博　士　論　文

Automatic Deformation Refinement for Animated Characters via Graph Neural Networks (グラフニューラルネットワークを用いたキャラクタ変形の自動精細化)

李　天行

Acknowledgement

I would like to thank:

Professor Takashi Kanai, who provided me a precious opportunity to come to the lab and gave me constant support. Without your patience and advice, I would never have a strong interest in the field of computer graphics, and have learned knowledge and skills far more than ever before. You acted as a role model to me not only academically but also in attitude. I hope I can become a teacher like you in the future.

The committee, who patiently gave me a lot of suggestions for completing this thesis. With these valuable comments, I can better and logically summarize the research during my Ph.D.

Rui Shi, who is my collaborator, my partner, and my best friend. Your advice, idea, and encouragement have accompanied me throughout my Ph.D. period. Thank you for spending countless moments of depression and excitement with me.

My family, my parents, grandparents, and other family members always here for me. They have always tolerated me and allowed me to do things of interest as carefree as a child. They are willing to help me in any way at any time. Special thanks to my grandma, although I will never see her again, her encouragement and love are always transmitted through my dreams.

Myself, my persistence, my optimism, and my luck.

Abstract

In animation production, animators always spend significant time and efforts to develop high-quality deformation systems for characters with complex appearances and details. However, achieving realistic deformations in real-time has always been a challenging task. Traditional geometry-based skinning methods are popular in interactive applications for their high performance but fail to generate convincing deformations, which have obvious artefacts. By formulating the deformation process within a simulation framework, physics-based methods can solve the unrealistic problem but they often trade performance for realism. As an alternative to traditional geometry-based and physics-based methods, learning-based methods train a data-driven model to compute deformation as a function of relative parameters, yet they cannot completely solve the generalization problem and cannot generate pose-related deformations, *i.e.*, the generalization to any mesh topology in any posture, and be accompanied by pose-dependent nonlinear effects.

In this work, we propose graph-learning-based, powerfully generalized methods for automatically generating nonlinear deformation for characters with an arbitrary number of vertices. Two cases are declared for different applications, *i.e.*, the 1^{st} case where the body and cloth are taken as a whole object, and the 2^{nd} case where the body and cloth are taken as separate objects. For both two cases, we adopt the idea of regarding mesh deformations as a combination of coarse and refined parts.

For the 1^{st} case, based on the coarse linear-based deformation, we propose novel graph feature representation methods and design feature propagation strategies to automatically generate the nonlinear deformation. Our designed frameworks, densely connected graph-attention-network (DenseGATs) and the multi-resolution graph network (MultiResGNet), can effectively learn the huge amount of features from existing

characters and easily apply them to new objects.

For the 2^{nd} case, in order to use fewer models to produce detailed clothing deformation in different poses, we design the fit parameter to express the suitableness between body and garment which is an important factor affecting wrinkles. Then, we propose an output decomposition solution to narrow the output range, thus making the learning much easier and avoiding overly smooth results. We call the framework as garment-fit-network (GarFitNet).

Experimental results show that our proposed DenseGATs, MultiResGNet, and GarFitNet can achieve better performance than prior studies in deformation approximation for unseen characters and poses.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Purpose

In the current virtual world, the demand for three-dimensional content is growing rapidly. More and more applications and industries such as games, movies, commercials *et al*. require to design virtual worlds, especially to create realistic animated characters. During the process of animating a character, rigging is a crucial task that involves steps of skeleton embedding, motion definition, and deformation system development. In principle, the first two steps can easily achieve automation thanks to the contributions of prior researchers. However, the development of deformation systems is extremely challenging which has been receiving considerable attention. The core problem is that animators need to make choices in terms of realism and time.

For those interactive applications, traditional skinning methods such as linear blend skinning (LBS) [54] and dual quaternion skinning (DQS) [33] are widely adopted because of their simplicity and efficiency. But both techniques tend to produce unrealistic artefacts: volume loss and joint bulging. Despite the manual modification of skinning weights, due to the nature of algorithms, their generated deformations often lack a high-level of detail. For applications to films, the accuracy requirements are greatly increased, physics-based approaches are actively adopted to formulate deformations within simulation frameworks therefore achieving appealing visual effects. However, the amount of computation makes it hard to run at interactive rates.

