equation to represent the position $w(u,t)$:

$$w(u,t) = \sum_{i=1}^{32} c_{wi} f_{wi}(u, t, D_{w1}, m_{w1}) \tag{4.23}$$

where c_{wi} $(i = 1, 2, \ldots, 32)$ are unknown constants combined by all constants of the 4 distinct conditions, and $f_{wi}(u, t, D_{w1}, m_{w1})$ $(i = 1, 2, \ldots, 8)$

are given as follows:

$$
\begin{aligned}
f_{w1}(u,t,D_{w1},m_{w1}) &= e^{m_{w1}t}e^{D_{w1}u}\\
f_{w2}(u,t,D_{w1},m_{w1}) &= e^{m_{w1}t}e^{-D_{w1}u}\\
f_{w3}(u,t,D_{w1},m_{w1}) &= e^{m_{w1}t}\cos D_{w1}u\\
f_{w4}(u,t,D_{w1},m_{w1}) &= e^{m_{w1}t}\sin D_{w1}u\\
f_{w5}(u,t,D_{w1},m_{w1}) &= e^{-m_{w1}t}e^{D_{w1}u}\\
f_{w6}(u,t,D_{w1},m_{w1}) &= e^{-m_{w1}t}e^{-D_{w1}u}\\
f_{w7}(u,t,D_{w1},m_{w1}) &= e^{-m_{w1}t}\cos D_{w1}u\\
f_{w8}(u,t,D_{w1},m_{w1}) &= e^{-m_{w1}t}\sin D_{w1}u\\
f_{w9}(u,t,D_{w1},m_{w1}) &= e^{m_{w1}t}e^{\xi D_{w1}u}\cos \xi D_{w1}u\\
f_{w10}(u,t,D_{w1},m_{w1}) &= e^{m_{w1}t}e^{\xi D_{w1}u}\sin \xi D_{w1}u\\
f_{w11}(u,t,D_{w1},m_{w1}) &= e^{m_{w1}t}e^{-\xi D_{w1}u}\cos \xi D_{w1}u\\
f_{w12}(u,t,D_{w1},m_{w1}) &= e^{m_{w1}t}e^{-\xi D_{w1}u}\sin \xi D_{w1}u\\
f_{w13}(u,t,D_{w1},m_{w1}) &= e^{-m_{w1}t}e^{\xi D_{w1}u}\cos \xi D_{w1}u
\end{aligned}
$$

$$
\begin{aligned}
f_{w14}(u,t,D_{w1},m_{w1}) &= e^{-m_{w1}t}e^{\xi D_{w1}u}\sin\xi D_{w1}u \\
f_{w15}(u,t,D_{w1},m_{w1}) &= e^{-m_{w1}t}e^{-\xi D_{w1}u}\cos\xi D_{w1}u \\
f_{w16}(u,t,D_{w1},m_{w1}) &= e^{-m_{w1}t}e^{-\xi D_{w1}u}\sin\xi D_{w1}u \\
f_{w17}(u,t,D_{w1},m_{w1}) &= \cos(m_{w1}t)e^{D_{w1}u} \\
f_{w18}(u,t,D_{w1},m_{w1}) &= \cos(m_{w1}t)e^{-D_{w1}u} \\
f_{w19}(u,t,D_{w1},m_{w1}) &= \cos(m_{w1}t)\cos D_{w1}u \\
f_{w20}(u,t,D_{w1},m_{w1}) &= \cos(m_{w1}t)\sin D_{w1}u \\
f_{w21}(u,t,D_{w1},m_{w1}) &= \sin(m_{w1}t)e^{D_{w1}u} \\
f_{w22}(u,t,D_{w1},m_{w1}) &= \sin(m_{w1}t)e^{-D_{w1}u} \\
f_{w23}(u,t,D_{w1},m_{w1}) &= \sin(m_{w1}t)\cos D_{w1}u \\
f_{w24}(u,t,D_{w1},m_{w1}) &= \sin(m_{w1}t)\sin D_{w1}u \\
f_{w25}(u,t,D_{w1},m_{w1}) &= \cos(m_{w1}t)e^{\xi D_{w1}u}\cos\xi D_{w1}u \\
f_{w26}(u,t,D_{w1},m_{w1}) &= \cos(m_{w1}t)e^{\xi D_{w1}u}\sin\xi D_{w1}u \\
f_{w27}(u,t,D_{w1},m_{w1}) &= \cos(m_{w1}t)e^{-\xi D_{w1}u}\cos\xi D_{w1}u \\
f_{w28}(u,t,D_{w1},m_{w1}) &= \cos(m_{w1}t)e^{-\xi D_{w1}u}\sin\xi D_{w1}u \\
f_{w29}(u,t,D_{w1},m_{w1}) &= \sin(m_{w1}t)e^{\xi D_{w1}u}\cos\xi D_{w1}u \\
f_{w30}(u,t,D_{w1},m_{w1}) &= \sin(m_{w1}t)e^{\xi D_{w1}u}\sin\xi D_{w1}u \\
f_{w31}(u,t,D_{w1},m_{w1}) &= \sin(m_{w1}t)e^{-\xi D_{w1}u}\cos\xi D_{w1}u \\
f_{w32}(u,t,D_{w1},m_{w1}) &= \sin(m_{w1}t)e^{-\xi D_{w1}u}\sin\xi D_{w1}u
\end{aligned}
$$

where $\xi = \sqrt{2}/2$.

For more complex scenarios, if 32 constants are insufficient for constructing deformation shapes, Eq. 4.23 can introduce $S = 1, 2, \ldots$ groups of unknown constants $c_{wi}(i = 1, 2, \ldots, 32S)$ through replacing D_{w1} and m_{w1} with D_{wS} and m_{wS}. Hence Eq. 4.23 is converted into the following function, which can be regarded as an infinite series when S approaches infinity:

$$
w(u,t) = \sum_{i=1}^{32S} c_{wi} f_{wi}(u,t,D_{wS},m_{wS}) \tag{4.24}
$$

4.3.4 Transformation Elimination

After forming the representation of $w(u,t)$ and approximating the closed-form solutions, the remaining tasks are to solve the unknown integration constants in Eq. 4.24 and then use it to create new deformation shapes.

Assuming the surface shapes of a 3D model at frame n ($n = 0, 1, \ldots, N$) are known and the time corresponds to frame n is t_n, we use w_{nk} ($w = x, y, z$) to indicate the x, y and z coordinate values of the k^{th} point ($k = 1, 2, 3, \ldots, K$) on one of the curves defining a 3D model at frame n.

When a 3D model moves from one frame to the next frame, rigid transformation may be involved. We remove rigid transformations through the following treatment.

First, we align the first point of a curve at different frames through

$$\overline{w}_{nk} = w_{nk} - (w_{n0} - w_{00}) \tag{4.25}$$

After that, we rotate the curve at frame n to align with the curve at frame 0. Assuming the general rotation matrix at frame n is:

$$\begin{bmatrix} a_{n,11} & a_{n,12} & a_{n,13} \\ a_{n,21} & a_{n,22} & a_{n,23} \\ a_{n,31} & a_{n,32} & a_{n,33} \end{bmatrix} \tag{4.26}$$

Using the above rotation matrix to align the curve at frame n to the curve at frame 0 means the following matrix multiplication:

$$\begin{bmatrix} \overline{x}_{nk} & \overline{y}_{nk} & \overline{z}_{nk} \end{bmatrix} \begin{bmatrix} a_{n,11} & a_{n,12} & a_{n,13} \\ a_{n,21} & a_{n,22} & a_{n,23} \\ a_{n,31} & a_{n,32} & a_{n,33} \end{bmatrix} \tag{4.27}$$

which changes the curve point $\overline{w}_{nk}$ into $\widetilde{w}_{nk}$ below:

$$\begin{aligned} \widetilde{x}_{nk} &= a_{n,11}\overline{x}_{nk} + a_{n,21}\overline{y}_{nk} + a_{n,31}\overline{z}_{nk} \\ \widetilde{y}_{nk} &= a_{n,12}\overline{x}_{nk} + a_{n,22}\overline{y}_{nk} + a_{n,32}\overline{z}_{nk} \\ \widetilde{z}_{nk} &= a_{n,13}\overline{x}_{nk} + a_{n,23}\overline{y}_{nk} + a_{n,33}\overline{z}_{nk} \end{aligned} \tag{4.28}$$

The elements $a_{i,j}$ ($i, j = 1, 2, 3$) in the rotation matrix are obtained by minimizing the sum of the squared errors between the points $\widetilde{w}_{nk}$ and w_{0k}, i.e.,

$$\begin{gathered} \frac{\partial}{\partial a_{n,rs}} \sum_{k=1}^{K} [(\widetilde{x}_{nk} - x_{0k})^2 + (\widetilde{y}_{nk} - y_{0k})^2 + (\widetilde{z}_{nk} - z_{0k})^2] = 0 \\ (n = 1, 2, ..., N;\ r, s = 1, 2, 3) \end{gathered} \tag{4.29}$$

After all the elements in the rotation matrix are determined, we use Eq. 4.28 to obtain $\widetilde{w}_{nk}$.

4.3.5 Approximation for Constants

In order to approximate the curve at frame n with Eq. 4.24, we should first determine the values of the parametric variable u. To do this, we find the minimum and maximum values of $\widetilde{w}_{nk}$ and denote them with $\widetilde{w}_{n,min}$ and $\widetilde{w}_{n,max}$, respectively. The u parametric value corresponding to the k^{th} point at frame n is obtained as:

$$u_{nk} = \frac{\widetilde{w}_{nk} - \widetilde{w}_{n,min}}{\widetilde{w}_{n,max} - \widetilde{w}_{n,min}} \tag{4.30}$$

Substituting t_n and u_{nk} into Eq. 4.24, we obtain the predicted values $w(u_{nk}, t_n)$. Then calculate the sum of the squared errors between the predicted values $w(u_{nk}, t_n)$ and real values $\widetilde{w}_{nk}$ for all the points on the curve for all the frames and minimizing the sum to determine the unknown integration constants involved in Eq. 4.24, i.e.,

$$\begin{aligned} &\frac{\partial}{\partial c_r} \sum_{n=0}^{N} \sum_{k=1}^{K} [w(u_{nk}, t_n) - \widetilde{w}_{nk}]^2 \\ =&\frac{\partial}{\partial c_r} \sum_{n=0}^{N} \sum_{k=1}^{K} [\sum_{i=1}^{32S} c_i f_i(u_{nk}, t_n, D_{wS}, m_{wS}) - \widetilde{w}_{nk}]^2 = 0 \\ &(r = 1, 2, ..., 32S) \end{aligned} \tag{4.31}$$

By solving the $32S$ linear equations in Eq. 4.31, we determine the $32S$ unknown integration constants. Substituting them back into Eq. 4.24, we use them to create new deformed shapes of the 3D models at new frames. In Section 4.4, we will take $S = 1$ as an example to demonstrate the application of Eq. 4.24 in dynamic reconstruction as it is the simplest scenario.

4.4 Experiment 1: Dynamic Model Reconstruction

In this experiment, our proposed dynamic reconstruction method will be compared with static modelling using B-splines and Bézier curves. We provide two instances to demonstrate the capability of our method to reconstruct complex curves while significantly reducing the number of design variables. Additionally, we present two examples of reconstructing a horse and an armadillo via our method.

Our method applies to scenarios where dynamic curve reconstruction involves determining 8 unknown constants from a series of keyframe

curves by initially setting $m_{w1} = D_{w1} = 1$, as specified in Eq. 4.19. In this experiment, we use two curves at two adjacent frames to determine the 8 unknown constants for dynamic curve reconstruction. For comparison, we also use the mathematical equations of two Bézier curves defined in (Abdel-Aziz et al. 2021) and two B-spline curves defined (Ravari and Taghirad 2016) to reconstruct the same pair of curves.

In the first curve instance, we reconstruct a closed curve and its open version at frames 20 and 100 using our proposed dynamic modelling method, as well as static modelling methods with B-splines and Bézier curves. For the time variable t_j, we normalized it to interval $[0, 1]$ by setting the 20^{th} frame as $t_0 = 0$ and the 100^{th} frame as $t_1 = 1$. Then, calculate the average error E_{aj} and maximum error E_{mj} between the N points on the original curve and the corresponding N points on the reconstructed curve as follows:

$$\begin{aligned} E_{aj} &= \frac{1}{N+1} \sum_{n=0}^{N} \frac{d_{jn}}{\overline{d}_{jn}} \\ E_{mj} &= max \left\{ \frac{d_{j0}}{\overline{d}_{j1}} \quad \frac{d_{j1}}{\overline{d}_{j1}} \quad \cdots \quad \frac{d_{jn}}{\overline{d}_{jn}} \right\} \\ & \qquad (j = 0, 1) \end{aligned} \tag{4.32}$$

where

$$\begin{aligned} d_{jn} &= \sqrt{[\sum_{i=1}^{8} c_{xi} f_x(t_j, u_n)]^2 + [\sum_{i=1}^{8} c_{yi} f_y(t_j, u_n)]^2 + [\sum_{i=1}^{8} c_{zi} f_z(t_j, u_n)]^2} \\ \overline{d}_{jn} &= \sqrt{x_{jn}^2 + y_{jn}^2 + z_{jn}^2} \\ & \qquad (j = 0, 1) \end{aligned} \tag{4.33}$$

and $f_x(t_j, u_n)$, $f_y(t_j, u_n)$ and $f_z(t_j, u_n)$ are separated from $f_w(t_j, u_n)$, which is defined in Eq. 4.23.

Fig. 4.3 shows the original and reconstructed curves by our proposed PDE-based method, Bézier curves-based method, and B-spline curve-based method. The results of open curves are on the left of six subfigures (a),(b),(c),(d),(e), and (f), while their closed versions are on the right of the six subfigures. The term **PDE** on the subfigures denotes the curve reconstructed by our proposed PDE-based dynamic reconstruction method. In Fig. 4.3, (a) is the original curves at the 20^{th} and 100^{th} frame; (b) is the reconstructed curves by our proposed method at the 20^{th} and 100^{th} frame; (c) is the reconstructed curves by Bézier-based method at

the 20^{th} and 100^{th} frame; (d) is the reconstructed curves by B-spline-based method at the 20^{th} and 100^{th} frame; (e) is the comparison among the original curves and reconstructed curves by three different methods at the 20^{th} frame; (f) is the comparison among the original curves and reconstructed curves by three different methods at the 100^{th} frame.

We can figure out that the shape of the original curves is quite complicated. From subfigures (e) and (f), it is demonstrated that our proposed method approximates the original curves quite well in visualization since the curves generated by our proposed method are slightly closer to the original curves than the ones generated by the other two methods in some local regions.

As for data size, our mathematical model only involves 8 vector-valued unknown constants, of which the design variables are only 5.7% of the original curves. This indicates the powerful capacity of accurate reconstruction for dynamic complex curves in open and closed versions with small data sizes.

The quantified comparison of design variables, average and maximum errors to the original curves, and the computational time of CPU among our proposed method, B-spline-based method, and Bézier-based method are provided in Table 4.1. The first column of the table denotes the method used for curve reconstruction. The second column stands for the type of curve, where OC is the open curve, and CC is the closed curve. PM denotes the number of all the vertices on the two curves. PBB is the number of design variables used for reconstruction. E_{A1} and E_{m1} are the average error and maximum error between the original and reconstructed curves at the 20^{th} keyframe, E_{A2} and E_{m2} are the average error and maximum error between the original and reconstructed curves at the 100^{th} keyframe, E_{A12} is the average value of the average errors at the 20^{th} and 100^{th} keyframes, E_{m12} is the bigger one of the maximum errors at the 20^{th} and 100^{th} keyframes, and all the errors in the table are multiplied by 10^{-3}. The average errors E_{A1} and E_{A2} and maximum errors E_{m1} and E_{m2} are determined by the first one and second one of Eq. 4.32, respectively.

Table 4.1 demonstrates that the average and maximum errors of our proposed dynamic reconstruction method are very small. Except for the maximum error in the open curve case, the errors are nearly identical to those of the static reconstruction methods using Bézier and B-

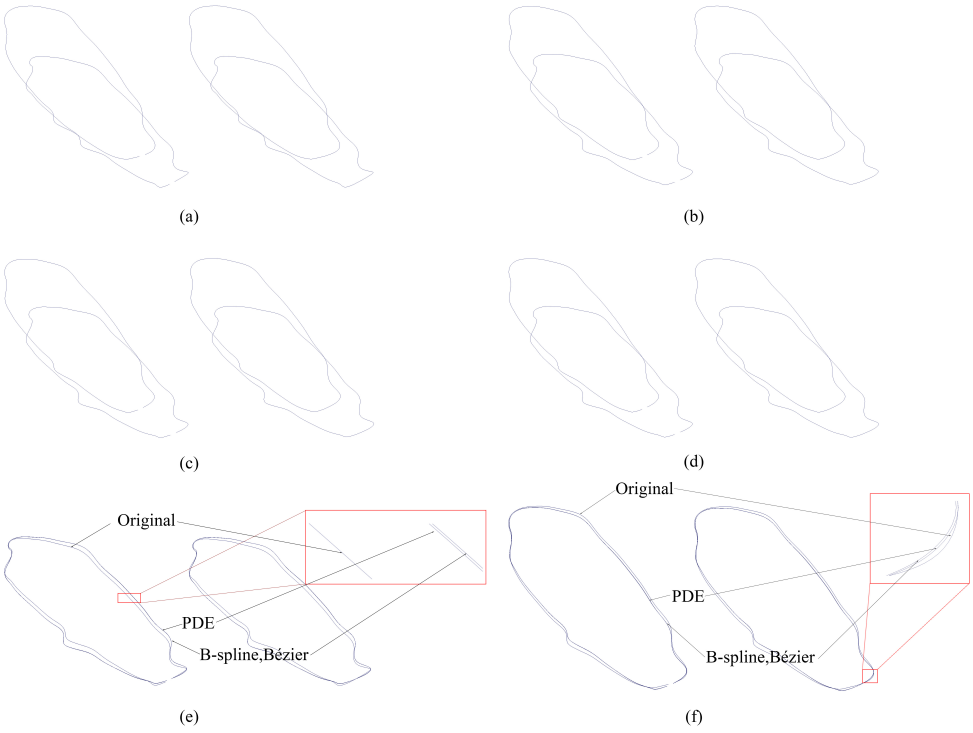


Figure 4.3: The comparison between the original and reconstructed curves with 71 points at the 20^{th} and 100^{th} frames. (a) Original curves at the 20^{th} and 100^{th} frames. (b) PDE reconstructed curves at the 20^{th} and 100^{th} frames. (c) B-spline reconstructed curves at the 20^{th} and 100^{th} frames. (d) Bézier reconstructed curves at the 20^{th} and 100^{th} frames. (e) Original, PDE, B-spline, and Bézier curves at the 20^{th} frame. (f) Original, PDE, B-spline, and Bézier curves at the 100^{th} frame.

splines. The error results from the B-spline and Bézier static reconstruction method are exactly the same. For the open curve, our method shows a maximum error of 71.60692, which is lower than the maximum error of 79.84222 from the Bézier and B-spline static reconstructions. The computation time for our proposed dynamic reconstruction method is approximately twice that of the Bézier static reconstruction method and $1.6 - 1.7$ times that of the B-spline static reconstruction method.

In the aforementioned reconstruction, the curve at the 20^{th} frame and its corresponding curve at the 100^{th} frame are quite distant from each other. Despite this, our proposed dynamic reconstruction method still achieves high accuracy. When the frames are closer together, the errors can be significantly decreased. To demonstrate this, we use two adjacent frames shown in Fig. 4.4. In this curve instance, the 37^{th} and 38^{th} frames are used, where the results are presented in Fig. 4.4, organized similarly to Fig. 4.3. From subfigures (e) and (f), we can clearly observe that the reconstructed curves from our dynamic method and the Bézier and B-spline static reconstructions are very close to the original curves.

The average and maximum errors generated by our proposed method and Bézier and B-spline static methods are given in Table 4.2, while all errors are multiplied by 10^{-4}. The average errors caused by our proposed method are very small and slightly larger than those caused by the Bézier and B-spline methods. The maximum errors of our proposed method are much smaller than those of Bézier and B-spline static methods, i.e., $8.077028 \sim 8.422275$ against $80.48240 \sim 80.73569$ and $18.39898 \sim 19.07610$ against $80.82564 \sim 81.08065$. However, the computational time of our proposed method is higher than the Bézier and B-spline methods, i.e., $1.9 \sim 2.01$ times that of the Bézier method and $1.77 \sim 1.94$ times that of the B-spline method.

The discussion above demonstrates that the errors introduced by our proposed dynamic reconstruction method are smaller than those caused by the Bézier and B-spline static reconstruction methods. Besides, unlike Bézier and B-spline static reconstructions, which do not incorporate a time variable and are thus unsuitable for dynamic or time-varying reconstructions, our proposed method effectively addresses this issue.

Additionally, we will provide two examples of dynamic 3D model reconstruction and compare them with B-spline static reconstruction. In

these comparisons, Bézier static reconstruction is slightly faster than B-spline static reconstruction but yields identical results. Therefore, Bézier static reconstruction is not considered in the subsequent examples.

The first example involves reconstructing the deformed shapes of a horse model at different keyframes obtained from PBD simulation. To achieve this, we first extract the curves to be reconstructed from the undeformed polygonal model. As shown in Fig. 4.5, the curve to be reconstructed in (b) is extracted from the undeformed polygonal horse model in (a). The original horse model has $15,389$ vertices and $30,710$ faces. Since each vertex has x, y, and z components, the original horse model has a total of $46,167$ design variables.

Table 4.1: The comparison among three methods of design variables, errors($\times 10^{-3}$), and CPU computing time for reconstruction of frame 20 and 100.

Method	**Curve Type**	**PM**	**PBB**	E_{A1}	E_{A2}	E_{A12}	E_{m1}	E_{m2}	E_{m12}	**CPU(ms)**
PDE	OC	140	8	4.641821	3.811424	4.226623	71.60692	7.954821	71.60692	5.7532
B-spline	OC	140	8	4.631861	3.813969	4.222915	79.84222	7.796782	79.84222	3.3210
Bézier	OC	140	8	4.631861	3.813969	4.222915	79.84222	7.796782	79.84222	2.8257
PDE	CC	142	8	4.624602	3.810121	4.217362	71.00726	7.470214	71.01385	3.5928
B-spline	CC	142	8	4.615302	3.811597	4.213450	71.01385	7.470214	71.01385	3.5928
Bézier	CC	142	8	4.615302	3.811597	4.213450	71.01385	7.470214	71.01385	2.9644

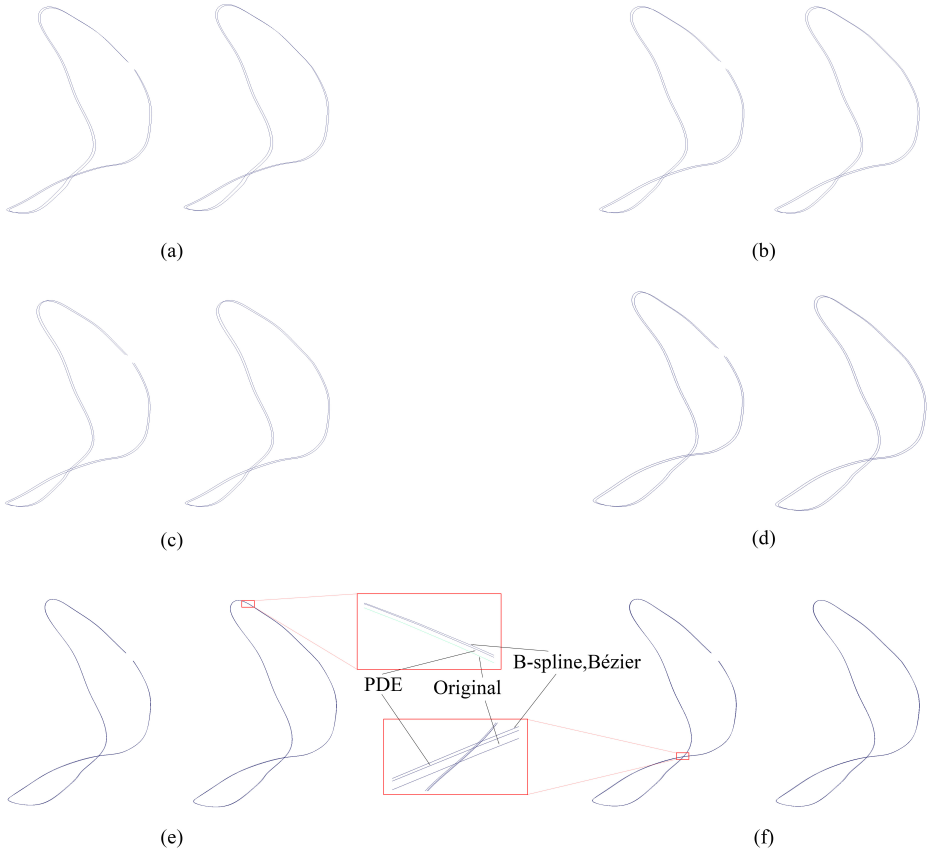
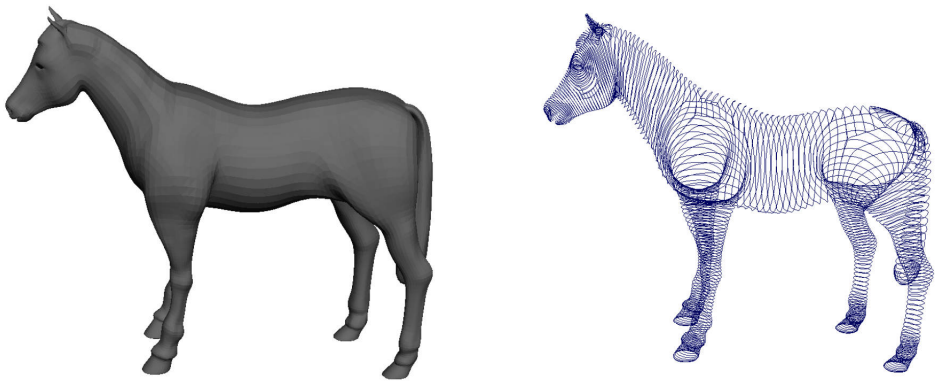


Figure 4.4: The comparison between the original and reconstructed curves with 71 points at the 37^{th} and 38^{th} frames. (a) Original curves at the 37^{th} and 38^{th} frames. (b) PDE reconstructed curves at the 37^{th} and 38^{th} frames. (c) B-spline reconstructed curves at the 37^{th} and 38^{th} frames. (d) Bézier reconstructed curves at the 37^{th} and 38^{th} frames. (e) Original, PDE, B-spline, and Bézier curves at the 37^{th} frame. (f) Original, PDE, B-spline, and Bézier curves at the 38^{th} frame.

Table 4.2: The comparison among three methods of design variables, errors($\times 10^{-4}$), and CPU computing time for reconstruction of frame 37 and 38.

Method	**Curve Type**	**PM**	**PBB**	E_{A1}	E_{A2}	E_{A12}	E_{m1}	E_{m2}	E_{m12}	**CPU(ms)**
PDE	OC	132	8	2.638812	3.142145	2.890479	8.422275	19.07610	19.07610	5.4184
B-spline	OC	132	8	2.172167	2.181193	2.176680	80.48240	80.82564	80.48240	3.0592
Bézier	OC	132	8	2.172167	2.181193	2.176680	80.48240	80.82564	80.48240	2.6934
PDE	CC	134	8	2.632932	3.169087	2.901010	8.077028	18.39898	18.39898	5.4873
B-spline	CC	134	8	2.186633	2.195735	2.191184	80.73569	81.08065	81.08065	3.1072
Bézier	CC	134	8	2.186633	2.195735	2.191184	80.73569	81.08065	81.08065	2.8346



(a) (b)

Figure 4.5: The undeformed horse model and its extracted curves. (a) Polygon horse model, (b) extracted curves.

For representing the horse model, we extracted 593 curves and used the $4,744$ vector-valued coefficients involved in Eq. 4.19 to define these curves at two different keyframes. Since each vector-valued coefficient has x, y, and z components, our proposed method uses a total of $14,232$ design variables to reconstruct the horse model at two different keyframes. Although the 593 curves were manually extracted without optimizing the number of points on each curve to reduce the number of unknowns, the extracted curves still reduce the design variables by more than two-thirds compared to the polygonal horse model.

Fig. 4.6 shows the original deformed models reconstructed using our proposed method and the B-spline method. In the figure, *PDE Model* refers to the dynamic reconstruction obtained using our proposed method, while the *B-spline Model* refers to the static reconstruction obtained using the B-spline method. The first and fourth rows display the original deformed models from the PBD simulation. The second and fifth rows show the models reconstructed by our dynamic method, and the third and sixth rows show the models reconstructed by the B-spline method.

When comparing the original models in the first and fourth rows with the reconstructed models from our dynamic method in the second and fifth rows and the B-spline method in the third and sixth rows, no significant visual differences are observed. We also calculated the average and maximum errors between the reconstructed models and the original deformed models for both methods. The average and maximum errors for reconstructing the first and tenth frames using our method are

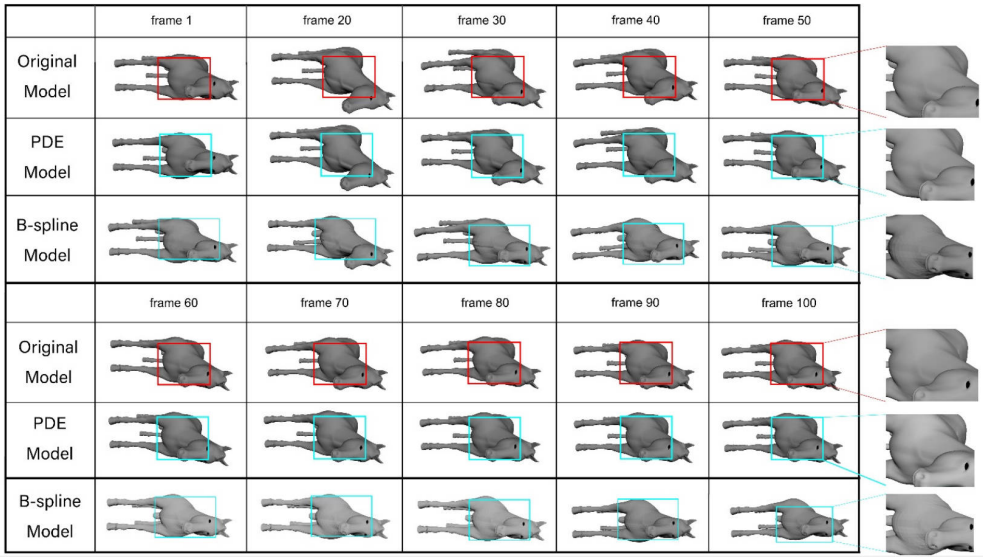


Figure 4.6: The comparison among the original deformation of the horse model generated by PBD and the reconstructed shapes created by our proposed method and B-spline method at frame $1, 20, 30, 40, 50, 60, 70, 80, 90, 100$.

6.1746×10^{-4} and 4.672209×10^{-2}, respectively. For the B-spline method, the average and maximum errors are 5.321225×10^{-4} and 5.191078×10^{-2}, respectively.

These data indicate that our proposed method achieves comparable reconstruction accuracy as the B-spline method, as both methods have very low average errors. These errors are three to four orders of magnitude lower than the differences between the maximum and minimum values of the x, y, and z coordinates, which are in the range of single or double digits. The results demonstrate that our method provides good reconstruction accuracy.

The second example involves reconstructing the deformed shapes of an armadillo polygon model obtained from position-based dynamics simulations at different keyframes. The original armadillo model has $5,182$ vertices and $5,180$ faces, resulting in $5,182$ vector-valued design variables. We extracted a total of 196 curves to represent the armadillo model and used $1,568$ vector-valued coefficients in Eq. 4.19 to define the curves at two different keyframes. Again, our method reduced the number of vector-valued design variables of the armadillo polygon model by more than two-thirds.

Fig. 4.7 shows the original deformed shapes of the armadillo polygon model, along with the shapes reconstructed using our method and the

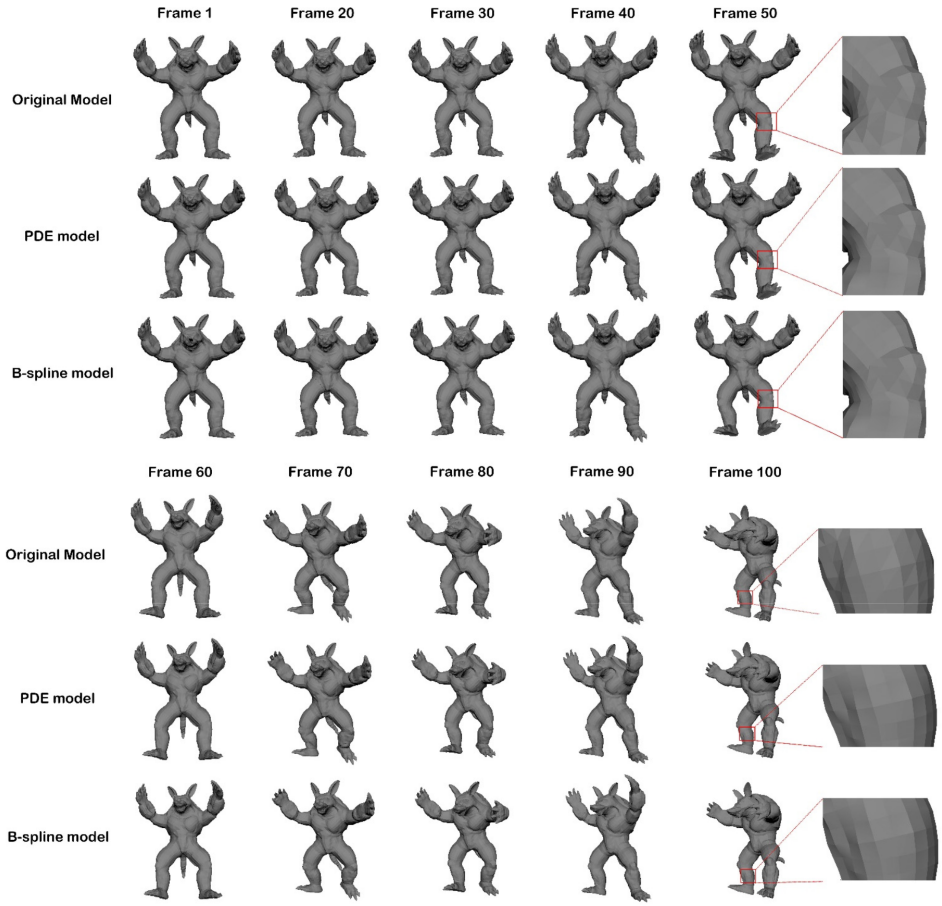


Figure 4.7: The comparison among the original deformation of the armadillo model generated by PBD and the reconstructed shapes created by our proposed method and B-spline method at frame $1, 20, 30, 40, 50, 60, 70, 80, 90, 100$.

B-spline method. Comparing the shapes reconstructed by both methods to the original deformed shapes, no significant differences were observed. This example also demonstrates that our method provides good reconstruction accuracy.

4.4.1 Summary

The first experiment utilized two instances for curve reconstruction and two examples for model reconstruction to prove the capacity of our proposed method in creating an accurate reconstruction of model deformation. It significantly reduces the data size to fit the real-time environments with realism successfully preserved.

Moreover, we calculated the average CPU time required to obtain deformed shapes using PBD for 100 keyframes of these two models. The

total average time needed for the PBD simulation was $6,420$ milliseconds. In contrast, our proposed method, by setting the time variable t in Eq. 4.19 to 98 different values, reconstructed the deformed shapes between two keyframes in average $1,827$ milliseconds, generating 98 new deformed shapes between these keyframes. Clearly, using our dynamic reconstruction method to replace some keyframes in the PBD simulation can reduce the total simulation time. On the other hand, by setting the time variable t to different values between $0 \leqslant t \leqslant 1$, new consistently deformed models can be created. These will be proven in Section 4.5.

4.5 Experiment 2: Generation of In-between Keyframe Models

As mentioned in the last part of Section 4.4, we will further explore the capacity of our proposed method in the generation of in-between keyframe models to further reduce the data required during storage and internet transmission, which is significant for creating realistic animation of digital characters in real-time environments. To describe more complex objects, the mathematical model is extended from 8 unknown constants in Eq. 4.19 to 32 unknown constants by considering all settings with different m_{w1} and D_{w1} in Eq. 4.23.

We first choose three different models from (Zhou and Jacobson 2016) to demonstrate the accuracy and efficiency of the proposed method, as shown in Fig. 4.8, whilst the cube-rope model will be used to illustrate our experiment.

In order to use the proposed algorithm to reconstruct the deformed shapes, we convert the surface models shown in Fig. 4.8 into curve-represented models. We manually extract curves along the length direction of the cube-rope model. The cube-rope model is defined by $19,968$ vertices. In total, 24 curves are extracted, and each curve has 832 points. Fig. 4.9 shows the curve-represented models at the 10 frames. Each extracted curve was then used to fit the proposed mathematical model and recreated by the method in Section 4.4. Finally, all the curves will be used to reconstruct the cube-rope models. The whole framework of this experiment is shown in Fig. 4.10 for a better understanding.

For the purpose of illustration, we take the 1^{st} extracted curve to demonstrate the dynamic reconstruction process. The same method is

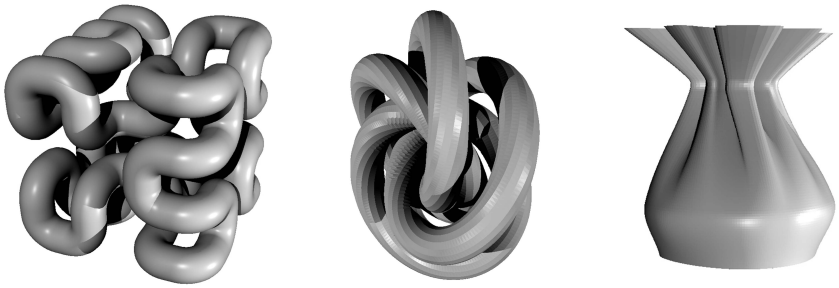


Figure 4.8: The models used for implementing our proposed method. From left to right are cube-rope, rings and dumpling. We take the cube-rope model as the example to explain our experiment and compare the errors and running time for all models with the PBD results and B-Spline.

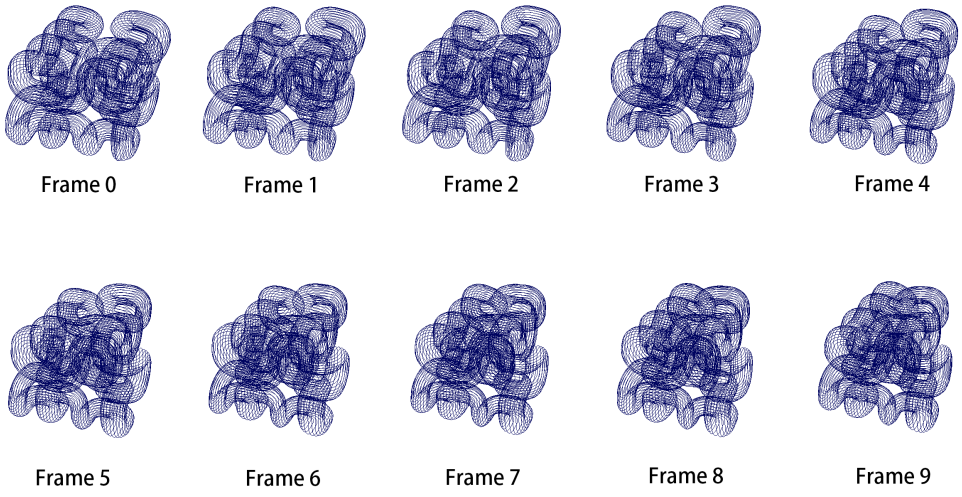


Figure 4.9: Curve representations of the cube-rope models shown in Fig. 4.8.

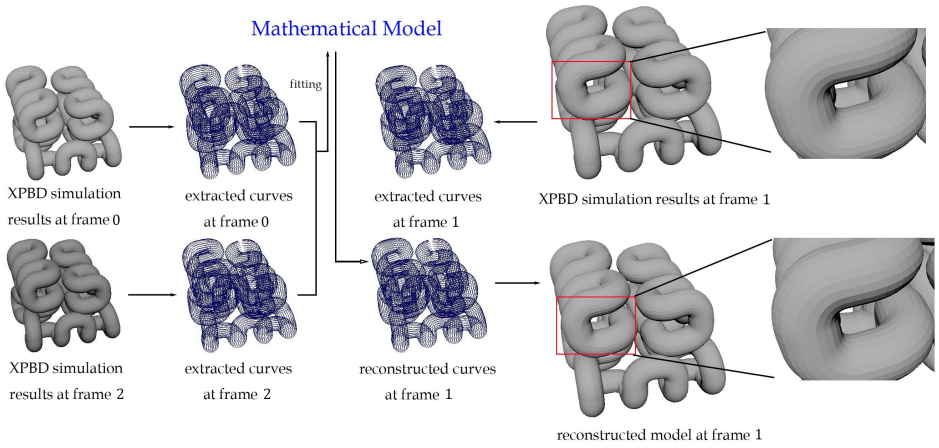


Figure 4.10: The framework of our experiment. We use simulation results generated by PBD at frames 0,2,4,6,8 and 10. After extracting the curves from the original models, we get the curves to fit our mathematical model. Then, the curve information at frames 1,3,5,7, and 9 were computed by the trained mathematical model. The calculated curves were then used to reconstruct the mesh models, which are compared with the original PBD deformation models at the same frames. As could be seen from the details on the right parts, few differences could be found between the original model from PBD and the reconstructed model by our proposed method.

applicable to the remaining 23 extracted curves.

For fitting our mathematical model along with the time variable t, we first align the first point of the 1^{st} curve at frame 1 to frame 10 with the first point of the 1^{st} curve at frame 0 through Eq. 4.25 and obtain $\hat{w}_{nk}$ $(n = 1, 2, \ldots, 10;\ k = 1, 2, \ldots, 832)$. Then we solve Eq. 4.29 to determine the elements of the rotation matrix, and use Eq. 4.28 to obtain $\widetilde{w}_{nk}$ $(n = 1, 2, \ldots, 10;\ k = 1, 2, \ldots, 832)$. After that, we use the deformed shapes at frames 0, 2, 4, 6, 8, and 10 to determine the unknown integration constants c_i in Eq. 4.23. In order to generate deformation shapes at frames 1, 3, 5, 7, and 9, we determine u_{nk} $(w = x, y, z;\ n = 1, 3, 5, 7, 9;\ k = 1, 2, \ldots, 832)$ at frames 1, 3, 5, 7, and 9 with linearly interpolating u_{nk} of two adjacent frames of frames 0, 2, 4, 6, 8, and 10, and use Eq. 4.23 to calculate the deformed shapes of one baseline curve at frames $0, 1, 2, 3, \ldots, 10$, depict them as red curves in Fig. 4.11, where the baseline curves are in blue, and obtain $w(u_{nk}, t_n)$ $(w = x, y, z;\ n = 0, 1, 2, \ldots, 10;\ k = 1, 2, \ldots, 832)$.

For the static reconstruction with B-Spline curves and linear interpolation, we use the normal cubic B-Spline method with 50 control points

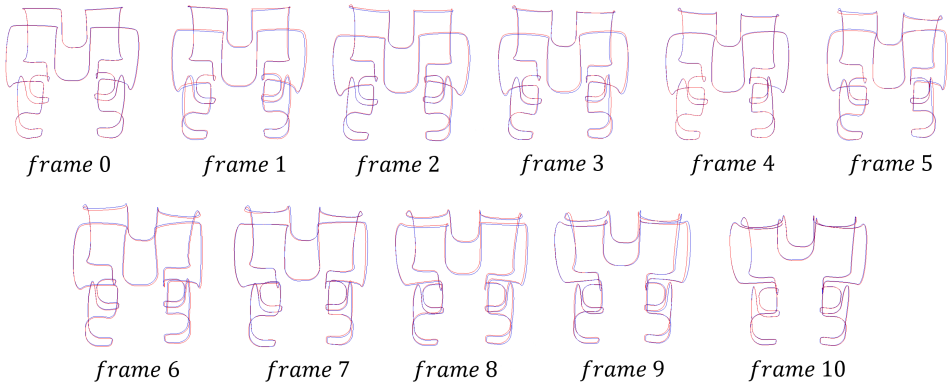


Figure 4.11: Visual comparison among the baseline curves highlighted in blue and reconstructed curves with the proposed algorithm highlighted in red and B-Spline curves combined with linear interpolation highlighted in pink.

provided in (Prautzsch et al. 2002) to fit the model and obtain B-Spline curves at frames 0, 2, 4, 6, 8, and 10. Then, we linearly interpolate two adjacent B-Spline curves at frames 0, 2, 4, 6, 8, and 10 to obtain the B-Spline curves at frames 1, 3, 5, 7, and 9, depict B-Spline curves at frame $0, 1, 2, 3, \ldots, 10$ as pink curves in Fig. 4.11 as well, and use the interpolated B-Spline curves to calculate coordinate values $w(u_{nk})$ $(w = x, y, z;\ n = 0, 1, \ldots, 10;\ k = 1, 2, \ldots, 832)$.