To make the deformation process easy, learning-based approaches train a data-driven

model that computes deformation as a function of designed feature parameters. Most data-driven approaches use linear models or multi-layer perceptions (MLPs) [9, 52, 63] to achieve the prediction task. Although the solution is simple and easy to implement, its underlying limitation is the dramatic inability to generalize to new characters and mesh topologies. To address the generalization problem and account for the spatial information, several studies achieve the deformation approximation by using graph neural networks (GNNs) due to their ability to reasoning about irregular 3D mesh data. Character meshes are regarded as graphs where each mesh node (vertex) contains its own features and they are then input into the designed GNN. After multiple layers of processing, the network will produce an output for each node based on its state. Such graph-based operation makes training parameters no longer dependent on the number of nodes and workable on general graphs. So far, the generalization problem of training for specific objects seems to be solved via GNNs. However, existing graph-learning-based deformation approaches still suffer from the following weaknesses. First, although the generalization to new mesh topologies becomes possible, the prediction accuracy is not ideal. For example, vanilla GNNs tend to produce overly smooth unrealistic results. The training relies on a large number of training samples and parameters with redundancies, which makes the learning task prone to overfitting. Second, pose-dependent effects are not considered that latest studies are only able to approximate deformation in rest-pose [75] or fixed skinning weights [51]. More elaborate nonlinear effects related to posture (*e.g.*, cloth wrinkles, muscle bulges *et al.*) are limited to achieve.

Based on the context, our overall goal is to successfully achieve deformation approximation in the following three aspects:

- **Automation**: enable the deformation process automated to avoid laborious time-consuming manual painting and fine-tuning work.
- **Accuracy**: generate high-quality nonlinear deformation effects while being suitable for real-time applications.
- **Generalization**: make trained model generalized and be able to efficiently apply the learned feature knowledge to new mesh deformation approximation.

These goals motivate the development of our unified deformation systems, so as to end-to-end achieve fast and accurate nonlinear deformation approximation for diverse characters in various poses.

1.2 Statement of Deformation Cases

Animating digital characters has numerous applications in 3D content production such as entertainment (games, virtual reality, and films) and commerce (online-shopping, virtual try-on). For different applications, the characteristics of animated characters and the technical difficulties that produce their deformations are also different. Therefore, at the beginning, we divide the character deformation into two cases according to different applications and technology.

1.2.1 1^{st} Case: Body and Cloth as a Whole Object

In animation production such as game, cartoon, and animation feature films, the appearance of characters is mostly similar to Figure 1.1.

Figure 1.1: Example characters of the 1^{st} case.

In this situation, the garment and the body always follow closely (the garment is relatively near to the body). The deformation emphasizes the integrity that details of the clothes are often not wrinkled, while people tend to pay more attention to overall

deformation quality like whether there exist artefacts like volume loss near the joints, etc. In the modeling process, animators always design outfits for one specific character which are not shared with other characters. Therefore, in this case, the body and the garment have a unit mesh (as shown in Figure 1.2). Libraries like mixamo [3] and Adobe Fuse cc [1] provide this type's characters that can be directly used in games, virtual reality, and films.

Figure 1.2: Mesh situation of the 1^{st} case. Body and cloth have a unit mesh.

In this context, due to the richness of animated characters, what is lacking is a unified model capable of producing nonlinear deformations for arbitrary animated characters in arbitrary poses. To achieve it, several technical challenges have to be addressed:

- ***Case1-chall.1***: how to encode the various mesh features from diverse animated characters in the arbitrary pose?
- ***Case1-chall.2***: how to achieve the complicated mapping from mesh features to nonlinear offsets for each mesh vertex?
- ***Case1-chall.3***: how to make the training task easy to converge and resolve the gradient vanishing problem?
- ***Case1-chall.4***: how to enhance the generalization ability to unseen characters and poses?

To this end, we introduce two frameworks: DenseGATs (densely connected graph-attention-based network) and MultiResGNet (multi-resolution graph network), both of which effectively approximate deformations in each pose step by adding nonlinear refinements. Meanwhile, they all inherit the advantages of GNNs that allow generalization to any mesh topology and achieve stronger generalization through our improvements.