After that, we quantify the errors between the baseline shapes $\widetilde{w}_{nk}$ and the reconstructed shapes $w(u_{nk}, t_n)$ obtained from the proposed algorithm and B-Spline method through the following equations:

$$E_{An} = \frac{1}{832D} \sum_{k=0}^{832} \|\widetilde{w}_{nk} - w(u_{nk}, t_n)\| \qquad (n = 0, 1, 2, \ldots, 10) \qquad (4.34)$$

$$E_{Mn} = \frac{1}{D} max\{\|\widetilde{w}_{nk} - w(u_{nk}, t_n)\|\} \qquad (n = 0, 1, 2, \ldots, 10;\ k = 1, 2, \ldots, 832) \qquad (4.35)$$

where E_{An} is the average error, E_{Mn} is the maximum error, D is the distance between the two furthest points of the 1st extracted curve.

We use the same way to calculate the errors for all three models. The rings model has $18,468$ vertices, formed by 16 curves, and the dumpling model has $24,847$ vertices, formed by 251 column curves. Among the three models, the curves on the rings model are the most simple, whilst those on the dumpling model are quite complex and difficult to fit with small design parameters. These qualitative observations are corroborated by the quantitative results presented in Table 4.3. The results provide

a visual comparison between our proposed method, the B-Spline and linear interpolation method, and the baseline shapes. Our reconstructed curves exhibit minimal differences with the baseline curves generated by PBD, except for some parts when the curvature fluctuates. Conversely, the curves produced by the B-Spline and linear interpolation method display noticeable deviations from the baseline curves. As could be seen from Table 4.3, for the first model (cube-rope), the average E_{An} and the maximum E_{Mn} of our proposed method are 0.006986 and 0.019478, whilst those of B-Spline method are 0.046644 and 0.270942. The accuracy improvement of E_{An} is obvious, with about 52.72% for averaging all models. The maximum E_{Mn} for all models decreases 21.33% , from 0.2709 to 0.2132.

The CPU used to generate the 11 frames is 7.442 seconds and frames 0, 2, 4, 6, 8, and 10 is 4.085 seconds by PBD for the cube-rope model. Using frames 0, 2, 4, 6, 8, and 10 to determine the unknown integration constants in Eq. 4.23 and generate reconstructed shapes at frame 1, 3, 5, 7 and 9 with Eq. 4.23 is 0.293 seconds. Using frame 0, 2, 4, 6, 8, and 10 to determine the unknown control points in the B-Spline method for reconstructing B-Spline curves at frames 0, 2, 4, 6, 8, and 10 and linearly interpolating the reconstructed B-Spline curves to obtain B-Spline curves at frame 1, 3, 5, 7 and 9 is 0.034 seconds. Plus 4.085 seconds used by PBD to generate frames 0, 2, 4, 6, 8, and 10, our proposed dynamic reconstruction algorithm takes 4.378 seconds to generate the shapes at the 11 frames (frame 0 to frame 10), and B-Spline and linear interpolation method take 4.119 seconds to generate the shapes at the 11 frames. The above data indicate our proposed dynamic reconstruction algorithm combined with the PBD simulation only requires 58.13% of the time required by PBD to generate the shapes at the 11 frames, indicating our proposed dynamic reconstruction algorithm combined with PBD simulation can greatly reduce computational time. Although the total time required by our proposed dynamic reconstruction algorithm is a little more than the total time required by B-Spline curves and linear interpolation, our proposed dynamic reconstruction algorithm has higher accuracy, involves fewer design variables, and is physics-based.

Table 4.3: Average errors and maximum errors of our method and B-Spline compared with PBD results. The three models with different complexity of curves are all better fitted with our proposed method with less error of both average error E_{An} and maximum error E_{Mn}.

	Cube-Rope				**Rings**				**Dumpling**			
	PDE		**B-Spline**		**PDE**		**B-Spline**		**PDE**		**B-Spline**	
	E_{An}	E_{Mn}	E_{An}	E_{Mn}	E_{An}	E_{Mn}	E_{An}	E_{Mn}	E_{An}	E_{Mn}	E_{An}	E_{Mn}
0	0.003035	0.005387	0.009169	0.110480	0.003962	0.058175	0.01364	0.044137	0.067166	0.213153	0.056047	0.265555
1	0.007886	0.017251	0.054134	0.187488	0.004284	0.063326	0.014811	0.04863	0.066267	0.204672	0.135865	0.254741
2	0.009044	0.014629	0.090012	0.111336	0.004683	0.066514	0.016719	0.05098	0.065469	0.196021	0.102285	0.248653
3	0.008655	0.019478	0.077795	0.143668	0.004277	0.063076	0.014775	0.048404	0.064333	0.189016	0.098727	0.248982
4	0.005158	0.008127	0.024759	0.2525	0.003483	0.023472	0.008039	0.027108	0.063253	0.183081	0.093986	0.245024
5	0.007315	0.01675	0.062312	0.199951	0.003755	0.025922	0.0083	0.02889	0.062121	0.177521	0.092062	0.239388
6	0.007807	0.018031	0.035425	0.245637	0.004059	0.027469	0.008054	0.026568	0.059731	0.165742	0.082506	0.228314
7	0.006296	0.013386	0.034231	0.106627	0.003747	0.025676	0.008265	0.028796	0.057284	0.154663	0.081785	0.2175
8	0.010102	0.016598	0.037255	0.181635	0.005324	0.045913	0.022571	0.066987	0.055027	0.147464	0.089405	0.205412
9	0.008118	0.016756	0.055076	0.169511	0.00553	0.036237	0.019776	0.048332	0.052616	0.142232	0.082467	0.194223
10	0.003429	0.015639	0.032914	0.270942	0.006366	0.054327	0.024832	0.081252	0.049931	0.136109	0.081886	0.183327

These results indicate that our proposed dynamic reconstruction algorithm can generate in-between keyframes with higher efficiency and accuracy compared with the B-spline method.

4.5.1 Summary

In this experiment, we extend the design variables from 8 to 32 for describing more complex models, according to Eq. 4.23. After comparing our method and the reconstruction with B-Spline curves and linear interpolation between frames, the results demonstrate that our proposed method further raises the efficiency of the dynamic deformation simulation and reduces data sizes while keeping good accuracy. More importantly, our proposed method can efficiently generate continuous deformation shapes, which are not generated by PBD and require less time for PBD to generate them.

4.6 Conclusion

In this chapter, we propose a PDE-based method for fast and precise model reconstruction in dynamic deformation simulation. By integrating the governing equation, which depicts the bending deformation of elastic beams, into Newton's second law, which depicts the object movement, we present a novel mathematical model. The closed-form solution of this mathematical model is obtained by utilizing the separation of variables technique and the least square method. We have combined four different closed-form solutions of our proposed mathematical model for dynamic deformation simulation, and make it able to be combined with infinity terms, significantly raising the capacity of PDE-based dynamic reconstruction. Besides, by normalizing the time variable t in the interval $[0, 1]$ and introducing it into the vector-valued PDE of the mathematical model, we bring the model capacity to generate in-between keyframe models with high efficiency and accuracy. With this time-dependent attribute, many keyframes simulated by PBD can be replaced with the ones generated by our method, so that the computational time cost on the PBD simulation can be saved to further enhance its real-time performance. Besides, our method is also able to generate skin deformation results by the proposed mathematical model. The experiments, which

integrate with PBD and compare with other baseline methods, demonstrate that our proposed method can reconstruct deformable models with small data sizes but retain most details. This greatly benefits the target of this research in creating realistic animations of digital characters in interactive systems.

Chapter 5

Skin deformation Method Based on Newton's Second Law for Generating Natural Human Facial Blendshapes

As introduced in Chapter 1 and 2, skin deformation techniques have become a standardized method for creating real-time animations in both the animation industry and academic research. Across various skin deformation methods, the primary focus remains on achieving realism and efficiency. To achieve realism, it is necessary to store a high level of details, including sufficient vertices, which often results in significant redundancy that is not frequently utilized in calculations. Geometric methods continue to be widely used for determining skin deformation. On the other hand, physics-based techniques, while capable of enhancing realism, require extensive numerical computations, consuming considerable CPU resources and thus reducing efficiency. As the most important and widely researched topic, digital humans have been focused, among all research for digital character animation. Thus the human facial blendshapes issue is addressed in this chapter as one of the skin deformation problems.

Inspired by the ODE-based sweeping surface methods (You et al. 2007) and ODE-based surface blending method (You et al. 2014), we propose a new physics-based method to generate natural facial blendshapes from ODE-based models for fulfilling the goal of this research in generating realistic animation of digital characters. Our method has the following contributions:

- Develop an ODE-based surface creation method to reconstruct 3D face models with small data sizes from extracted curves, which are extracted from reference models.
- Provide an interpolation method based on Newton's second law, which has the capacity to generate in-between curves from the curves on two sides.
- Combine the ODE-based surface creation method and physics-based interpolation method to efficiently create natural facial animation.

5.1 ODE-based Surface Creation

The foundation for the 3D model surface creation method in this chapter comes from (Bian et al. 2019), where this section will give a concise introduction to it.

5.1.1 Mathematical Model

The mathematical model of ODE-based sweeping surfaces consists of a vector-valued equation $w = S(u, v)$ along with boundary conditions, where $w = (x, y, z)$ denotes the positions of vertices on surface S, which is defined by two normalized parametric variables u and v. Fixing v at v_i, the surface S is converted to a curve $C_i = S(u, v_i)$. The form of the ODE and the boundary conditions vary depending on the continuity requirements. For creating smooth and controllable surfaces, we aim to fulfil C^2 continuity between adjacent surface patches, including position, tangent, and curvature continuities. These control parameters allow for fine-tuning the details of the facial patches. Thus, the boundary constraints contain 3 different parts for each $u = 0, 1$:

$$\begin{aligned} u = 0 \quad & S(0, v) = c_1(v), \quad \partial S(0, v)/\partial v = c_2(v), \quad \partial S^2(0, v)/\partial v^2 = c_3(v) \\ u = 1 \quad & S(1, v) = c_4(v), \quad \partial S(1, v)/\partial v = c_5(v), \quad \partial S^2(1, v)/\partial v^2 = c_6(v) \end{aligned} \tag{5.1}$$

where $c_j(v)$ $(j = 1, 2, \ldots, 6)$ are unknown functions.

For achieving all the boundary constraints, a vector-valued sixth-order ODE is employed as follows, as it involves 6 unknown constants, which represent six continuity values:

$$\rho \frac{d^6 S(u, v_i)}{du^6} + \eta \frac{d^4 S(u, v_i)}{du^4} + \lambda \frac{d^2 S(u, v_i)}{du^2} = 0 \tag{5.2}$$

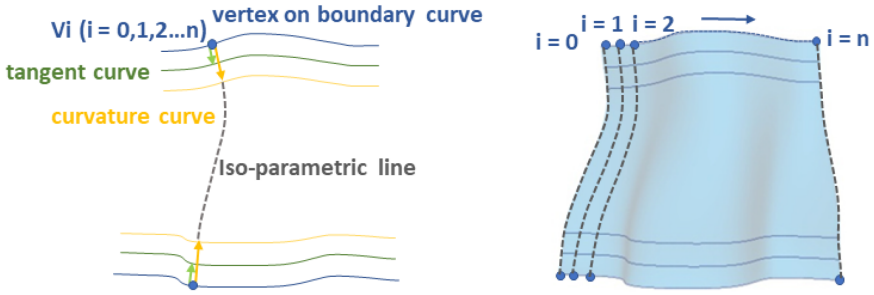


Figure 5.1: ODE-based surface creation by two boundary and four control curves in (Bian et al. 2019).

where ρ, η and λ are shape control parameters, $S(u, v_i)$ stands for the i^{th} curve C_i of the surface S.

Fig. 5.1 shows the creation of a surface by forming an iso-parametric curve with six corresponding vertices from the two boundary curves and four control curves.

5.1.2 Closed-form Solution

The sixth-order ODE in Eq. 5.2 can be converted into a fourth-order ODE by using the method of substitution method to reduce order as:

$$\begin{gathered} \rho\frac{d^4\bar{S}(u,v_i)}{du^4} + \eta\frac{d^2\bar{S}(u,v_i)}{du^2} + \lambda\bar{S}(u,v_i) = 0 \\ \bar{S}(u,v_i) = \frac{d^2S(u,v_i)}{du^2} \end{gathered} \tag{5.3}$$

Similar to the processing of Eq. 4.10 in Section 4.3, substituting $\bar{S}(u, v_i) = e^{ru}$ into Eq. 5.3, we can get this quartic equation:

$$\rho r^4 + \eta r^2 + \lambda = 0 \tag{5.4}$$

Further introducing $t = r^2$, Eq. 5.4 can be rewritten as a quadratic equation:

$$\rho t^2 + \eta t + \lambda = 0 \tag{5.5}$$

of which roots are

$$t_{1,2} = -\eta\frac{1 \pm \sqrt{1 - 4\rho\lambda/\eta^2}}{2\rho} \tag{5.6}$$

Taking the situation $4\rho\lambda/\eta^2 < 1$ as an example, we can get the roots of Eq. 5.4:

$$r_{1,2} = \pm i\xi_1 \qquad r_{3,4} = \pm i\xi_2 \tag{5.7}$$

where $\xi_{1,2} = \sqrt{\eta \frac{1 \pm \sqrt{1-4\rho\lambda/\eta^2}}{2\rho}}$.

Thus, the final solution to Eq. 5.2 can be gained as:

$$S(u, v_i) = b_1 \cos \xi_1 u + b_2 \sin \xi_1 u + b_3 \cos \xi_2 u + b_4 \sin \xi_2 u + b_5 u + b_6 \quad (5.8)$$

where $b_k (k = 1, 2, \ldots, 6)$ are vector-valued unknown constants.

Combining Eq. 5.8 and Eq. 5.2, the surface S with boundary constraints of six control curves can be defined as:

$$\begin{aligned} S(u, v) = {} & g_1(u)c_1(v) + g_2(u)c_2(v) + g_3(u)c_3(v) \\ & + g_4(u)c_4(v) + g_5(u)c_5(v) + g_6(u)c_6(v) \end{aligned} \quad (5.9)$$

where

$$\begin{aligned} g_1(u) = {} & -d_1 \cos \xi_1 u - d_4 \sin \xi_1 u + d_1(e_{11} + 1) \cos \xi_2 u \\ & + e_9 \sin \xi_2 u - (\xi_2 e_9 - \xi_1 d_4)u - (e_{11} d_1 - 1) \\ g_2(u) = {} & -(d_1 + d_2) \cos \xi_1 u - (d_3 + d_4) \sin \xi_1 u + (e_9 + e_{10}) \sin \xi_2 u \\ & - [\xi_2(e_9 + e_{10}) - \xi_1(d_3 + d_4) - 1]u - e_{11}(d_1 + d_2) \\ g_3(u) = {} & -d_5 \cos \xi_1 u - d_7 \sin \xi_1 u + d_{10} \cos \xi_2 u + (e_5 e_9 + e_6 e_1 0 + e_1 2/\xi_2) \\ & \cdot \sin \xi_2 u - [\xi_2(e_5 e_9 + e_6 e_{10}) - \xi_1 d_7 + e_{12}]u - (e_{11} d_5 - 1/\xi_2^2) \\ g_4(u) = {} & d_1 \cos \xi_1 u + d_4 \sin \xi_1 u - d_1(e_{11} + 1) \cos \xi_2 u \\ & - e_9 \sin \xi_2 u + (\xi_2 e_9 - \xi_1 d_4)u + e_{11} d_1 \\ g_5(u) = {} & d_2 \cos \xi_1 u + d_3 \sin \xi_1 u - d_2(e_{11} + 1) \cos \xi_2 u \\ & - e_{10} \sin \xi_2 u + (\xi_2 e_{10} - \xi_1 d_3)u + e_{11} d_2 \\ g_6(u) = {} & -d_6 \cos \xi_1 u - d_8 \sin \xi_1 u + d_6(e_{11} + 1) \cos \xi_2 u + (e_7 e_9 + e_8 e_{10} \\ & - e_{13}/\xi_2) \sin \xi_2 u - [\xi_2(e_7 e_9 + e_8 e_{10}) - \xi_1 d_8 - e_{13}]u - e_{11} d_6 \end{aligned} \quad (5.10)$$

$$\begin{aligned} d_1 &= e_1/(e_1 e_3 - e_2 e_4) & d_2 &= -e_2/(e_1 e_3 - e_2 e_4) \\ d_3 &= e_3/(e_1 e_3 - e_2 e_4) & d_4 &= -e_4/(e_1 e_3 - e_2 e_4) \\ d_5 &= d_1 e_5 + d_2 e_6 & d_6 &= d_1 e_7 + d_2 e_8 \\ d_7 &= d_4 e_5 + d_3 e_6 & d_8 &= d_4 e_7 + d_3 e_8 \\ d_9 &= (e_{11} + 1)(d_1 + d_2) & d_{10} &= -1/\xi_2^2 + (e_{11} + 1) d_5 \end{aligned} \quad (5.11)$$

and

$$
\begin{aligned}
e_1 &= \xi_1(\cos\xi_1 - 1) + \xi_1^2 \sin\xi_1(1-\cos\xi_2)/(\xi_2 \sin\xi_2)\\
e_2 &= \sin\xi_1 - \xi_1 + \xi_1^2 \sin\xi_1(1/\sin\xi_2 - 1/\xi_2)/\xi_2\\
e_3 &= \cos\xi_1 - 1 + \xi_2^2(1-\cos\xi_2)/\xi_2^2 + \xi_1^2(\cos\xi_2 - \cos\xi_1)(1/\xi_2 - 1/\sin\xi_2)/\xi_2\\
e_4 &= \xi_1(-\sin\xi_1 + \xi_1 \sin\xi_2/\xi_2) + \xi_1^2(\cos\xi_2 - \cos\xi_1)(\cos\xi_2 - 1)/(\xi_2 \sin\xi_2)\\
e_5 &= (1/\xi_2 - \cos\xi_2/\sin\xi_2)/\xi_2\\
e_6 &= (\sin\xi_2 + \cos^2\xi_2/\sin\xi_2 - \cos\xi_2/\sin\xi_2)/\xi_2\\
e_7 &= (1/\sin\xi_2 - 1/\xi_2)/\xi_2\\
e_8 &= (1-\cos\xi_2)/(\xi_2 - \sin\xi_2)\\
e_9 &= (\xi_1^2)[d_4 \sin\xi_1 - d_1(\cos\xi_2 - \cos\xi_1)]/(\xi_2^2 \sin\xi_2)\\
e_{10} &= (\xi_1^2)[d_3 \sin\xi_1 - d_2(\cos\xi_2 - \cos\xi_1)]/(\xi_2^2 \sin\xi_2)\\
e_{11} &= \xi_1^2/\xi_2^2 - 1\\
e_{12} &= \cos\xi_2/(\xi_2 \sin\xi_2)\\
e_{13} &= 1/(\xi_2 \sin\xi_2)
\end{aligned}
\tag{5.12}
$$

5.2 Facial Blendshape based on Newton's Second Law

In this section, we integrate Newton's second law to present a new, efficient, and physics-based facial blendshape method. We will start by introducing Newton's second law, then deriving its exact closed-form solution, and finally determining the unknown constants involved in this solution.

5.2.1 Algorithm

Newton's second law can be defined in a 3D space as:

$$m\mathbf{a} = \mathbf{f} \tag{5.13}$$

where m is the mass, $\mathbf{a} = (a_x, a_y, a_z)$ is the acceleration, and $\mathbf{f} = (f_x, f_y, f_z)$ denotes the external force.

The acceleration $\mathbf{a}$ is the second derivative of the displacement $\mathbf{x} = (x, y, z)$ with respect to time t:

$$\mathbf{a} = \frac{d^2\mathbf{x}}{dt^2} \tag{5.14}$$

Substituting Eq. 5.14 into Eq. 5.13, we can get:

$$m\frac{d^2\mathbf{x}}{dt^2} = \mathbf{f} \tag{5.15}$$

Then, we establish the initial and boundary conditions for solving the aforementioned second-order ordinary differential equation by considering the position of the initial and final postures, as well as the velocity of the initial posture.

Considering a 3D model consists of a group of N vertices. We use $\bar{\mathbf{x}}_n(n = 1, 2, \ldots, N)$ to denote the vertices at the initial pose, where $t = 0$, and $\bar{\bar{\mathbf{x}}}_n(n = 1, 2, \ldots, N)$ to denote the vertices at the final pose, where $t = 1$, to normalize t in the interval $[0, 1]$. Thus, the position change $\mathbf{x}$ is $\mathbf{x} = 0$ when $t = 0$, and is $\mathbf{x} = \bar{\bar{\mathbf{x}}}_n - \bar{\mathbf{x}}_n$ when $t = 1$.

The initial and boundary conditions can be given as:

$$\begin{aligned} t = 0 \quad & \mathbf{x}_n = 0 \\ & \frac{d\mathbf{x}_n}{dt} = 0 \\ t = 1 \quad & \mathbf{x}_n = \bar{\bar{\mathbf{x}}}_n - \bar{\mathbf{x}}_n \end{aligned} \tag{5.16}$$

Substituting $\mathbf{x}$ with $\mathbf{x}_n$ in Eq. 5.15, we have:

$$m\frac{d^2\mathbf{x}_n}{dt^2} = \mathbf{f}_n \tag{5.17}$$

Integrating Eq. 5.17 with respect to t twice, we can gain:

$$\mathbf{x}_n = \frac{\mathbf{f}_n}{2m}t^2 + c_0 t + c_1 \tag{5.18}$$

where c_0 and c_1 are unknown constants.

Substituting Eq. 5.18 into Eq. 5.16, we can solve the two constants $c_0 = c_1 = 0$, and the external force can be defined as:

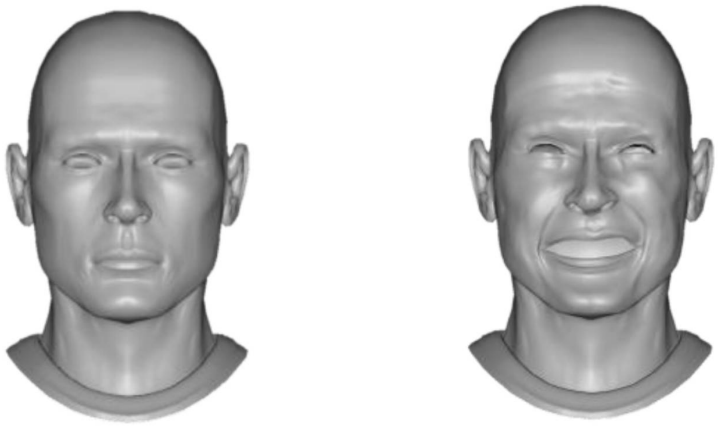
$$\mathbf{f}_n = 2m(\bar{\bar{\mathbf{x}}}_n - \bar{\mathbf{x}}_n) \tag{5.19}$$

Then, introducing Eq. 5.19 into Eq. 5.18, the position change $\mathbf{x}$ can be calculated as:

$$\mathbf{x}_n = (\bar{\bar{\mathbf{x}}}_n - \bar{\mathbf{x}}_n)t^2 \tag{5.20}$$

and the position values $\hat{\mathbf{x}}_n$ at any pose between the initial and final pose can be determined as:

$$\hat{\mathbf{x}}_n = \bar{\mathbf{x}}_n + (\bar{\bar{\mathbf{x}}}_n - \bar{\mathbf{x}}_n)t^2 \tag{5.21}$$



(a) Neutral Model (b) Laugh Model

Figure 5.2: Two face models with the same topology but different poses, achieved from (Sumner and Popović 2004). The left is a neutral model, while the right is a laugh model.

5.2.2 Experiment

Compared to the animation of other parts of digital characters, facial deformation is more critical as it requires higher precision to be perceived as realistic. To validate the experimental results, we used the model library provided in (Sumner and Popović 2004) and selected two different face models with the same topology, as shown in Fig. 5.2. First, we apply the ODE-based method introduced in Section 5.1 to reconstruct the two models. Then, we use both the linear interpolation method and our proposed method to compute the keyframes between them. Finally, we created an animation transitioning between the two different face models to demonstrate the implementation of our proposed approach and its application to facial deformation animation.

5.2.2.1 Patch Segmentation

There has been various research for parametrizing facial expressions. Ekman and Friesen (1978) proposed the Facial Action Coding System (FACS), which uses corresponding parameters to control different facial shapes. Hamm et al. (2011) presented that facial appearance can encode the movement of each facial muscle, with slight variations in commands. Furthermore, any anatomically possible facial expression can be encoded by FACS. By breaking facial expressions down into temporal segments and action units, it can be further interpolated to fit any intelligent

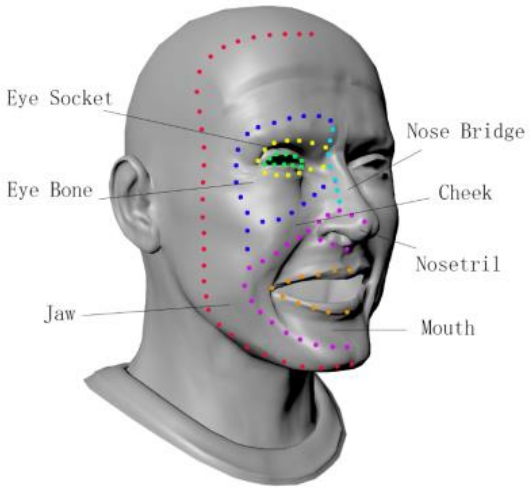


Figure 5.3: Face patch segmentation.

environment, as discussed in (Freitas-Magalhães 2013). More recently, the Skinned Multi-Person Linear Model (SMPL) has been proposed by Loper et al. (2023), which learns the relationship between mesh vertices, presenting shapes, and joints, describing poses. It utilizes the relationship to control facial expression by manipulating facial joints. Unlike these methods, we leverage wireframes to define 3D models and recreate 3D models from these wireframes using the ODE-based surface creation method. However, face patch segmentation is required for all approaches due to the complexity of facial surfaces.

In this experiment, the face model is separated into 7 patches, including eye socket, eye bone, nose bridge, nostril cheek, mouth, and jaw, as shown in Fig. 5.3.

5.2.2.2 Surface Reconstruction

We first extract all the boundary curves of each face patch, as well as add two control curves to each boundary curve by the method introduced in Section 5.1. These control curves define the tangents and curvatures on the extracted boundary curves, which correspond to the first and second derivatives. Fig. 5.4 shows the boundary curves and their control curves on the left and the resulting curves from the ODE surface creation method on the right. We compare the original model with the reconstructed surface of the neutral model in Fig. 5.5(a) and the laughing model in Fig. 5.5(b). In Fig. 5.5(a) and Fig. 5.5(b), the left models are the original model, while the right models are reconstructed models.

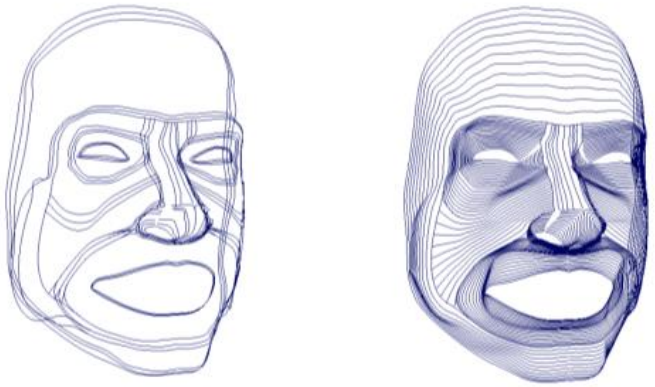
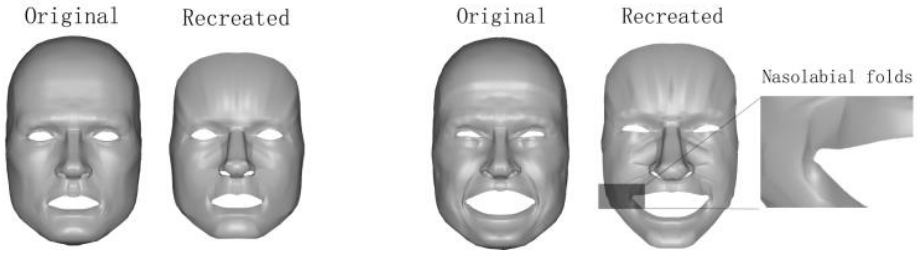


Figure 5.4: Extracted wireframes of each face patch. The left is the six control curves of each patch, and the right is all curves generated by the ODE-based surface creation method using extracted control curves.



(a) Neutral Pose (b) Laugh Pose

Figure 5.5: Comparison of the original and recreated models reveals that the original laughing model contains detailed features like nasolabial folds, which are effectively restored using the ODE-based surface recreation. The ODE sweeping technique preserves significant details by manipulating the control curves.

Despite using a minimal number of patches to create the ODE-scanned surface representation of the 3D model, the reconstructed facial model (right images in Fig. 5.5(a) and Fig. 5.5(b)) retains most details compared to the original facial model (left images in Fig. 5.5(a) and Fig. 5.5(b)). As shown in Table 5.1, the ODE-based reconstructed models is defined by $4,764$ vertices on boundary and control curves along with three shape control parameters defined in Eq. 5.2. This results in the reconstructed model having only about 58% of the design variables compared to the original models with $8,821$ and $8,246$ vertices, significantly saving data storage space while maintaining similar quality. Additionally, the constructed model offers a convenient method for quickly manipulating facial shapes and expressions by adjusting boundary curves, control curves, and shape control parameters. Besides, as adjacent ODE patches share the same boundary conditions, C^2 continuity is naturally achieved, eliminating the need for manual stitching and saving substantial time in ensuring smooth transitions between patches.

Table 5.1: The numerical comparison between the original models and the reconstructed models.

	Model Type	**Original Model**	**Recreated Model**	**Reduce Rate**
vertices number	neutral	8221	4764	42.05%
	laugh	8246	4764	42.23%
polygon number	neutral	15378	4236	72.45%
	laugh	15411	4236	72.51%

5.2.2.3 Facial Blendshape

Applying vertex interpolation methods between polygon models with different poses can achieve facial blendshapes. However, establishing correspondences between models with different topologies or vertex numbers can be labour-intensive and time-consuming. The use of numerous polygonal vertices to define these models further increases the computational cost for interpolating between models.

By reconstructing models using the ODE-based surface creation method, we can achieve facial blendshapes through interpolating boundary and control curves rather than vertices. This approach eliminates the need

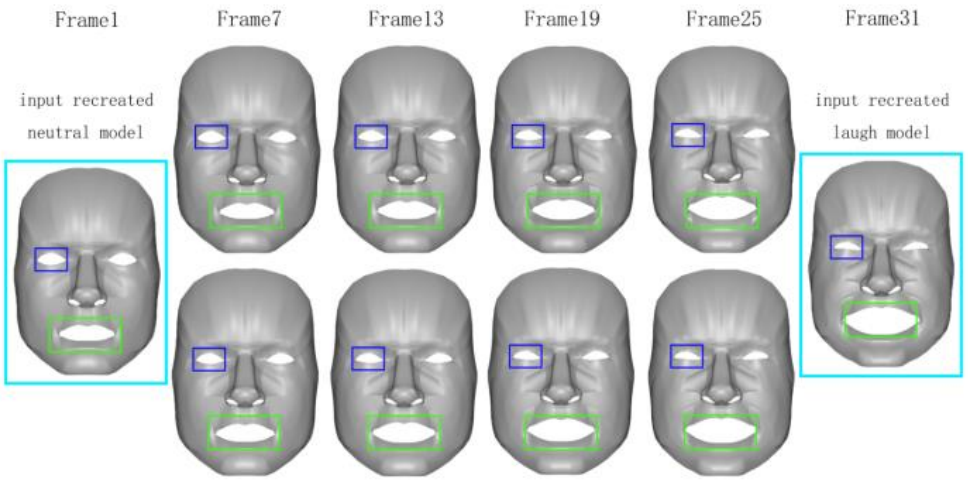


Figure 5.6: Comparison between 29 reconstructed keyframe models by our proposed method and LBS. We use the green and blue boxes to emphasize the shape alteration of the mouth and eyes.

to establish correspondences between models with different topologies or vertex numbers, and reduces the time required to interpolate multiple polygonal vertices.

Unlike the widely used purely geometric linear interpolation for facial blendshapes like LBS (Magnenat et al. 1988), our method employs Newton's second law, making it physics-based. This has the potential to create more natural-looking facial blendshapes. We have implemented both our proposed algorithm and LBS, and used them to create facial blend shapes for demonstrating the advantages of our proposed method.

The facial blendshape results are shown in Fig. 5.6, where the left and right models are original models with the neutral pose and laugh pose. The models on the first raw are reconstructed models by the ODE-based surface creation method, while the models on the second raw are reconstructed by LBS. There are 31 keyframe models created in total, whilst only the 7^{th}, 13^{th},19^{th}, and 25^{th} keyframe models are displayed for comparison.

Through observing Fig. 5.6, we can find that facial blend shapes based on linear interpolation produce uniform changes between two adjacent models, resulting in expression changing at a constant speed. However, in reality, facial movements do not occur at a uniform speed. For instance, when opening or closing the mouth, the lips may accelerate or decelerate at the beginning or end of the motion. This acceleration and deceleration can create more natural lip movements. By using inter-

polation derived from Newton's second law, we can easily achieve these acceleration effects, resulting in more natural-looking blend shapes.

5.3 Conclusion

In this chapter, we developed a physics-based skin deformation method that integrates an improved ODE surface creation framework with Newton's second law to generate realistic facial animations. By leveraging this approach, we addressed several key challenges in digital character animation, such as achieving visually natural deformations, maintaining edge continuity, and ensuring computational efficiency. The ODE-based surface creation method allows for smooth and physically plausible transitions between different facial expressions while minimizing computational overhead, making it suitable for real-time environments.

One of the primary advantages of our method lies in its ability to achieve realistic deformations with small data sizes. This is particularly significant in real-time applications, where memory and bandwidth constraints often limit the complexity of animations. Additionally, the framework's capacity to maintain edge continuity ensures that the resulting animations appear smooth and free from visual artefacts, an essential requirement for creating convincing facial expressions in digital characters.

Our work builds on the foundational goals of this research by providing a solution that balances realism and computational efficiency. Unlike traditional approaches such as Linear Blend Skinning (LBS), which often suffer from artefacts like volume loss, our physics-based method maintains the integrity of the facial model during deformation. Moreover, the incorporation of Newton's second law adds a layer of physical realism, enabling animations that respond naturally to external forces and dynamics.

In conclusion, the proposed physics-based skin deformation method represents a significant step forward in achieving realistic and efficient facial animations. Its ability to handle dynamic expressions, maintain physical accuracy, and operate effectively under real-time constraints makes it a valuable contribution to the field of digital animation.

Chapter 6

Advanced Motion Prediction Method from Monocular Videos

6.1 Introduction

It has been mentioned in Chapter 2 that, as the most important subfield of the digital character motion research field, 3D human pose estimation (HPE) has become a fundamental topic in computer vision research and remains a hot issue. It has broad applications in areas such as action recognition, human-computer interaction, and pose tracking. The goal of HPE is to describe the human body's shape in images, videos, and webcam video streams. This involves multiple tasks, including object recognition, segmentation, regression, and detection. For fulfilling the target of this research in creating realistic digital character animation in real-time environments, we will dive into the digital human motion construction procedure of the whole process in this chapter.

Traditional 3D HPE has gradually become less of a research focus with the recent advent and proliferation of deep learning technologies. Deep learning-based 3D HPE approaches can be broadly categorized into direct regression-based methods and 2D information-based methods. Direct regression-based methods often utilize a large neural network to process all the data and predict 3D human pose information directly from a single image. This allows for end-to-end training and end-to-end output in practical applications. However, the direct regression-based approach requires handling vast amounts of data, demanding high capabilities in network structure and data preprocessing. Besides, they face challenges such as the mismatch between the amount of labelled data and the network's scale.

In contrast, 2D information-based methods assist 3D HPE by incorporating 2D information, such as 2D keypoints and 2D heatmaps. This reduces the hardware burden and training complexity. These methods can be further divided into two categories based on whether the 2D poses and the 3D poses are jointly trained. Approaches that train 2D and 3D pose networks simultaneously use the 2D information obtained within the network as feature values for training. However, tiny errors in 2D joints can be magnified in 3D space, leading to inaccurate 3D HPE results. Another approach is to pre-train a 2D pose network to obtain 2D skeleton sequences, which are then used as additional inputs for training the 3D pose network. Since 2D HPE methods are relatively mature, this approach significantly reduces the overall complexity, making it easier for the network to learn the mapping from 2D to 3D. Moreover, it allows for semi-supervised training through re-projection. Our research also builds on this pre-training method. However, there is inherent ambiguity in the mapping between 3D and 2D, which is a significant challenge that needs to be addressed.

Existing works have proven that the ambiguities can be mitigated by integrating 3D motion priors into the network. These approaches can be roughly classified as explicit and implicit methods depending on the representation of the priors imposed. The explicit methods include observed data optimizing (Akhter and Black 2015) and biomechanics-based joint angle restricting (Hatze 1997, Kodek and Munih 2002). However, it is difficult to completely configure the joint angle limitations related to poses for the entire body. The implicit methods focus on implicit priors, such as denoising score matching (DSM) (Ci et al. 2023), generative adversarial networks (GAN) (Davydov et al. 2022, Peng et al. 2021b), variational auto-encoder (VAE) (Pavlakos et al. 2019, Ling et al. 2020), and Gaussian mixture model (Hou et al. 2021, Bogo et al. 2016). Nevertheless, they face challenges, including training trouble and collapse of posterior mode.

In light of the training framework of the contrastive language-image pre-training (CLIP) model (Radford et al. 2021), we can approach the alignment of cross-modal latent manifolds between video data and 3D skeletal motion data to recreate human motion from monocular videos. The constructed motion priors can be then leveraged to simplify the

learning of meaningful relationships and mappings between these two distinct modalities. This enhances the details and semantic information of the shared latent manifold, of which advantages have been demonstrated in various cross-modal tasks (Ao et al. 2023, Komarichev et al. 2022, Theodoridis et al. 2020). By utilizing recently collected motion datasets, Human3.6m (Ionescu et al. 2013) and AIST++ (Li et al. 2021b), we can construct a compact, well-defined shared latent manifold that encompasses videos of human performances and their corresponding 3D motion data. Since sufficient information for accurate human pose reconstruction is contained in this manually constructed latent manifold, faithful motion can be reconstructed by aligning video data to 3D motion priors.

To achieve the above process, we design a notable Video-to-Motion (VTM) framework, which includes a two-part motion auto-encoder (MoAE) and a two-part visual encoder (TPVE). In our framework, MoAE is first utilized to identify the latent manifold as the motion prior. It separates the learning of the motion latent manifold into upper body and lower body components, effectively reducing the complexity of modelling the entire human pose manifold. Furthermore, MoAE is trained on normalized 3D skeletal motion data, which helps eliminate the impact of skeletal scale on the manifold. Then, a manifold alignment loss is employed to align the video feature manifold with the motion prior. The video feature manifold is constructed from video inputs by TPVE, which is co-trained with MoAE to reconstruct 3D human motion, and also consists of two separate parts: one for the upper body and one for the lower body. This carefully designed process enables motion and visual representations to remain in harmony within the VTM framework.

Our VTM framework has undergone extensive evaluation on the AIST++ dataset (Li et al. 2021b). The reconstruction results indicate that VTM outperforms or matches state-of-the-art (SOTA) methods in terms of various metrics for HPE. It is pointed out that, VTM demonstrates its capability to rapidly generate accurate human motions synchronized with video sequences ($\sim$ 145 fps). This real-time performance significantly benefits the target of this research in creating realistic digital character animation in real-time environments.

The main contribution of this chapter is as follows:

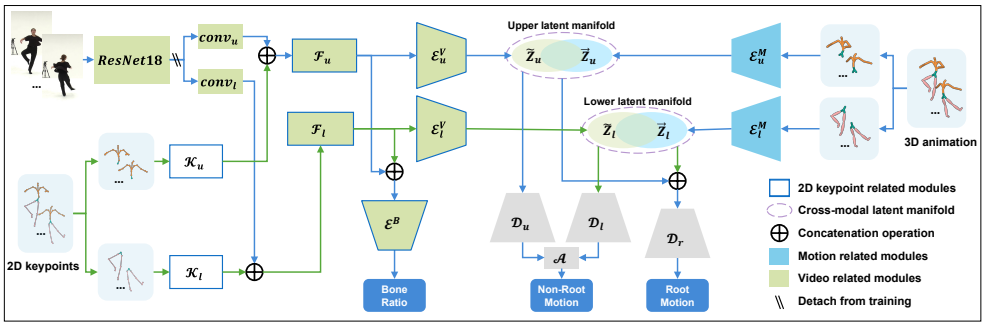


Figure 6.1: The overview of our VTM framework.

- We introduce a novel cross-modal method that effectively leverages 3D motion priors to reconstruct 3D human motion from monocular videos due to the alignment between the video data and motion data on the two-body-part latent feature manifolds. Compared to SOTA methods, our improvements on MRPE and MBLE are 70.2% and 92.9%, respectively.

- We propose a new strategy to learn scale-invariant motion priors from normalized motion data, which aids in achieving more accurate motion reconstruction.

- We conduct thorough experiments to validate the effectiveness of the learned 3D motion priors and explore the best ways to integrate these priors.

6.2 Process of VTM

Fig. 6.1 shows the overview of our proposed Video-to-Motion framework. MoAE denotes a motion auto-encoder, comprising motion encoders $\mathcal{E}_u^M$ and $\mathcal{E}_l^M$, motion decoders $\mathcal{D}_u$ and $\mathcal{D}_l$, and a root decoder $\mathcal{D}_r$. It is first trained on the motion data to learn latent manifolds as the motion priors. TPVE presents a two-part motion encoder, comprising 2D keypoints feature extractors $\mathcal{K}_u$ and $\mathcal{K}_l$, visual fusion blocks $\mathcal{F}_u$ and $\mathcal{F}_l$, and visual encoders $\mathcal{E}_u^V$ and $\mathcal{E}_l^V$. It is then co-trained with MoAE to align the visual features with the motion priors for reconstructing 3D human motion. $\mathcal{E}^B$ is a bone ratio predictor, which is used for predicting the bone ratios. The superscripts M and V stand for *Motion* and *Video*; u and l represent *upper body part*, *lower body part* and *root*. In this section, the process of our VTM will be comprehensively introduced.

6.2.1 Dataset Preparation

6.2.1.1 Dataset Selection

Since 3D human pose reconstruction from videos is a cross-modal issue, the dataset we require for this research should also be capable of providing cross-modal information. The mentioned datasets, AIST++ and Human3.6m, are both cross-modal datasets, containing skeletal motion data and video data, which can train our model to create complete motion representations from monocular videos. However, the AIST++ dataset features a greater diversity of performers compared with the Human3.6m dataset. More performers align well with our goal of a general method in motion reconstruction for different skeletons, whilst they bring more challenges. Thus, the AIST++ dataset is selected as our primary dataset for training and evaluation. Specifically, we utilize the 2D keypoints and videos of camera 1 in the AIST++ dataset as our monocular input.

6.2.1.2 Data Preprocess

We first examine the AIST++ dataset and exclude sequences with incorrect poses to ensure the learned latent manifold is well-defined. Then a skeleton with J joints is constructed for all 27 performers in the AIST++ dataset by using the 3D joint positions, of which the bone lengths are adjusted to match the characters in the videos. Using the joint rotations and the created skeletons to generate motion and store them in BVH format, the BVHAIST++ dataset is created.

Subsequently, we create a universal skeleton $\bar{\mathbf{s}}$, which averages all 27 skeletons in the BVHAIST++ dataset. All the motion data can be aligned with it to create a normalized dataset, where an extra parameter, bone ratios $\mathbf{b}$, is introduced to represent ratios of each bone between the universal skeleton $\bar{\mathbf{s}}$ and the original skeleton. The ratios predicted by our VTM can be used to restore the original skeleton from $\bar{s}$ during inference, preserving fidelity to the original skeletal structures.

6.2.1.3 3D Motion Representation

Utilizing the camera parameters provided by the datasets as the transfer matrix, the BVH files of the average skeleton $\bar{\mathbf{s}}$ can be converted from the world space, i.e. vertex coordinates, to the camera space. Hence the t^{th} frame of the transformed motion sequence can be presented as

$\mathbf{x}_t = \{\mathbf{r}_t^q, \mathbf{r}_t^p, \mathbf{r}_t^v, \mathbf{x}_t^q, \mathbf{x}_t^p, \mathbf{x}_t^v\}$, where $\mathbf{r}_t^q \in \mathbb{R}^6$ and $\mathbf{x}_t^q \in \mathbb{R}^{6(J-1)}$ stand for the 6D rotation representations (Zhou et al. 2019b) of the root and non-root joints, $\mathbf{r}_t^p \in \mathbb{R}^3$ and $\mathbf{x}_t^p \in \mathbb{R}^{3(J-1)}$ denote the global 3D joint positions, and $\mathbf{r}_t^v \in \mathbb{R}^3$ and $\mathbf{x}_t^v \in \mathbb{R}^{3(J-1)}$ represent linear velocities.

6.2.1.4 2D Keypoints Representation

Camera parameters are utilized for projecting 3D joint positions, which are computed by forward kinematics (FK), into 2D space to generate 2D keypoints. Thus, the scale consistency between 2D keypoints and 3D motion can be conserved. The t^{th} frame of the 2D keypoints can be presented as $\mathbf{k}_t = \{\mathbf{k}_t^p, \mathbf{k}_t^v\}$, where $\mathbf{k}_t^p \in \mathbb{R}^{2J}$ represents the 2D virtual keypoints, and $\mathbf{k}_t^v \in \mathbb{R}^{2J}$ denotes their linear velocities.

6.2.2 Pre-training Motion Priors

Siyao et al. (2022) has proven the effectiveness of independently reconstructing root and non-root joints in 3D HPE. Thus, a motion autoencoder(MoAE) is developed to distil compact and well-defined motion priors from motion data, which consists of a two-part motion encoder (TPME), a two-part motion decoder (TPMD), and a root decoder (RD), as shown in Fig. 6.1. Similar to (Zhang et al. 2023), our approach integrates a two-part design into a convolutional architecture.

6.2.2.1 TPME

TPME contains two motion encoders $\mathcal{E}_u^M$ and $\mathcal{E}_l^M$, which are designated for the upper and lower body parts, respectively. The inputs $\mathbf{X}_u$ and $\mathbf{X}_l$ are split from the input sequence $\mathbf{X} = \{\mathbf{x}_t, \ldots, \mathbf{x}_{t+T-1}\} \in \mathbb{R}^{T\times J\times 12}$ and contain root data. The encoders process the inputs to the latent vectors $\mathbf{Z}_u \in \mathbb{R}^{\frac{T}{4}\times 128}$ and $\mathbf{Z}_l \in \mathbb{R}^{\frac{T}{4}\times 64}$ as follows:

$$\mathbf{Z}_u = \mathcal{E}_u^M(\mathbf{X}_u), \qquad \mathbf{Z}_l = \mathcal{E}_l^M(\mathbf{X}_l) \tag{6.1}$$

According to the alignment between the universal skeleton $\bar{\mathbf{s}}$ and the input $\mathbf{X}$, the two latent vectors are normalized for the capacity of capturing 3D motion kinematic constraints without the dependency on the skeleton scale. Thus, these two vectors can serve as motion priors, which benefit for gaining a decreased Mean Per Joint Position Error (MPJPE) value for the reconstructed motion.

6.2.2.2 TPMD

TPMD leverages decoders $\mathcal{D}_u$, $\mathcal{D}_l$ and a 1D convolutional aggregation layer $\mathcal{A}$ to decode non-root motion $\hat{\mathbf{X}}_{nr}$ from the latent vectors $\mathbf{Z}_u$ and $\mathbf{Z}_l$ as:

$$\hat{\mathbf{X}}_{nr} = \mathcal{A}(\mathcal{D}_u(\mathbf{Z}_u) \oplus \mathcal{D}_l(\mathbf{Z}_l)) \tag{6.2}$$

where $\oplus$ denotes the concatenation operation.

6.2.2.3 RD

By comparison, RD only leverage decoder $\mathcal{D}_r$ to directly decode root motion $\hat{\mathbf{X}}_r$ from the concatenation of $\mathbf{Z}_u$ and $\mathbf{Z}_l$ as:

$$\hat{\mathbf{X}}_r = \mathcal{D}_r(\mathbf{Z}_u \oplus \mathbf{Z}_l) \tag{6.3}$$

where $\hat{\mathbf{X}}_r$ only include the velocities rz^v and 3D root positions rz^p on the Z-axis, and the root rotation $\mathbf{r}^q$, which can be computed by rz^p, 2D keypoints and the camera intrinsic parameters during inference.

6.2.3 Learn Human Motion from Videos

For learning 3D human motion from videos, we present a two-part visual encoder (TPVE), which achieves the mapping between motion priors and visual features, to leverage the learned motion priors. Thus, the joint rotations can be decoded using TPMD and RD. Besides, a bone ratio predictor (BRP) and a root translation predictor (RTP) are designed for directly reconstructing root translations and bone ratios from videos, respectively.

6.2.3.1 TPVE

As shown in Fig. 6.1, TPVE consists of several components: a video feature extractor, two 2D keypoint feature extractors for the upper and lower body, two visual feature fusion blocks that combine video features with keypoint features, and two visual encoders that map the fused visual features to the latent manifold shared by the motion data.

The video feature extractor comprises two learnable 1D convolutional layers cov_u and cov_l, which allow TPVE to adapt and refine video features, and a pre-trained ResNet18 (He et al. 2016). We fix the weights of ResNet18 and remove its last fully connected layer.

The two 2D keypoint feature extractors $\mathcal{K}_u$ and $\mathcal{K}_l$ are three-layer CNNs. 2D keypoint features can be extracted by utilizing these two extractors to process sub-sequences $\mathbf{K}_u$ and $\mathbf{K}_l$, which are separated from 2D keypoint sequences $\mathbf{K} \in \mathbb{R}^{T \times J \times 4}$.

The visual feature fusion blocks leverage two 1D convolutional layers as two residual blocks $\mathcal{F}_u$ and $\mathcal{F}_l$ to fuse video and keypoint features $\tilde{\mathbf{V}}_u$ and $\tilde{\mathbf{V}}_l$ as:

$$\begin{aligned} \tilde{\mathbf{V}}_u &= \mathcal{F}_u(conv_u(\mathbf{V}) \oplus \mathcal{K}(\mathbf{K}_u)), \\ \tilde{\mathbf{V}}_l &= \mathcal{F}_l(conv_l(\mathbf{V}) \oplus \mathcal{K}(\mathbf{K}_l)) \end{aligned} \tag{6.4}$$

The visual encoders consist of $\mathcal{E}_u^V$ and $\mathcal{E}_l^V$ and are designed for mapping between latent manifold and visual features. For better capturing temporal correlations, they are improved by two cross-temporal context aggregation (CTCA) modules (Guo et al. 2023). The latent manifolds $\tilde{\mathbf{Z}}_u \in \mathbb{R}^{\frac{T}{4} \times 128}$ and $\tilde{\mathbf{Z}}_l \in \mathbb{R}^{\frac{T}{4} \times 64}$ are encoded from the visual features $\tilde{\mathbf{V}}_u$ and $\tilde{\mathbf{V}}_l$ as:

$$\begin{aligned} \tilde{\mathbf{Z}}_u &= \mathcal{E}_u^V(\tilde{\mathbf{V}}_u), \\ \tilde{\mathbf{Z}}_l &= \mathcal{E}_l^V(\tilde{\mathbf{V}}_l) \end{aligned} \tag{6.5}$$

6.2.3.2 BRP

Due to the significant difference in human height and proportions among individuals, the scale of skeletons is crucial in 3D human motion. To address this, we created the universal skeleton $\bar{\mathbf{s}}$ to standardize the skeleton during training. As video and 2D keypoints inherently reflect the scale information of characters, we utilize the Bone Ratio Predictor (BRP) $\mathcal{E}^B$ to predict bone ratios $\tilde{\mathbf{b}}$ from the visual features $\tilde{\mathbf{V}}_u$ and $\tilde{\mathbf{V}}_l$ as:

$$\tilde{\mathbf{b}} = \mathcal{E}^B(\tilde{\mathbf{V}}_u \oplus \tilde{\mathbf{V}}_l) \tag{6.6}$$

The predicted bone ratios $\tilde{\mathbf{b}}$ can be used to reconstruct individual skeletons $\tilde{\mathbf{b}} \cdot \bar{\mathbf{s}}$ during training.

6.2.3.3 RTP

Global root translation is critical for accurately reconstructing 3D human motion. Nevertheless, it is challenging to gain precise root translation due to the ambiguity caused by global 3D joint positions. For example, the same 2D pose can be performed by the same person at different 3D locations. To address this issue, the Root Translation Predictor (RTP) module is introduced, which directly reconstructs these translations from the video.

6.2.4 Training Loss

6.2.4.1 Loss for Motion Priors

The loss L^M that we use for training MoAE to learn motion priors consists of a motion reconstruction loss L^M_{rec} and a motion smoothness loss L^M_s:

$$L^M = L^M_{rec} + L^M_s \tag{6.7}$$

L^M_{rec} is a smooth L1 loss that considers the relative importance of distinct joints to ensure the latent manifold is well-defined (Holden et al. 2017), formulated as:

$$L^M_{rec} = \boldsymbol{\omega}_r L_1(\hat{\mathbf{X}}_r, \mathbf{X}_r) + \boldsymbol{\omega}_{nr} L_1(\hat{\mathbf{X}}_{nr}, \mathbf{X}_{nr}) \tag{6.8}$$

where $\boldsymbol{\omega}_r$ and $\boldsymbol{\omega}_{nr}$ are relative weights.

L^M_s helps to smooth out sudden changes in the reconstructed motion over time. It is formulated as:

$$\begin{aligned} L^M_s =& \boldsymbol{\omega}_r L_1(\hat{\mathbf{Vel}}_r, \mathbf{Vel}_r) + L_1(\hat{\mathbf{Vel}}_{nr}, \mathbf{Vel}_{nr}) + \\ & \boldsymbol{\omega}_r L_1(\hat{\mathbf{Acc}}_r, \mathbf{Acc}_r) + L_1(\hat{\mathbf{Acc}}_{nr}, \mathbf{Acc}_{nr}) \end{aligned} \tag{6.9}$$

where $\mathbf{Vel}$ & $\hat{\mathbf{Vel}}$ and $\mathbf{Acc}$ & $\hat{\mathbf{Acc}}$ denotes the velocities and accelerations of the ground-truth and reconstructed data, respectively.

6.2.4.2 Loss for Motion Reconstruction

The loss L that we use for co-training TPVE and BRP with MoAE consists of a manifold alignment loss L_{ma}, a bone prediction loss L_b, a motion prediction loss L^V_{pred} and a smoothness loss L^V_s, formulated as:

$$L = L_{ma} + L_b + L^V_{pred} + L^V_s + \lambda L^M \tag{6.10}$$

L_{ma} aligns visual latent manifold with motion priors, formulated as:

$$L_{ma} = L_1(\tilde{\mathbf{Z}}_u, \mathbf{Z}_u) + L_1(\tilde{\mathbf{Z}}_l, \mathbf{Z}_l) \tag{6.11}$$

L_b corrects the bone ratio prediction by BRP, formulated as:

$$L_b = L_1(\tilde{\mathbf{b}}, \mathbf{b}) \tag{6.12}$$

L^V_{pred} optimizes the motion prediction, formulated as:

$$L^V_{pred} = \boldsymbol{\omega}_r L_1(\tilde{\mathbf{X}}_r, \mathbf{X}_r) + \boldsymbol{\omega}_{nr} L_1(\tilde{\mathbf{X}}_{nr}, \mathbf{X}_{nr}) \tag{6.13}$$

where $\tilde{\mathbf{X}}_r$ and $\tilde{\mathbf{X}}_{nr}$ denote the root and non-root data, generated by decoders using visual inputs of TPVE.

$$\begin{aligned} L_s^V =& \boldsymbol{\omega}_r L_1(\tilde{\mathbf{Vel}}_r, \mathbf{Vel}_r) + L_1(\tilde{\mathbf{Vel}}_{nr}, \mathbf{Vel}_{nr}) + \\ & \boldsymbol{\omega}_r L_1(\tilde{\mathbf{Acc}}_r, \mathbf{Acc}_r) + L_1(\tilde{\mathbf{Acc}}_{nr}, \mathbf{Acc}_{nr}) \end{aligned} \tag{6.14}$$

where $\tilde{\mathbf{Vel}}$ and $\tilde{\mathbf{Acc}}$ denote the velocities and accelerations of the predicted motion.

6.2.4.3 Root Translation Reconstruction

The loss L^T that we use for RTP to reconstruct root translation consists of a motion prediction loss L_{pred}^{VT}, and a smoothness loss L_s^{VT}, a motion prediction loss L_{pred}^{V} and a smoothness loss L_s^{V}, formulated as:

$$L^T = L_{pred}^V + L_s^V \tag{6.15}$$

$$L_{pred}^{VT} = \boldsymbol{\omega}_r L_1(\tilde{\mathbf{X}}'_r, \mathbf{X}'_r) + \boldsymbol{\omega}_{nr} L_1(\tilde{\mathbf{X}}'_{nr}, \mathbf{X}'_{nr}) \tag{6.16}$$

where $\tilde{\mathbf{X}}'_r$ and $\tilde{\mathbf{X}}'_{nr}$ denote the root and non-root data generated by RTP.

$$\begin{aligned} L_s^{VT} =& \boldsymbol{\omega}_r L_1(\tilde{\mathbf{Vel}}'_r, \mathbf{Vel}'_r) + L_1(\tilde{\mathbf{Vel}}'_{nr}, \mathbf{Vel}'_{nr}) + \\ & \boldsymbol{\omega}_r L_1(\tilde{\mathbf{Acc}}'_r, \mathbf{Acc}'_r) + L_1(\tilde{\mathbf{Acc}}'_{nr}, \mathbf{Acc}'_{nr}) \end{aligned} \tag{6.17}$$

We set the relative weights from Eq. 6.7 to Eq. 6.17 as: the relative importance for the root is 2.0, the end-effectors is 1.5, and all other joints are 1.0.

Motion encoders are ignored during training, with only 2D inputs fed into our VTM for human motion reconstruction during inference. For the rest of the digital characters, motion reconstruction can be dealt with in a similar way, with different skeletons and motion priors. The final generated motion consists of rotations of all joints, which are generated by decoders with visual inputs, and global 3D positions, which RTP generates.

6.3 Experiments

6.3.1 Implementation Details

The experiments for our VTM are implemented by PyTorch 1.10.1 on a desktop PC, which is equipped with an Intel Xeon(R) E5-2678 CPU, 128GB RAM, and two GeForce RTX 3090 24GB graphics cards.

The setup configuration is:

1. Set joint number J of the universal skeleton $\bar{\mathbf{s}}$ to 24.
2. Extract 32 frames from each of the video sequences by a sliding window of length 4 for training.
3. Train MoAE for 500 epochs by the AdamW optimizer (Loshchilov and Hutter 2017) with a batch size of 100 and a learning rate of initially $1e-4$, decayed by 0.5 every 100 epochs.
4. Co-train TPVE with MoAE by the same settings but a batch size of 64.
5. Train RTP independently by the same settings of TPVE.

6.3.2 Comparisons

As far as we surveyed, MotioNet (Shi et al. 2020) is the only existing method that, from monocular video inputs, generates a thorough 3D human representation, including a kinematic skeleton with scale information, root joint translation, and joint rotations. For a comprehensive evaluation, VTM is also compared with other SOTA methods that predict 3D joint positions or SMPL parameters by different model structures and inputs. To ensure a fair comparison, the BVHAIST++ dataset and the same skeletal topology are applied for training these models via their released code and training configurations. Specifically, we choose the VPose backbone (Pavllo et al. 2019) for the Poseaug (Gong et al. 2021) model owing to its superior performance. For the RLE (Li et al. 2021a) model, we add a depth channel to extend their codes to implementation on 3D HPE, as their experiments are all in 2D space. Since no detailed 3D pose representation is provided in PCT (Geng et al. 2023), we only train the first-stage motion tokens by leveraging the 3D joint positions in camera space.

6.3.2.1 Quantitative Comparisons

We utilize two standard metrics to evaluate the performance of HPE, MPJPE and PA-MPJPE. Mean Per Joint Position Error(MPJPE) measures the average distance between the predicted joints and the ground truth joints of a human skeleton, formulated as:

$$\mathbf{MPJPE} = \frac{1}{24T}\sum_{n=0}^{T-1}\sum_{j=0}^{23}\left\|\hat{\mathbf{p}}_n^j - \mathbf{p}_n^j\right\|_2 \tag{6.18}$$

where $\hat{\mathbf{p}}_n^j$ and $\mathbf{p}_n^j$ denote the generated and ground-truth positions of the j^{th} joints at the n^{th} frame, respectively. Procrustes Aligned Mean Per Joint Position Error (PA-MPJPE) is a variant of MPJPE, which measures the average Euclidean distance between the predicted and ground-truth joint positions. The lower value of MPJPE and PA-MPJPE stands for better performance of the algorithm.

Specifically, we compute Mean Bone Length Error (MBLE) for different approaches to evaluate the level of the reconstruction of the complete motion representation. Its formulation is:

$$\mathbf{MBLE} = \frac{1}{23T} \sum_{n=0}^{T-1} \sum_{j=0}^{22} \left\| \hat{\mathbf{bl}}_n^j - \mathbf{bl}_n^j \right\|_2 \tag{6.19}$$

where $\hat{\mathbf{r}}_n^p$ and $\mathbf{r}_n^p$ denote the generated and ground-truth root positions at the n^{th} frame, respectively.

Moreover, Mean Root Position Error(MRPE) is employed to evaluate the root translation reconstruction when compared with MotioNet and Ray3D (Zhan et al. 2022). Its formulation is:

$$\mathbf{MRPE} = \frac{1}{T} \sum_{n=0}^{T-1} \left\| \hat{\mathbf{r}}_n^p - \mathbf{r}_n^p \right\|_2 \tag{6.20}$$

where $\hat{\mathbf{bl}}_n^j$ and $\mathbf{bl}_n^j$ denote the length of the generated and ground-truth j^{th} bone at the n^{th} frame, respectively. All metrics are computed on the validation dataset and measured in millimetres.

The joint rotations generated by VTM and MotioNet utilize FK to compute 3D joint positions for metric computation. As shown in Table 6.1, VTM ranks second and third in MPJPE and PA-MPJPE, respectively. This variation is due to the unique objective of VTM, which prioritizes reconstructing comprehensive 3D motion representations from monocular video, differentiating it from methods mainly focused on 3D joint positions. Notably, VTM significantly outperforms MotioNet and Ray3D in MRPE. Additionally, VTM and MotioNet hold the top two positions in MBLE. This highlights the effectiveness of reconstructing complete motion representations in addressing the issue of bone length inconsistencies in 3D human motion reconstruction, which is vital for applications requiring skinning.

Except for the experiments implemented on AIST++, we also evaluate the performance of VTM from scratch on the Human3.6m and 3DPW datasets.

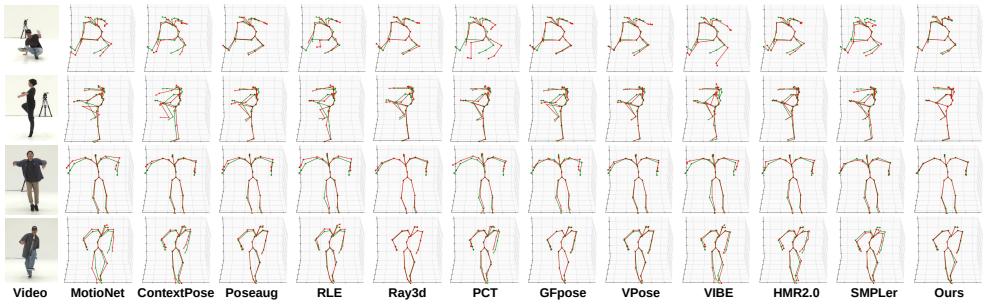


Figure 6.2: The qualitative comparisons on motion reconstruction of our VTM to SOTA methods.

MPJPE, PA-MPJPE, and MRPE on the Human3.6m dataset are 66.0, 50.2, and 15.6, respectively. MRPE of VTM is less than the reported value of 109.5 from Ray3D as the skeleton of the Human3.6m dataset contains three extra joints, which correspond to zero-length bones. These additional joints are crucial for human motion driven by skeletons and rotations. Nevertheless, these extra joints increase the difficulty in optimizing networks, resulting in MPJPE of VTM not matching SOTA methods.

Since the 3DPW dataset has jittery roots, we only evaluate MPJPE and PA-MPJPE on it. Their values on the validation datasets of BVHAIST++ and 3DPW are $(20.9, 18.1)$ and $(108.0, 79.0)$, respectively, which are comparable to other implementations on 3DPW of SOTA methods. This demonstrates that VTM is robust for implementation on in-the-wild datasets.

6.3.2.2 Qualitative Comparisons

We select four random video frames from the validation dataset to visually assess the motion quality reconstructed by different approaches. The construction results are shown in Fig. 6.2, where the ground truth poses are represented as green skeletons, while the reconstructed poses are represented as red skeletons. It can be observed that the constructed motions generated by VTM match the ground-truth motions much better than most SOTA methods, especially in scenarios with significant self-occlusions like the motions on the second row. This distinct advantage is attributed to the introduced motion priors, which encode plausible 3D motion manifolds. The correlations between joint rotations can be captured by this encoding to facilitate the inference of reasonable human poses in these challenging situations.

Table 6.1: The quantitative comparisons on metrics of our VTM to SOTA methods.

Method	MotioNet (Shi et al. 2020)	ContextPose (Ma et al. 2021)	Poseaug (Gong et al. 2021)	RLE (Li et al. 2021a)	Ray3d (Zhan et al. 2022)	PCT (Geng et al. 2023)
MPJPE↓	37.0	20.1	24.8	45.1	13.2	23.7
PA-MPJPE↓	33.4	14.8	19.6	37.4	10.8	17.8
MRPE↓	45.3	-	-	-	141.4	-
MBLE↓	1.4	40.4	4.8	9.5	3.5	5.4
Method	GFpose (Ci et al. 2023)	VPose (Pavllo et al. 2019)	VIBE (Kocabas et al. 2020)	HMR2.0 (Goel et al. 2023)	SMPLer (Xu et al. 2023)	ours
MPJPE↓	27.3	20.6	59.1	32.2	40.8	17.5
PA-MPJPE↓	19.2	17.0	47.7	26.4	25.8	15.3
MRPE↓	-	-	-	-	-	13.5
MBLE↓	6.0	3.8	7.2	13.6	9.0	0.1

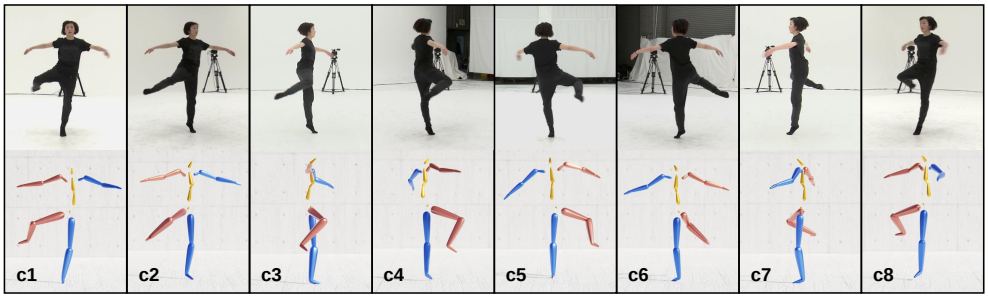


Figure 6.3: The reconstruction results of the same performer at the same frame from different unseen view angles.

6.3.3 Evaluations

6.3.3.1 Robustness to Unseen View Angles

As described in Section 6.2.1, our VTM was specifically trained on 2D keypoints and videos from camera 1 in the AIST++ dataset. However, the performance of VTM in employing 2D keypoints and videos extracted from cameras 2 to 8 is shown in Fig. 6.3. The first row shows the same performer in the same frame with distinct unused camera settings, while the second row shows their corresponding reconstructed motions. This experiment demonstrates the robustness of VTM across previously unseen view angles during training.

6.3.3.2 Robustness to in-the-wild videos

A network for mapping is introduced to allow VTM with motion reconstruction from in-the-wild videos, of which the structure is a four-layer MLP. With this network, the COCO-formatted keypoints can be converted into our virtual 2D keypoints. It advantages our VTM to implement on in-the-wild videos, resulting in direct use of 2D keypoints detected by readily available 2D HPE models. Two examples of reconstructing motion from arbitrary videos are shown in Fig. 6.4 and Fig. 6.5, where the first row shows randomly grabbed frames of a continuous video sequence and the second row shows their corresponding reconstructed motions. This experiment illustrates the robustness of VTM in consistently producing accurate 3D human motion from in-the-wild videos.

Figure 6.4: The reconstruction results from in-the-wild videos.

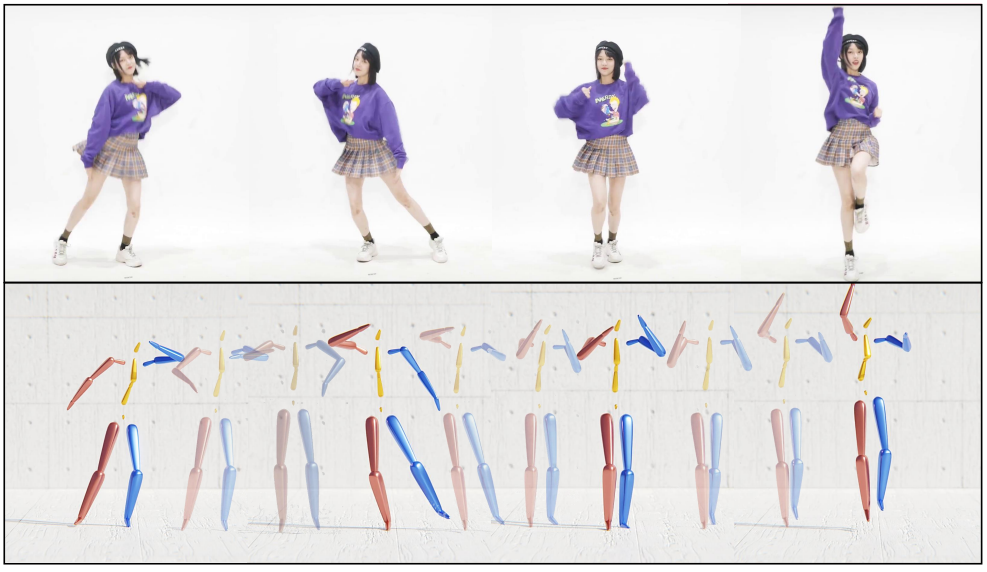


Figure 6.5: Another reconstruction results from in-the-wild videos.

6.3.3.3 Evaluations on MoAE

The effectiveness of MoAE is validated through three experiments with distinct configurations:

1. MoAE-SKW: We use the unnormalized skeletons to train MoAE instead of the normalized universal skeleton $\bar{\mathbf{s}}$ for investigating the impact of different skeleton scales on reconstruction accuracy.

2. MoAE-Q: We follow the methods in (Holden et al. 2017, Hou et al. 2021) to leverage quaternion-based rotation and a local 3D joint position representation for evaluating the superiority of our proposed 6D rotation representation.

3. OPMoAE: We do not process the skeleton with the upper and lower parts but increase the latent space dimension to 196 to conserve its size for evaluating the necessity of the two-part design.

The quantitative evaluation results are shown in Table 6.2. It can be observed that MPJPE on MoAE-Q is much larger than MPJPE. It is due to the error accumulation caused by representing root rotations as angular velocities to the Y-axis, which is more severe for long sequences. The evaluation comparison results demonstrate the efficacy of our three designs in MoAE, including the normalized skeleton $\bar{\mathbf{s}}$, the 6D rotation representation, and the two-part design.

Table 6.2: The evaluation comparisons of MoAE and its variants on MPJPE, PA-MPJPE, and MRPE.

Method	MoAE-SKW	MoAE-Q	OPMoAE	MoAE
MPJPE↓	4.9	84.0	5.7	4.8
PA-MPJPE↓	4.1	10.7	4.9	4.0
MRPE↓	0.8	98.5	1.1	0.8

6.3.3.4 Evaluations on Motion Priors

VTM co-trains the motion priors with MoAE by creating the alignment between the visual latent manifold and the pre-trained motion priors. The following experiments are implemented to assess its efficacy, where

the quantitative results on MPJPE, PA-MPJPE, and MRPE are shown in Table 6.3.

Joint training. We set different values of λ for VTM training to evaluate the co-training with MoAE. As shown in Table 6.3, when $\lambda > 0$, an inverse trend can be found on all three metrics, where we achieve the best MPJPE but the worst MRPE when $\lambda = 0.3$. All metrics are poor when $\lambda = 0.0$, which is fixed for the rest models.

`VTM-w/o_prior` stands for the model where MoAE is deleted and L_{ma} is discarded from Eq. 6.10 to remove the motion priors. Its result in Table 6.3 illustrates that a more precise root translation reconstruction result can be achieved without motion priors. The negative impact of motion priors on root translation reconstruction accuracy is caused by the reason explained in Section 6.2.3.3. Therefore, `VTM-`λ0.3 is employed to produce joint rotations, and `VTM-w/o_prior` is treated as the RTP module to produce root translations. Since motion priors enhance the bone ratio prediction, we can find that `VTM-w/o_prior`'s MRPE 13.5 is worse than VTM's MRPE 13.6 with better-predicted bone ratios.

Since different models are used to generate joint rotations and root rotations, we discard the unused outputs from `VTM-`λ0.3 and `VTM-w/o_prior`. Consequently, we implement two models `VTM-w/o_RT`, which configures `VTM-`λ0.3 to generate only joint rotations, and `VTM-w/o_prior-OR`, which configures `VTM-w/o_prior` to generate only root translations. However, their worse results in Table 6.3 illustrate that it is mutually beneficial to treat joint rotations and root translations as auxiliary outputs for each other.

`VTM-w/PT-D` jointly trains both TPVE and decoders, which are initialized with the pre-trained MoAE, to generate motion priors, whilst `VTM-w/PT-fixD` only trains TPVE and leverages the pre-trained MoAE decoders with fixed weights to generate motion priors. The result comparison between Table 6.3 and `VTM-w/o_prior` in Table 6.3 indicate that joint training for motion priors assist in precisely reconstructing joint rotations. However, the results of `VTM-w/o_PT-fixD` illustrate the decreased performance when imposing improper motion priors.

Latent manifolds alignment. We experiment with three models to figure out a suitable method for aligning cross-modal latent manifolds. `VTM-w/o_ma` discards L_{ma} from Eq. 6.10, `VTM-CL` follow the methodology used in CLIP to replace L_{ma} with a contrastive loss(CL), and `VTM-CL+L1`

integrates CL into L_{ma}. Nevertheless, the metric results of them are worse than VTM. This indicates that CL is inappropriate for our dataset as it contains extensive similar poses.

Scale-independent motion priors. To explore the influence of skeleton scales on 3D motion priors, we conduct `VTM-SKW`, which employs MoAE-SKW to train the motion data generated from the original skeletons. The results indicate the significance of our universal skeleton $\bar{\mathbf{s}}$ as VTM only learns kinematic constraints as joint rotations when skeleton scale information is eliminated.

6.3.3.5 Evaluations on two-part design

As shown in Fig. 6.1, a module $\mathcal{A}$ is adopted to conserve the correlations between the upper and lower body parts. Thus, for evaluating the efficacy of this design, we conduct `VTM-CA`, which adds two cross-attention modules before decoders, and `VTM-w/o_A`, which discards $\mathcal{A}$ and concatenates the outputs of decoders as the final non-root motion. The worse results illustrate the significance of the aggregation layer.

6.3.3.6 Other Evaluations

`VTM-OP` removes the two-part design and replaces MoAE with OPMoAE, of which the results are much worse than VTM though these two autoencoders have similar performance in Table 6.2. This proves that HPE from monocular videos is much more complex than normal HPE tasks.

`VTM-w/o_CTCA` removes the CTCA modules from the visual encoders, of which the results indicate CTCA's effectiveness in improving the temporal correlations to enhance reconstruction accuracy.

`VTM-w/o_K` only inputs videos for training, of which the results are notably inferior in all three metrics. On the other hand, `VTM-w/o_V` only inputs 2D keypoints for training, of which the results remain top-3 ranking in MPJPE, but distinctly drop in MRPE. This is because 2D keypoints contain motion contours, whilst video data benefits building global spatial awareness.

`VTM-w/o_prior-L` increases the dimensions of $\mathbf{Z}_u$ and $\mathbf{Z}_l$ to 192 and 96, of which the results are comparable to `VTM-w/o_prior` but are worse than VTM.

Table 6.3: The evaluation comparisons of VTM and its variants on MPJPE, PA-MPJPE, and MRPE.

Method	VTM-λ1.0	VTM-λ0.3	VTM-λ0.1	VTM-λ0.05	VTM-λ0.0	VTM-w/o_prior	VTM-w/o_RT	VTM-w/o_prior-OR
MPJPE↓	17.8	17.5	17.8	18.3	23.0	20.8	18.0	-
PA-MPJPE↓	15.7	15.3	15.6	16.1	18.9	18.0	15.9	-
MRPE↓	16.8	17.6	15.0	14.9	23.3	13.6	-	20.9
Method	VTM-PT-D	VTM-PT-fixD	VTM-w/o_ma	VTM-CL	VTM-CL+L1	VTM-SKW	VTM-CA	VTM-w/o_A
MPJPE↓	19.3	21.8	25.6	23.1	21.4	18.9	24.5	42.4
PA-MPJPE↓	16.9	18.1	19.2	19.8	18.9	16.6	22.0	40.3
MRPE↓	15.1	24.3	14.5	20.9	20.1	19.9	19.9	18.3
Method	VTM-OP	VTM-w/o_CTCA	VTM-w/o_K	VTM-w/o_V	VTM-w/o_prior-L	VTM		
MPJPE↓	75.7	17.5	73.2	18.0	20.9	17.5		
PA-MPJPE↓	60.7	15.7	59.3	15.5	18.0	15.3		
MRPE↓	63.5	14.5	65.0	25.2	13.9	13.3		

6.3.4 Skeleton-driven Mesh Animation

Skeleton-only animation often fails to showcase realistic and natural movements for digital humans in real-time environments. To address this, the industry typically uses mesh-based animation to represent a wide range of dynamic motions and deformations, achieving a more natural and lifelike effect. As introduced in (Rodriguez 2013), driving a mesh model involves two interconnected steps: rigging and skinning. Rigging involves adding controllers to simplify the manipulation of complex poses by assigning weights to multiple bones. Skinning binds the mesh vertices to corresponding bones, ensuring that when the skeleton moves, the mesh deforms accordingly. This allows the skeleton's motion to drive the mesh and produce high-quality performance of deformation animations. For most cases, rigging and skinning are predominantly performed manually by designers, which is both time-consuming and labour-intensive. Consequently, significant efforts have been made to develop automatic techniques for rigging and skinning. For example, Baran and Popović (2007) proposed Pinocchio, a system designed for auto-rigging models, which solves the weight-assigning problem and optimal skeleton embedding searching problem. Pan et al. (2009) presented another simple and efficient automatic rigging approach. It constructs the hierarchical skeleton by refining a coarse curve skeleton, which is extracted from the character, with a perpendicular silhouette. However, current commercial software has integrated these advanced algorithms into the program to benefit the designers.

To evaluate the capability of our proposed method to generate real-time animated digital humans driven by predicted skeletal motion, we designed an additional experiment, which is inspired by (Liu et al. 2022c). Since our method predicts skeletal motion directly from video, the only remaining step is to bind the predicted skeleton to a digital human mesh via skinning. We utilized *Blender* as the software platform for this experiment. First, we generated a new male human mesh model using the add-on called *HumanGeneratorV4*, which is general and unique. Then, we employed another add-on, *AutoRigPro*, to automatically rig the skeleton to the mesh model. After rigging, we used Blender's Python API to render the predicted skeletal motion and the mesh animation accordingly in real-time from video. Sample frames of the rendered animation are shown in Fig. 6.6, where the first row is the sample frames of the original

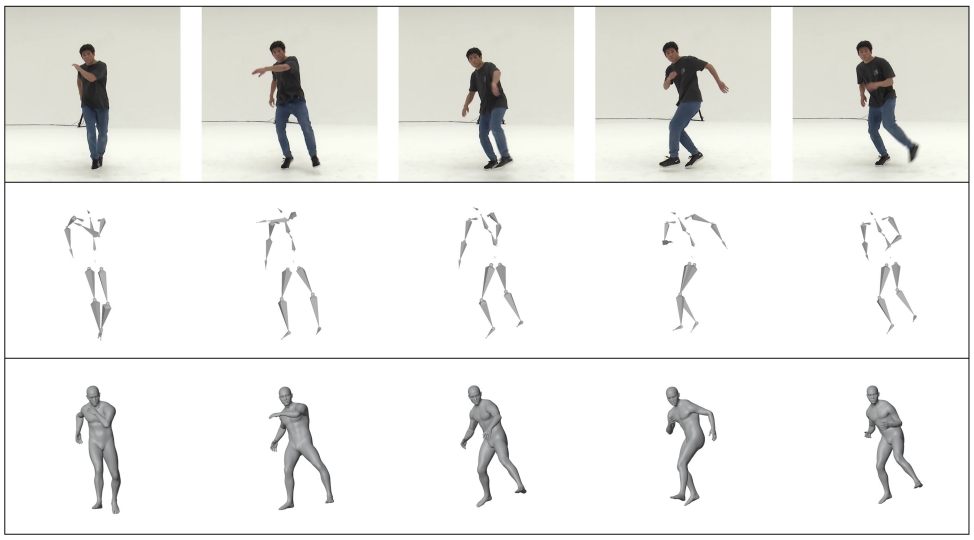


Figure 6.6: Sample frame comparisons among videos, predicted skeletons, and mesh animations.

videos, and the second and the third rows are the same frames of the predicted skeleton and the skinned human models, respectively. Our method predicts motion on the universal skeleton $\bar{\mathbf{s}}$, which means the prediction results are not affected by the topology and shape of the skeletons. After the pre-skinning process is complete, the rendered animation closely matches the frame rate of skeletal motion prediction. Specifically, our system achieves approximately 145 frames per second (FPS), which is also synchronized with video sequences and motion generation, demonstrating that our approach is capable of producing real-time and accurate skeleton-driven digital human animations.

6.4 Conclusion

In this chapter, we propose a novel Video-to-Motion(VTM) framework to reconstruct human motion from monocular videos with a two-part strategy. VTM first uses a two-part motion auto-encoder to learn well-defined motion priors and then leverages them to reconstruct motions from video data. With comprehensive experiments, it has been demonstrated to be effective and robust for accurately generating high-quality motions from unseen view angles and from in-the-wild videos. Besides, it has been proven to achieve realistic mesh animation in real-time environments, which fits the target of this research in creating realistic animation of digital characters in interactive systems.

Though VTM is only implemented on digital humans, other digital characters can deal with motion reconstruction in a similar way with different skeletons and motion priors. However, acquiring high-quality datasets for other digital characters is still a big challenge.

Chapter 7

Conclusion and Future Work

This thesis studied the existing research on 3D modelling, skin deformation, and 3D human pose reconstruction and gained insight into the fundamental techniques in the creation of realistic animations for digital characters, especially digital humans, in real-time interactive environments, such as games and virtual chatting. Then, a series of advanced methods toward answering the research questions (Section 1.2) are presented.

In chapter 3, the latest developments and applications of approaches based on PBD are reviewed. It begins with the introduction to the core algorithm of the basic PBD for a particle-based system, including the three primary parts: damping, collision, and the solver. Then, the improvements in position-based approaches since 2018 are thoroughly studied, including its extensions addressing PBD's inherent limitations, such as PD and XPBD, and integration with other systems, such as cloth, rigid body, and fluids. The recent applications of PBD in various fields since 2018 are also summarized, including medical systems, deep learning, and the architectural industry. These works demonstrate the efficacy of position-based approaches in real-time systems. Upon these works, the PBD techniques are utilized for the later chapter to achieve the goal of our research.

In Chapter 4, a PDE-based modelling method is introduced for fast and accurate model reconstruction in dynamic deformation simulation. Its mathematical model is derived from Newton's second law and the governing equation of elastic beams, and gains the close-form solutions leveraging the separation of variables technique. The time variable is integrated into the mathematical model by normalizing it in the interval $[0, 1]$, making our method capable of generating in-between keyframe

models with high efficiency and accuracy. This allows to eliminate an enormous amount of time spent on PBD simulation, resulting in our proposed method being time-saving. The comparison results to other baseline methods demonstrate that our presented method can reconstruct deformable models with small data sizes while maintaining most details, which can be the answer to **Q1**.

In Chapter 5, a physics-based skin deformation method is proposed, which integrates an improved ODE-based surface creation method with Newton's second law to create natural facial animation. The blendshape models created by our method have natural C^2 continuities between adjacent patches, controllable shapes, and small data sizes, which ensure the superior performance of our method in real-time scenarios. This skin deformation method can be the answer to **Q2**.

In Chapter 6, a novel Video-to-Motion(VTM) framework is presented, which reconstructs human motion from monocular videos in a two-part strategy. VTM first utilizes MoAE to learn well-defined motion priors and then employs the motion priors for predicting motions from video inputs. With comprehensive experiments, VTM has been demonstrated to be effective and robust for precisely predicting high-quality motions from unseen view angles and reconstructing motions from in-the-wild videos. Its performance in achieving realistic animation of digital human models synchronized with video sequences indicates that, it can be applied in interactive systems. This framework can be the answer to **Q3**.

7.1 Limitations and Future Work

Although the stated objectives of this thesis are achieved as mentioned above, there still exist several future works required to be further studied in the proposed methods.

Dynamic modelling. There are future works in dynamic modelling techniques. For example, an automated and optimal curve extraction method is required to be investigated for reducing labour costs. This method first determines the maximum reconstruction error E_{max}. According to the limitation of E_{max}, we can automatically identify the vertices of a single curve and calculate the reconstruction error E_{rec}, comparing it to E_{max}. If E_{rec} is less than E_{max}, more vertices should be added to the curve. If E_{rec} exceeds E_{max}, some selected vertices should

be removed. This process is repeated until E_{rec} is less than or equal to E_{max}.

Furthermore, how to determine optimal terms in Eq. 4.24 and optimal values of the parameters m_S and D_S to achieve optimal balance among the computational efficiency, reconstruction accuracy, and data sizes have not been investigated in this research. The work given in this research only investigates the integration between PBD and PDE-based dynamic reconstruction. It can be extended to integrate PDE-based dynamic reconstruction with other numerical methods such as the finite element method, finite difference method, mass-spring systems, etc. Apart from reconstructing new shapes from known shapes obtained by PBD simulation, how to introduce dynamic PDE-based simulation into the PBD algorithm and other numerical methods to improve computational efficiency and convergence can also be a future research topic.

Skin deformation. Our proposed skin deformation method can also be enhanced. For example, the wireframe extraction procedure involves manual work, which is tedious and time-consuming. This manual work can be saved by a procedural extraction method to parametrize face models. By introducing a template face model with pre-segmented surface patches, we can modify the input models by manipulating the control parameters. Besides, since we only investigate facial blendshapes using interpolation between a neutral pose model and a laugh pose model, facial blendshapes among a neutral pose model and many target models with different poses have not been studied, which can be tackled by developing new algorithms of facial blendshapes based on Newton's second law. Moreover, except Newton's second law, there are many physical laws required to be investigated for integration, which can generate different animation effects. Furthermore, the potential of introducing our method as a constraint into PBD to create more realistic skin deformation results with high efficiency requires to be investigated.

Motion reconstruction. With the help of professional motion capture systems, various high-quality human motion datasets have been proposed, like CMU mocap[1]. Though they provide sufficient motion information, the paired video data of the motions are lacking, which makes them incapable of supervised learning. We will explore the potential of leveraging these datasets with readily available videos in unsupervised

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

or semi-supervised learning. Besides, since the videos in the AIST++ dataset are all captured with simple backgrounds, we will develop background augmentation approaches to integrate the background variation for more robust training.

Entire system. After researching the techniques for different stages in creating realistic animation, we plan to develop a framework to reconstruct the complete animation of digital characters, especially digital humans, from videos, with real-time performance and accuracy being focused. Thus, the target of this research can be regarded as being truly achieved.

Bibliography

Abdel-Aziz, H., Zanaty, E., Ali, H. A. and Saad, M. K., 2021. Generating bézier curves for medical image reconstruction. *Results in Physics*, 23, 103996.

Abhyankar, S., 1973. *Algebraic space curves*. University Montreal. Séminaire de Mathématiques Supérieures: Publications, Presses de l'Univ. de Montréal.

Ahmat, N., Ugail, H. and Castro, G. G., 2011. Method of modelling the compaction behaviour of cylindrical pharmaceutical tablets. *International journal of pharmaceutics*, 405 (1-2), 113–121.

Akhter, I. and Black, M. J., 2015. Pose-conditioned joint angle limits for 3d human pose reconstruction. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1446–1455.

Allaire, G., Jouve, F. and Toader, A.-M., 2004. Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics*, 194 (1), 363–393.

Andriluka, M., Pishchulin, L., Gehler, P. and Schiele, B., 2014. 2d human pose estimation: New benchmark and state of the art analysis. *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, 3686–3693.

Ao, T., Zhang, Z. and Liu, L., 2023. Gesturediffuclip: Gesture diffusion model with clip latents. *ACM Transactions on Graphics (TOG)*, 42 (4), 1–18.

Baak, A., Müller, M., Bharaj, G., Seidel, H.-P. and Theobalt, C., 2013. A data-driven approach for real-time full body pose reconstruction from a depth camera. *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*, 71–98.

Bailey, S. W., Otte, D., Dilorenzo, P. and O'Brien, J. F., 2018. Fast and deep deformation approximations. *ACM Transactions on Graphics (TOG)*, 37 (4), 1–12.

Bajaj, C. L., 1988. Geometric modeling with algebraic surfaces.

Baran, I. and Popović, J., 2007. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)*, 26 (3), 72–es.

Barrielle, V., Stoiber, N. and Cagniart, C., 2016. Blendforces: A dynamic framework for facial animation. *Computer Graphics Forum*, Wiley Online Library, volume 35, 341–352.

Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R. and Srinivasan, P. P., 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5855–5864.

Bay, H., Tuytelaars, T. and Van Gool, L., 2006. Surf: Speeded up robust features. *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, Springer, 404–417.

Bender, J., Dequidt, J., Duriez, C. and Zachmann, G., 2013. Physically-based character skinning. *Virtual Reality Interactions and Physical Simulations (VRIPhys) nov*, 4.

Bender, J., Erleben, K. and Trinkle, J., 2014a. Interactive simulation of rigid body dynamics in computer graphics. *Computer Graphics Forum*, Wiley Online Library, volume 33, 246–270.

Bender, J., Koschier, D., Charrier, P. and Weber, D., 2014b. Position-based simulation of continuous materials. *Computers & Graphics*, 44, 1–10.

Bender, J., Müller, M. and Macklin, M., 2015. Position-based simulation methods in computer graphics. *Eurographics (tutorials)*, 8.

Bender, J., Müller, M. and Macklin, M., 2017. A survey on position based dynamics, 2017. *Proceedings of the European Association for Computer Graphics: Tutorials*, 1–31.

Bender, J., Müller, M., Otaduy, M. A., Teschner, M. and Macklin, M., 2014c. A survey on position-based simulation methods in computer graphics. *Computer graphics forum*, Wiley Online Library, volume 33, 228–251.

Berndt, I., Torchelsen, R. and Maciel, A., 2017. Efficient surgical cutting with position-based dynamics. *IEEE computer graphics and applications*, 37 (3), 24–31.

Bertiche, H., Madadi, M., Tylson, E. and Escalera, S., 2021. Deepsd: Automatic deep skinning and pose space deformation for 3d garment animation. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5471–5480.

Bian, S., Deng, Z., Chaudhry, E., You, L., Yang, X., Guo, L., Ugail, H., Jin, X., Xiao, Z. and Zhang, J. J., 2019. Efficient and realistic character animation through analytical physics-based skin deformation. *Graphical Models*, 104, 101035.

Bleyer, M., Rhemann, C. and Rother, C., 2011. Patchmatch stereo-stereo matching with slanted support windows. *Bmvc*, volume 11, 1–11.

Bloor, M. and Wilson, M., 1989. Generating blend surfaces using partial differential equations. *Computer-aided design*, 21 (3), 165–171.

Bloor, M. I. and Wilson, M. J., 1990. Using partial differential equations to generate free-form surfaces. *Computer-Aided Design*, 22 (4), 202–212.

Bogo, F., Kanazawa, A., Lassner, C., Gehler, P., Romero, J. and Black, M. J., 2016. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, Springer, 561–578.

Bose, N. K., 1995. *Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory*, Dordrecht: Springer Netherlands. 89–127. URL `https://doi.org/10.1007/978-94-017-0275-1_4`.

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P. and Lévy, B., 2010. *Polygon mesh processing*. CRC press.

Bouaziz, S., Martin, S., Liu, T., Kavan, L. and Pauly, M., 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33 (4). URL https://doi.org/10.1145/2601097.2601116.

Brau, E. and Jiang, H., 2016. A bayesian part-based approach to 3d human pose and camera estimation. *2016 23rd International Conference on Pattern Recognition (ICPR)*, IEEE, 1762–1767.

Burenius, M., Sullivan, J. and Carlsson, S., 2013. 3d pictorial structures for multiple view articulated pose estimation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3618–3625.

Cao, S. and Ji, G., 2021. Automatically generating layouts of large-scale office park using position-based dynamics. *Proceedings of the 26th CAADRIA Conference*, 21–30.

Capell, S., Burkhart, M., Curless, B., Duchamp, T. and Popović, Z., 2005. Physically based rigging for deformable characters. *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 301–310.

Chadwick, J. E., Haumann, D. R. and Parent, R. E., 1989. Layered construction for deformable animated characters. *ACM Siggraph Computer Graphics*, 23 (3), 243–252.

Chang, J. Y. and Nam, S. W., 2013. Fast random-forest-based human pose estimation using a multi-scale and cascade approach. *ETRI Journal*, 35 (6), 949–959.

Chaudhry, E., Bian, S., Ugail, H., Jin, X., You, L. and Zhang, J. J., 2015. Dynamic skin deformation using finite difference solutions for character animation. *Computers & Graphics*, 46, 294–305.

Chen, A., Xu, Z., Geiger, A., Yu, J. and Su, H., 2022. Tensorf: Tensorial radiance fields. *European Conference on Computer Vision*, Springer, 333–350.

Chen, C.-H. and Ramanan, D., 2017. 3d human pose estimation= 2d pose estimation+ matching. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7035–7043.

Chen, L., Ai, H., Chen, R., Zhuang, Z. and Liu, S., 2020. Cross-view tracking for multi-human 3d pose estimation at over 100 fps. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3279–3288.

Ci, H., Wu, M., Zhu, W., Ma, X., Dong, H., Zhong, F. and Wang, Y., 2023. Gfpose: Learning 3d human pose prior with gradient fields. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4800–4810.

Dahl, A. and Bargteil, A., 2019. Global momentum preservation for position-based dynamics. *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games*, 1–5.

Dalal, N. and Triggs, B., 2005. Histograms of oriented gradients for human detection. *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Ieee, volume 1, 886–893.

Daniels, J., Silva, C. T., Shepherd, J. and Cohen, E., 2008. Quadrilateral mesh simplification. *ACM transactions on graphics (TOG)*, 27 (5), 1–9.

Davydov, A., Remizova, A., Constantin, V., Honari, S., Salzmann, M. and Fua, P., 2022. Adversarial parametric pose prior. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10997–11005.

Deng, K., Liu, A., Zhu, J.-Y. and Ramanan, D., 2022. Depth-supervised nerf: Fewer views and faster training for free. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12882–12891.

DeRose, T., Kass, M. and Truong, T., 2023. Subdivision surfaces in character animation. *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 801–810.

Deul, C., Charrier, P. and Bender, J., 2016. Position-based rigid-body dynamics. *Computer Animation and Virtual Worlds*, 27 (2), 103–112.

Dinev, D., Liu, T., Li, J., Thomaszewski, B. and Kavan, L., 2018. Fepr: Fast energy projection for real-time simulation of deformable objects. *ACM Transactions on Graphics (TOG)*, 37 (4), 1–12.

Ebke, H.-C., Campen, M., Bommes, D. and Kobbelt, L., 2014. Level-of-detail quad meshing. *ACM Transactions on Graphics (TOG)*, 33 (6), 1–11.

Ekman, P. and Friesen, W. V., 1978. Facial action coding system. *Environmental Psychology & Nonverbal Behavior*.

English, E. and Bridson, R., 2008. Animating developable surfaces using nonconforming elements. *ACM SIGGRAPH 2008 papers*, 1–5.

Fischler, M. A. and Elschlager, R. A., 1973. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100 (1), 67–92.

Frâncu, M. and Moldoveanu, F., 2017. Unified simulation of rigid and flexible bodies using position based dynamics. *VRIPHYS*, 49, 58.

Freitas-Magalhães, A., 2013. The face of lies.

Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B. and Kanazawa, A., 2022. Plenoxels: Radiance fields without neural networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5501–5510.

Fu, G., Huang, M., Bo, W., Hao, H. and Wu, R., 2018. Mapping morphological shape as a high-dimensional functional curve. *Briefings in bioinformatics*, 19 (3), 461–471.

Fu, H., Bian, S., Li, O., Macey, J., Iglesias, A., Chaudhry, E., You, L. and Zhang, J. J., 2022. 3d modelling with c 2 continuous pde surface patches. *Mathematics*, 10 (3), 319.

Ganapathi, V., Plagemann, C., Koller, D. and Thrun, S., 2012. Real-time human pose tracking from range data. *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*, Springer, 738–751.

Garbin, S. J., Kowalski, M., Johnson, M., Shotton, J. and Valentin, J., 2021. Fastnerf: High-fidelity neural rendering at 200fps. *Proceedings of the IEEE/CVF international conference on computer vision*, 14346–14355.

Geng, Z., Wang, C., Wei, Y., Liu, Z., Li, H. and Hu, H., 2023. Human pose as compositional tokens. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 660–671.

Georgii, J. and Westermann, R., 2006. A multigrid framework for real-time simulation of deformable bodies. *Computers & Graphics*, 30 (3), 408–415.

Geyer, L., 2022. *Adaptive sampling in position based fluids*. Ph.D. thesis, Wien.

Ghali, S., 2008. Constructive solid geometry. *Introduction to geometric computing*, 277–283.

Gibson, S. F. and Mirtich, B., 1997. A survey of deformable modeling in computer graphics. Technical report, Technical report, Mitsubishi Electric Research Laboratories.

Gladilin, E., Zachow, S., Deuflhard, P. and Hege, H. C., 2004. Anatomy- and physics-based facial animation for craniofacial surgery simulations. *Medical and Biological Engineering and Computing*, 42, 167–170.

Goel, S., Pavlakos, G., Rajasegaran, J., Kanazawa, A. and Malik, J., 2023. Humans in 4d: Reconstructing and tracking humans with transformers. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14783–14794.

Gong, K., Zhang, J. and Feng, J., 2021. Poseaug: A differentiable pose augmentation framework for 3d human pose estimation. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 8575–8584.

Gu, Y., Yang, K. and Lu, C., 2017. Constraint solving order in position based dynamics. *2017 2nd International Conference on Electrical, Automation and Mechanical Engineering (EAME 2017)*, Atlantis Press, 191–194.

Guo, L., Xue, W., Guo, Q., Liu, B., Zhang, K., Yuan, T. and Chen, S., 2023. Distilling cross-temporal contexts for continuous sign language recognition. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10771–10780.

Guo, Q., 2022. The status quo, key and future of virtual digital human development. *Journalism and Writing*, 7.

Hamm, J., Kohler, C. G., Gur, R. C. and Verma, R., 2011. Automated facial action coding system for dynamic analysis of facial expressions in neuropsychiatric disorders. *Journal of neuroscience methods*, 200 (2), 237–256.

Han, Y., Liu, F. and Yip, M. C., 2020. A 2d surgical simulation framework for tool-tissue interaction. *arXiv preprint arXiv:2010.13936*.

Haraway, D., 1985. A manifesto for cyborgs.

Hatze, H., 1997. A three-dimensional multivariate model of passive human joint torques and articular boundaries. *Clinical Biomechanics*, 12 (2), 128–135.

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Holden, D., Duong, B. C., Datta, S. and Nowrouzezahrai, D., 2019. Subspace neural physics: Fast data-driven interactive simulation. *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 1–12.

Holden, D., Komura, T. and Saito, J., 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 36 (4), 1–13.

Hoschek, J. and Lasser, D., 1993. *Fundamentals of computer aided geometric design*. AK Peters, Ltd.

Hossain, M. R. I. and Little, J. J., 2018. Exploiting temporal information for 3d human pose estimation. *Proceedings of the European conference on computer vision (ECCV)*, 68–84.

Hou, S., Xu, W., Chai, J., Wang, C., Zhuang, W., Chen, Y., Bao, H. and Wang, Y., 2021. A causal convolutional neural network for motion modeling and synthesis. *arXiv preprint arXiv:2101.12276*.

Huang, P.-H., Matzen, K., Kopf, J., Ahuja, N. and Huang, J.-B., 2018. Deepmvs: Learning multi-view stereopsis. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2821–2830.

Insafutdinov, E., Campbell, D., Henriques, J. F. and Vedaldi, A., 2022. Snes: Learning probably symmetric neural surfaces from incomplete data. *European Conference on Computer Vision*, Springer, 367–383.

Ionescu, C., Li, F. and Sminchisescu, C., 2011. Latent structured models for human pose estimation. *2011 International Conference on Computer Vision*, IEEE, 2220–2227.

Ionescu, C., Papava, D., Olaru, V. and Sminchisescu, C., 2013. 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.

Jacobson, A., Baran, I., Kavan, L., Popović, J. and Sorkine, O., 2012. Fast automatic skinning transformations. *ACM Transactions on Graphics (TOG)*, 31 (4), 1–10.

Jacobson, A. and Sorkine, O., 2011. Stretchable and twistable bones for skeletal shape deformation. *Proceedings of the 2011 SIGGRAPH Asia Conference*, 1–8.

Jakobsen, T., 2001. Advanced character physics. *Game developers conference*, IO Interactive, Copenhagen Denmark, volume 3, 383–401.

Kavan, L., Collins, S., Žára, J. and O'Sullivan, C., 2007. Skinning with dual quaternions. *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, 39–46.

Kavan, L., Collins, S., Žára, J. and O'Sullivan, C., 2008. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)*, 27 (4), 1–23.

Kavan, L. and Sorkine, O., 2012. Elasticity-inspired deformers for character articulation. *ACM Transactions on Graphics (TOG)*, 31 (6), 1–8.

Kavan, L. and Žára, J., 2005. Spherical blend skinning: a real-time deformation of articulated models. *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, 9–16.

Kerbl, B., Kopanas, G., Leimkühler, T. and Drettakis, G., 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 1–14.

Kim, J.-H., Kim, S.-J. and Lee, J., 2022. Geometry image super-resolution with anisocbconvnet architecture for efficient cloth modeling. *PloS one*, 17 (8), e0272433.

Kim, T.-Y., Chentanez, N. and Müller-Fischer, M., 2012. Long range attachments-a method to simulate inextensible clothing in computer games. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 305–310.

Kocabas, M., Athanasiou, N. and Black, M. J., 2020. Vibe: Video inference for human body pose and shape estimation. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5253–5263.

Kodek, T. and Munih, M., 2002. Identifying shoulder and elbow passive moments and muscle contributions during static flexion-extension movements in the sagittal plane. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, volume 2, 1391–1396.

Komarichev, A., Hua, J. and Zhong, Z., 2022. Learning geometry-aware joint latent space for simultaneous multimodal shape generation. *Computer Aided Geometric Design*, 93, 102076.

Köster, M. and Krüger, A., 2016. Adaptive position-based fluids: improving performance of fluid simulations for real-time applications. *arXiv preprint arXiv:1608.04721*.

Kry, P. G., James, D. L. and Pai, D. K., 2002. Eigenskin: real time large deformation character skinning in hardware. *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 153–159.

Kurihara, T. and Miyata, N., 2004. Modeling deformable human hands from medical images. *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 355–363.

Lasseter, J., 1998. Principles of traditional animation applied to 3d computer animation. *Seminal graphics: pioneering efforts that shaped the field*, 263–272.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 (11), 2278–2324.

Lee, K., Lee, I. and Lee, S., 2018. Propagating lstm: 3d pose estimation based on joint interdependency. *Proceedings of the European conference on computer vision (ECCV)*, 119–135.

Lee, S.-H., Sifakis, E. and Terzopoulos, D., 2009. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Transactions on Graphics (TOG)*, 28 (4), 1–17.

Lewis, J. P., Cordner, M. and Fong, N., 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, USA: ACM Press/Addison-Wesley Publishing Co., SIGGRAPH '00, 165–172.

Lewis, J. P., Cordner, M. and Fong, N., 2023. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 811–818.

Li, C., Kao, C.-Y., Gore, J. C. and Ding, Z., 2007. Implicit active contours driven by local binary fitting energy. *2007 IEEE conference on computer vision and pattern Recognition*, IEEE, 1–7.

Li, C. and Lee, G. H., 2019. Generating multiple hypotheses for 3d human pose estimation with mixture density network. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9887–9895.

Li, J., Bian, S., Zeng, A., Wang, C., Pang, B., Liu, W. and Lu, C., 2021a. Human pose regression with residual log-likelihood estimation. *Proceedings of the IEEE/CVF international conference on computer vision*, 11025–11034.

Li, J., Liu, T. and Kavan, L., 2019. Fast simulation of deformable characters with articulated skeletons in projective dynamics. *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 1–10.

Li, J., Yao, F., Liu, Y. and Wu, Y., 2010. Reconstruction of broken blade geometry model based on reverse engineering. *2010 third international conference on intelligent networks and intelligent systems*, IEEE, 680–682.

Li, R., Yang, S., Ross, D. A. and Kanazawa, A., 2021b. Ai choreographer: Music conditioned 3d dance generation with aist++. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 13401–13412.

Li, S. and Chan, A. B., 2015. 3d human pose estimation from monocular images with deep convolutional neural network. *Computer Vision–ACCV 2014: 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part II 12*, Springer, 332–347.

Li, S., Huang, J., de Goes, F., Jin, X., Bao, H. and Desbrun, M., 2014. Space-time editing of elastic motion through material optimization and reduction. *ACM Transactions on Graphics (TOG)*, 33 (4), 1–10.

Lin, C.-H., Ma, W.-C., Torralba, A. and Lucey, S., 2021. Barf: Bundle-adjusting neural radiance fields. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5741–5751.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C. L., 2014. Microsoft coco: Common objects in context. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, Springer, 740–755.

Ling, H. Y., Zinno, F., Cheng, G. and Van De Panne, M., 2020. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)*, 39 (4), 40–1.

Liu, F., Li, M., Lu, J., Su, E. and Yip, M. C., 2022a. Parameter identification and motion control for articulated rigid body robots using differentiable position-based dynamics. *arXiv preprint arXiv:2201.05753*.

Liu, F., Su, E., Lu, J., Li, M. and Yip, M. C., 2022b. Differentiable robotic manipulation of deformable rope-like objects using compliant position-based dynamics. *arXiv preprint arXiv:2202.09714*.

Liu, H., Zhu, Z., Iwamoto, N., Peng, Y., Li, Z., Zhou, Y., Bozkurt, E. and Zheng, B., 2022c. Beat: A large-scale semantic and emotional multi-modal dataset for conversational gestures synthesis. *European conference on computer vision*, Springer, 612–630.

Liu, L., Gu, J., Zaw Lin, K., Chua, T.-S. and Theobalt, C., 2020a. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33, 15651–15663.

Liu, Y., Guan, C., Li, J., Yu, X. and Zhang, S., 2020b. The pbd model based simulation for soft tissue deformation in virtual surgery. *Journal of Physics: Conference Series*, IOP Publishing, volume 1621, 012043.

Loper, M., Mahmood, N., Romero, J., Pons-Moll, G. and Black, M. J., 2023. Smpl: A skinned multi-person linear model. *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 851–866.

Loshchilov, I. and Hutter, F., 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60, 91–110.

Ma, X., Su, J., Wang, C., Ci, H. and Wang, Y., 2021. Context modeling in 3d human pose estimation: A unified perspective. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 6238–6247.

Macklin, M., Erleben, K., Müller, M., Chentanez, N., Jeschke, S. and Makoviychuk, V., 2019a. Non-smooth newton methods for deformable multi-body dynamics. *ACM Transactions on Graphics (TOG)*, 38 (5), 1–20.

Macklin, M. and Müller, M., 2013. Position based fluids. *ACM Transactions on Graphics (TOG)*, 32 (4), 1–12.

Macklin, M., Müller, M. and Chentanez, N., 2016. Xpbd: position-based simulation of compliant constrained dynamics. *Proceedings of the 9th International Conference on Motion in Games*, 49–54.

Macklin, M., Storey, K., Lu, M., Terdiman, P., Chentanez, N., Jeschke, S. and Müller, M., 2019b. Small steps in physics simulation. *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 1–7.

Macross Wiki, 2024. *The Wiki of Lynn Minmay*. USA, Fandom. Available from: `https://macross.fandom.com/wiki/Lynn_Minmay`. [Accessed 23 December 2024].

Magnenat, T., Laperrière, R. and Thalmann, D., 1988. Joint-dependent local deformations for hand animation and object grasping. *Proceedings of Graphics Interface'88*, Canadian Inf. Process. Soc, 26–33.

Martin-Brualla, R., Radwan, N., Sajjadi, M. S., Barron, J. T., Dosovitskiy, A. and Duckworth, D., 2021. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7210–7219.

Martinez, J., Hossain, R., Romero, J. and Little, J. J., 2017. A simple yet effective baseline for 3d human pose estimation. *Proceedings of the IEEE international conference on computer vision*, 2640–2649.

McAdams, A., Zhu, Y., Selle, A., Empey, M., Tamstorf, R., Teran, J. and Sifakis, E., 2011. Efficient elasticity for character skinning with contact and collisions. *ACM SIGGRAPH 2011 papers*, 1–12.

Menolotto, M., Komaris, D.-S., Tedesco, S., O'Flynn, B. and Walsh, M., 2020. Motion capture technology in industrial applications: A systematic review. *Sensors*, 20 (19), 5687.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R. and Ng, R., 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65 (1), 99–106.

Mirtich, B. V., 1996. *Impulse-based dynamic simulation of rigid body systems*. University of California, Berkeley.

Mohammed, M., Al-Sharify, T., Kolivand, H. et al., 2020. Real-time cloth simulation on virtual human character using enhanced position based dynamic framework technique. *Baghdad Science Journal*, 17 (4), 1294–1294.

Mohr, A. and Gleicher, M., 2003. Building efficient, accurate character skins from examples. *ACM Transactions on Graphics (TOG)*, 22 (3), 562–568.

Monaghan, J. J., 1992. Smoothed particle hydrodynamics. *In: Annual review of astronomy and astrophysics. Vol. 30 (A93-25826 09-90), p. 543-574.*, 30, 543–574.

Moreno-Noguer, F., 2017. 3d human pose estimation from a single image via distance matrix regression. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2823–2832.

Müller, M., 2008. Hierarchical position based dynamics.

Müller, M., Heidelberger, B., Hennix, M. and Ratcliff, J., 2007. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18 (2), 109–118.

Müller, M., Macklin, M., Chentanez, N., Jeschke, S. and Kim, T.-Y., 2020. Detailed rigid body simulation with extended position based dynamics. *Computer Graphics Forum*, Wiley Online Library, volume 39, 101–112.

Müller, T., Evans, A., Schied, C. and Keller, A., 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41 (4), 1–15.

Murai, A., Youn Hong, Q., Yamane, K. and Hodgins, J. K., 2017. Dynamic skin deformation simulation using musculoskeletal model and soft tissue dynamics. *Computational Visual Media*, 3, 49–60.

National Library of Medicine, 1994. *The Visible Human Project*. USA, National Library of Medicine. Available from: `https://www.nlm.nih.gov/research/visible/visible_human.html`. [Accessed 23 December 2024].

Nealen, A., Müller, M., Keiser, R., Boxerman, E. and Carlson, M., 2006. Physically based deformable models in computer graphics. *Computer graphics forum*, Wiley Online Library, volume 25, 809–836.

Newell, A., Yang, K. and Deng, J., 2016. Stacked hourglass networks for human pose estimation. *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, Springer, 483–499.

Newzoo, 2024. *Newzoo's Global Games Market Report 2023*. USA, Newzoo. Available from: https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2023-free-version. [Accessed 23 December 2024].

Nguyen-Phuoc, T., Liu, F. and Xiao, L., 2022. Snerf: stylized neural implicit representations for 3d scenes. *arXiv preprint arXiv:2207.02363*.

Niemeyer, M. and Geiger, A., 2021. Giraffe: Representing scenes as compositional generative neural feature fields. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11453–11464.

Oleynikova, H., Millane, A., Taylor, Z., Galceran, E., Nieto, J. and Siegwart, R., 2016. Signed distance fields: A natural representation for both mapping and planning. *RSS 2016 workshop: geometry and beyond-representations, physics, and scene understanding for robotics*, University of Michigan.

Osher, S. and Sethian, J. A., 1988. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79 (1), 12–49.

Otepka, J., Ghuffar, S., Waldhauser, C., Hochreiter, R. and Pfeifer, N., 2013. Georeferenced point clouds: A survey of features and point cloud management. *ISPRS International Journal of Geo-Information*, 2 (4), 1038–1065.

Pan, J., Bai, J., Zhao, X., Hao, A. and Qin, H., 2015. Real-time haptic manipulation and cutting of hybrid soft tissue models by extended position-based dynamics. *Computer Animation and Virtual Worlds*, 26 (3-4), 321–335.

Pan, J., Yang, X., Xie, X., Willis, P. and Zhang, J. J., 2009. Automatic rigging for animation characters with 3d silhouette. *Computer Animation and Virtual Worlds*, 20 (2-3), 121–131.

Pan, Z. and Manocha, D., 2018. Time integrating articulated body dynamics using position-based collocation methods. *International Workshop on the Algorithmic Foundations of Robotics*, Springer, 673–688.

Park, J. J., Florence, P., Straub, J., Newcombe, R. and Lovegrove, S., 2019. Deepsdf: Learning continuous signed distance functions for shape representation. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 165–174.

Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M. and Martin-Brualla, R., 2021a. Nerfies: Deformable neural radiance fields. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5865–5874.

Park, K., Sinha, U., Hedman, P., Barron, J. T., Bouaziz, S., Goldman, D. B., Martin-Brualla, R. and Seitz, S. M., 2021b. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*.

Park, S., Hwang, J. and Kwak, N., 2016. 3d human pose estimation using convolutional neural networks with 2d pose information. *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, Springer, 156–169.

Park, S., Yong Chang, J., Jeong, H., Lee, J.-H. and Park, J.-Y., 2017. Accurate and efficient 3d human pose estimation algorithm using single depth images for pose analysis in golf. *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 49–57.

Patrikalakis, N. M. and Maekawa, T., 2002. *Shape interrogation for computer aided design and manufacturing*, volume 15. Springer.

Pavlakos, G., Choutas, V., Ghorbani, N., Bolkart, T., Osman, A. A., Tzionas, D. and Black, M. J., 2019. Expressive body capture: 3d hands, face, and body from a single image. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10975–10985.

Pavlakos, G., Zhou, X., Derpanis, K. G. and Daniilidis, K., 2017. Coarse-to-fine volumetric prediction for single-image 3d human pose. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7025–7034.

Pavllo, D., Feichtenhofer, C., Grangier, D. and Auli, M., 2019. 3d human pose estimation in video with temporal convolutions and semi-supervised training. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7753–7762.

Peng, S., Zhang, Y., Xu, Y., Wang, Q., Shuai, Q., Bao, H. and Zhou, X., 2021a. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9054–9063.

Peng, X. B., Ma, Z., Abbeel, P., Levine, S. and Kanazawa, A., 2021b. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (ToG)*, 40 (4), 1–20.

Piegl, L. and Tiller, W., 1997. *The NURBS book*. Springer Science & Business Media.

Prautzsch, H., Boehm, W. and Paluszny, M., 2002. *Bézier and B-spline techniques*, volume 6. Springer.

Qi, C. R., Su, H., Mo, K. and Guibas, L. J., 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J. et al., 2021. Learning transferable visual models from natural language supervision. *International conference on machine learning*, PMLR, 8748–8763.

Ramakrishna, V., Munoz, D., Hebert, M., Andrew Bagnell, J. and Sheikh, Y., 2014. Pose machines: Articulated pose estimation via inference machines. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*, Springer, 33–47.

Ravari, A. N. and Taghirad, H. D., 2016. Reconstruction of b-spline curves and surfaces by adaptive group testing. *Computer-Aided Design*, 74, 32–44.

Reiser, C., Peng, S., Liao, Y. and Geiger, A., 2021. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *Proceedings of the IEEE/CVF international conference on computer vision*, 14335–14345.

Rematas, K., Liu, A., Srinivasan, P. P., Barron, J. T., Tagliasacchi, A., Funkhouser, T. and Ferrari, V., 2022. Urban radiance fields. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12932–12942.

Remelli, E., Han, S., Honari, S., Fua, P. and Wang, R., 2020. Lightweight multi-view 3d pose estimation through camera-disentangled representation. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 6040–6049.

Renno, F. and Papa, S., 2015. Direct modeling approach to improve virtual prototyping and fem analyses of bicycle frames. *Engineering Letters*, 23 (4).

Research China, 2024. *Global and China Animation Industry Report, 2019-2025*. USA, Research China. Available from: `http://www.researchinchina.com/Report/ReportInfo.aspx?id=11609`. [Accessed 23 December 2024].

Rodriguez, D., 2013. *Animation Methods: Rigging Made Easy : Rig Your First 3D Character in Maya*. Animation Methods, CreateSpace Independent Publishing Platform.

Roetenberg, D., Luinge, H., Slycke, P. et al., 2009. Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep*, 1 (2009), 1–7.

Romeo, M., Monteagudo, C. and Sánchez-Quirós, D., 2018. Muscle simulation with extended position based dynamics. *CEIG*, 1–10.

Romeo, M., Monteagudo, C. and Sánchez-Quirós, D., 2020. Muscle and fascia simulation with extended position based dynamics. *Computer graphics forum*, Wiley Online Library, volume 39, 134–146.

Roussellet, V., Rumman, N. A., Canezin, F., Mellado, N., Kavan, L. and Barthe, L., 2018. Dynamic implicit muscles for character skinning. *Computers & Graphics*, 77, 227–239.

Sapp, B. and Taskar, B., 2013. Modec: Multimodal decomposable models for human pose estimation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3674–3681.

Schenck, C. and Fox, D., 2018. Spnets: Differentiable fluid dynamics for deep neural networks. *Conference on Robot Learning*, PMLR, 317–335.

Sederberg, T. W. and Parry, S. R., 1986. Free-form deformation of solid geometric models. *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 151–160.

SEMA Game Studio, 2024. *Digital Characters*. USA, Sketch Fab. Available from: `https://sketchfab.com/3d-models`. [Accessed 23 December 2024].

Shao, H., 2022. *Fluids, threads and fibers: towards high performance physics-based modeling and simulation*. Ph.D. thesis.

Shao, X., Liao, E. and Zhang, F., 2017. Improving sph fluid simulation using position based dynamics. *IEEE Access*, 5, 13901–13908.

Sharma, R., Weiss, T. and Kallmann, M., 2020. Plane-based local behaviors for multi-agent 3d simulations with position-based dynamics. *2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, IEEE, 214–217.

Sheng, Y., Sourin, A., Castro, G. G. and Ugail, H., 2010. A pde method for patchwise approximation of large polygon meshes. *The Visual Computer*, 26, 975–984.

Shi, M., Aberman, K., Aristidou, A., Komura, T., Lischinski, D., Cohen-Or, D. and Chen, B., 2020. Motionet: 3d human motion reconstruction from monocular video with skeleton consistency. *Acm transactions on graphics (tog)*, 40 (1), 1–15.

Shirman, L. A. and Sequin, C. H., 1987. Local surface interpolation with bézier patches. *Computer Aided Geometric Design*, 4 (4), 279–295.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A. and Blake, A., 2011. Real-time human pose recognition in parts from single depth images. *CVPR 2011*, Ieee, 1297–1304.

Singh, K. and Kokkevis, E., 2000. Skinning characters using surface oriented free-form deformations. *Graphics interface*, volume 2000, 35–42.

Siyao, L., Yu, W., Gu, T., Lin, C., Wang, Q., Qian, C., Loy, C. C. and Liu, Z., 2022. Bailando: 3d dance generation by actor-critic gpt with choreographic memory. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11050–11059.

Sloan, P.-P. J., Rose III, C. F. and Cohen, M. F., 2001. Shape by example. *Proceedings of the 2001 symposium on Interactive 3D graphics*, 135–143.

Söderlund, H. H., Evans, A. and Akenine-Möller, T., 2022. Ray tracing of signed distance function grids. *Journal of Computer Graphics Techniques Vol*, 11 (3).

Soler, C., Martin, T. and Sorkine-Hornung, O., 2018. Cosserat rods with projective dynamics. *Computer Graphics Forum*, Wiley Online Library, volume 37, 137–147.

Stam, J., 1998. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 139–148.

Stutz, D. and Geiger, A., 2018. Learning 3d shape completion from laser scan data with weak supervision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1955–1964.

Sumner, R. W. and Popović, J., 2004. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23 (3), 399–405.

Sun, C., Sun, M. and Chen, H.-T., 2022. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5459–5469.

Tagliabue, E., Pore, A., Dall'Alba, D., Magnabosco, E., Piccinelli, M. and Fiorini, P., 2020. Soft tissue simulation environment to learn manipulation tasks in autonomous robotic surgery. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 3261–3266.

Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P. P., Barron, J. T. and Kretzschmar, H., 2022. Block-nerf: Scalable

large scene neural view synthesis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8248–8258.

Tatarchenko, M., Dosovitskiy, A. and Brox, T., 2017. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. *Proceedings of the IEEE international conference on computer vision*, 2088–2096.

Tekin, B., Katircioglu, I., Salzmann, M., Lepetit, V. and Fua, P., 2016. Structured prediction of 3d human pose with deep neural networks. *arXiv preprint arXiv:1605.05180*.

Tekin, B., Márquez-Neila, P., Salzmann, M. and Fua, P., 2017. Learning to fuse 2d and 3d image cues for monocular body pose estimation. *Proceedings of the IEEE international conference on computer vision*, 3941–3950.

Teng, Y., Otaduy, M. A. and Kim, T., 2014. Simulating articulated subspace self-contact. *ACM Transactions on Graphics (TOG)*, 33 (4), 1–9.

Terzopoulos, D., Platt, J., Barr, A. and Fleischer, K., 1987. Elastically deformable models. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 205–214.

Theodoridis, T., Chatzis, T., Solachidis, V., Dimitropoulos, K. and Daras, P., 2020. Cross-modal variational alignment of latent spaces. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 960–961.

Timoshenko, S., 1983. *History of strength of materials: with a brief account of the history of theory of elasticity and theory of structures*. Courier Corporation.

Turki, H., Ramanan, D. and Satyanarayanan, M., 2022. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12922–12931.

Turner, R. and Thalmann, D., 1993. The elastic surface layer model for animated character construction. *Communicating with virtual worlds*, Springer, 399–412.

Ugail, H., Bloor, M. I. and Wilson, M. J., 1999. Techniques for interactive design using the pde method. *ACM Transactions on Graphics (TOG)*, 18 (2), 195–212.

Ullman, S., 1979. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203 (1153), 405–426.

Vaillant, R., Barthe, L., Guennebaud, G., Cani, M.-P., Rohmer, D., Wyvill, B., Gourmel, O. and Paulin, M., 2013. Implicit skinning: Real-time skin deformation with contact modeling. *ACM Transactions on Graphics (TOG)*, 32 (4), 1–12.

Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J. T. and Srinivasan, P. P., 2022. Ref-nerf: Structured view-dependent appearance for neural radiance fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 5481–5490.

Walczak, L., Georgii, J., Tautz, L., Neugebauer, M., Wamala, I., Sündermann, S., Falk, V. and Hennemuth, A., 2022. Using position-based dynamics for simulating mitral valve closure and repair procedures. *Computer Graphics Forum*, Wiley Online Library, volume 41, 270–287.

Walczak, L., Georgii, J., Tautz, L., Neugebauer, M., Wamala, I., Sündermann, S. H., Falk, V. and Hennemuth, A., 2019. Using position-based dynamics for simulating the mitral valve in a decision support system. *VCBM*, 165–175.

Walczak, L., Goubergrits, L., Hüllebrand, M., Georgii, J., Falk, V. and Hennemuth, A., 2020. Using position-based dynamics to simulate deformation in aortic valve replacement procedure. *Current Directions in Biomedical Engineering*, De Gruyter, volume 6, 20200042.

Wang, M., Chen, X., Liu, W., Qian, C., Lin, L. and Ma, L., 2018. Drpose3d: Depth ranking in 3d human pose estimation. *arXiv preprint arXiv:1805.08973*.

Wang, Q., Wang, Z., Genova, K., Srinivasan, P. P., Zhou, H., Barron, J. T., Martin-Brualla, R., Snavely, N. and Funkhouser, T., 2021a. Ibrnet: Learning multi-view image-based rendering. *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4690–4699.

Wang, R. Y., Pulli, K. and Popović, J., 2007. Real-time enveloping with rotational regression. *ACM SIGGRAPH 2007 papers*, 73–es.

Wang, S., Wang, R., Xia, Y., Sun, Z., You, L. and Zhang, J., 2021b. Multi-objective aerodynamic optimization of high-speed train heads based on the pde parametric modeling. *Structural and Multidisciplinary Optimization*, 64, 1285–1304.

Wang, S., Xia, Y., Wang, R., You, L. and Zhang, J., 2019. Optimal nurbs conversion of pde surface-represented high-speed train heads. *Optimization and Engineering*, 20, 907–928.

Wang, S., Xiang, N., Xia, Y., You, L. and Zhang, J., 2021c. Real-time surface manipulation with c 1 continuity through simple and efficient physics-based deformations. *The Visual Computer*, 37 (9), 2741–2753.

Wang, X. C. and Phillips, C., 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 129–138.

Wang, Y., Jacobson, A., Barbič, J. and Kavan, L., 2015. Linear subspace design for real-time shape deformation. *ACM Transactions on Graphics (TOG)*, 34 (4), 1–11.

Wang, Z., Wu, S., Xie, W., Chen, M. and Prisacariu, V. A., 2021d. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*.

Warren, J. and Weimer, H., 2001. *Subdivision methods for geometric design: A constructive approach*. Elsevier.

Wei, Y., Liu, S., Rao, Y., Zhao, W., Lu, J. and Zhou, J., 2021. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5610–5619.

Weiss, T., Litteneker, A., Jiang, C. and Terzopoulos, D., 2017. Position-based multi-agent dynamics for real-time crowd simulation. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 1–2.

Wu, J., Westermann, R. and Dick, C., 2015a. A survey of physically based simulation of cuts in deformable bodies. *Computer Graphics Forum*, Wiley Online Library, volume 34, 161–187.

Wu, J., Zhang, C., Xue, T., Freeman, B. and Tenenbaum, J., 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X. and Xiao, J., 2015b. 3d shapenets: A deep representation for volumetric shapes. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1912–1920.

Xie, R., Wang, C. and Wang, Y., 2020. Metafuse: A pre-trained fusion model for human pose estimation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13686–13695.

Xu, H. and Barbič, J., 2016. Pose-space subspace dynamics. *ACM Transactions on Graphics (TOG)*, 35 (4), 1–14.

Xu, L., Lu, Y. and Liu, Q., 2018. Integrating viscoelastic mass spring dampers into position-based dynamics to simulate soft tissue deformation in real time. *Royal Society open science*, 5 (2), 171587.

Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K. and Neumann, U., 2022. Point-nerf: Point-based neural radiance fields. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5438–5448.

Xu, X., Liu, L. and Yan, S., 2023. Smpler: Taming transformers for monocular 3d human shape and pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xu, Z., Zhou, Y., Kalogerakis, E., Landreth, C. and Singh, K., 2020. Rignet: Neural rigging for articulated characters. *arXiv preprint arXiv:2005.00559*.

Yang, B., Zhang, Y., Xu, Y., Li, Y., Zhou, H., Bao, H., Zhang, G. and Cui, Z., 2021. Learning object-compositional neural radiance field for editable scene rendering. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 13779–13788.

Yang, S., He, X. and Zhu, B., 2020. Learning physical constraints with neural projections. *Advances in Neural Information Processing Systems*, 33, 5178–5189.

Yang, W., Chen, G., Chen, C., Chen, Z. and Wong, K.-Y. K., 2022. S3-nerf: Neural reflectance field from shading and shadow under a single viewpoint. *Advances in Neural Information Processing Systems*, 35, 1568–1582.

Yang, X., Somasekharan, A. and Zhang, J. J., 2006. Curve skeleton skinning for human and creature characters. *Computer Animation and Virtual Worlds*, 17 (3-4), 281–292.

Yao, Y., Luo, Z., Li, S., Shen, T., Fang, T. and Quan, L., 2019. Recurrent mvsnet for high-resolution multi-view stereo depth inference. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5525–5534.

Ye, M., Wang, X., Yang, R., Ren, L. and Pollefeys, M., 2011. Accurate 3d pose estimation from a single depth image. *2011 International Conference on Computer Vision*, IEEE, 731–738.

Ye, M. and Yang, R., 2014. Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2345–2352.

Yen-Chen, L., Florence, P., Barron, J. T., Rodriguez, A., Isola, P. and Lin, T.-Y., 2021. inerf: Inverting neural radiance fields for pose estimation. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 1323–1330.

You, L., Ugail, H., Tang, B., Jin, X., You, X. Y. and Zhang, J. J., 2014. Blending using ode swept surfaces with shape control and cˆ 1 c 1 continuity. *The Visual Computer*, 30, 625–636.

You, L., Yang, X., Pachulski, M. and Zhang, J. J., 2007. Boundary constrained swept surfaces for modelling and animation. *Computer Graphics Forum*, Wiley Online Library, volume 26, 313–322.

Yu, A., Li, R., Tancik, M., Li, H., Ng, R. and Kanazawa, A., 2021. Plenoctrees for real-time rendering of neural radiance fields. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5752–5761.

Zhan, Y., Li, F., Weng, R. and Choi, W., 2022. Ray3d: ray-based 3d human pose estimation for monocular absolute 3d localization. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13116–13125.

Zhang, J., Zhang, Y., Cun, X., Huang, S., Zhang, Y., Zhao, H., Lu, H. and Shen, X., 2023. T2m-gpt: Generating human motion from textual descriptions with discrete representations. *arXiv preprint arXiv:2301.06052*.

Zhang, J., Zhang, Y., Fu, H., Zhou, X., Cai, B., Huang, J., Jia, R., Zhao, B. and Tang, X., 2022. Ray priors through reprojection: Improving neural radiance fields for novel view extrapolation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18376–18386.

Zhang, K., Riegler, G., Snavely, N. and Koltun, V., 2020. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*.

Zhang, L., Chen, S. and Zou, B., 2021. Estimation of 3d human pose using prior knowledge. *Journal of Electronic Imaging*, 30 (4), 040502–040502.

Zhang, Y., Matuszewski, B. J., Shark, L.-K. and Moore, C. J., 2008. Medical image segmentation using new hybrid level-set method. *2008 fifth international conference biomedical visualization: information visualization in medical and biomedical informatics*, IEEE, 71–76.

Zhao, H., Lei, N., Li, X., Zeng, P., Xu, K. and Gu, X., 2018. Robust edge-preserving surface mesh polycube deformation. *Computational Visual Media*, 4, 33–42.

Zhou, K., Han, X., Jiang, N., Jia, K. and Lu, J., 2019a. Hemlets pose: Learning part-centric heatmap triplets for accurate 3d human pose estimation. *Proceedings of the IEEE/CVF international conference on computer vision*, 2344–2353.

Zhou, Q. and Jacobson, A., 2016. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*.

Zhou, X., Sun, X., Zhang, W., Liang, S. and Wei, Y., 2016. Deep kinematic pose regression. *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, Springer, 186–201.

Zhou, Y., Barnes, C., Lu, J., Yang, J. and Li, H., 2019b. On the continuity of rotation representations in neural networks. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5745–5753.

Zucker, M., Bagnell, J. A., Atkeson, C. G. and Kuffner, J., 2010. An optimization approach to rough terrain locomotion. *2010 IEEE International Conference on Robotics and Automation*, IEEE, 3589–3595.