1.2.2 2^{nd} Case: Body and Cloth as Separate Objects

In many application areas including virtual try-on, online shopping, and fashion design games, the digitization of clothing (as shown in Figure 1.3) is a long-standing goal in computer animation. In these scenarios, how the garment interacts with the human body usually gets more attention from users. Computer graphics technology promises an opportunity to support these applications through deformation approximation, but to date deformation approximation solutions lack the visual sense of real clothes deformation.

Figure 1.3: Example characters of the 2^{nd} case.

In this situation, different fits or designs of garments usually have different interactions with bodies, such as detailed folds. As the pose changes, these details will increase/decrease and move around in various positions of garments. More realistic contact effects will give users a better experience, and guide their purchase in online

shopping or arouse their interest in continuously playing games.

Therefore, in this case, the body and the garment have separate meshes (as shown in Figure 1.4) where the designed garments are usually worn by many bodies. Softwares such as Marvelous Designer [2], TUKA3D [5], and Optitex [4] provide a convenient way to make realistic garments and drape them to avatars.

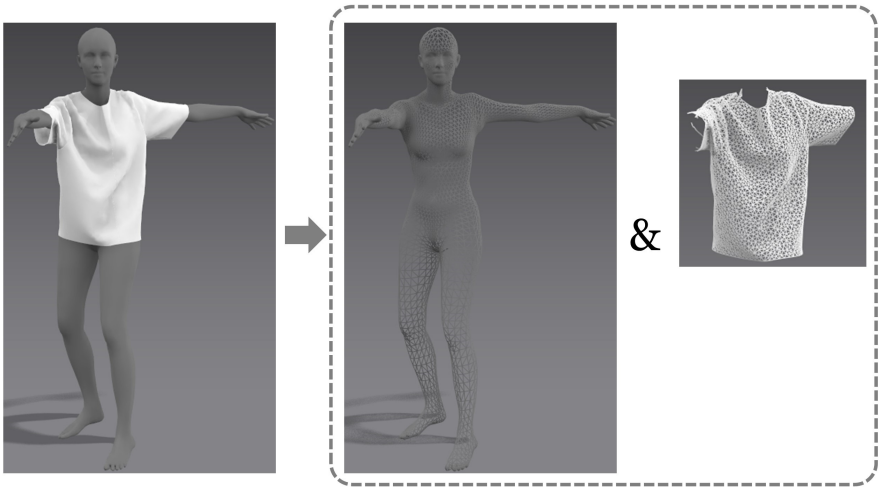


Figure 1.4: Mesh situation of the 2^{nd} case. Body and cloth have their own mesh.

Compared with the deformation of the 1^{st} case, this 2^{nd} case's deformation tends to represent more complicated wrinkles of the clothing appearance. Most previous data-driven cloth deformation approaches only work for a specific garment topology or for a specific pose to guarantee the realism of the approximated result. However, models that do not have generalization are obviously limited to large-scale predictions and cannot meet practical needs. We conclude the major limitations and challenges in related studies as follows:

- ***Case2-chall.1***: how to find and encode the implicit features that have an important influence on the detail wrinkles?
- ***Case2-chall.2***: how to simplify the learning task while ensuring the output of high-quality deformation?
- ***Case2-chall.3***: how to achieve superior generalization capabilities to unseen garment topologies, body shapes, and animated poses?

These challenges motivate the proposal of our GarFitNet (garment-fit-based network), which is able to generate deformations for diverse garments and animate them on any body shape in any pose while retaining rich wrinkle details.

1.3 Contributions

Our research provides artists with a novel perspective of deformation calculations, which can automatically generate better visual effects than traditional geometry-based techniques and other state-of-art learning-based approaches while keeping real-time performance. To make a better performance to achieve the automatic approximation task, we solve the problem of huge amounts of complicated learned features and the problem of weak generalization ability from three aspects: (a), designing input features, (b), designing the novel convolution operation and network structures, and (c), reconstructing output structures.

To summarize, the main contributions of our work for the overall deformation approximation are:

- To achieve deformations for animated characters in different applications, we first propose to divide the character deformation into two cases in order to meet the quality requirements in different scenarios.
- To make deformations learnable, we create two datasets for two cases with various deformations (characters and garments) in possible poses. With the datasets, we then use them to train and test to verify the effectiveness of our proposed methods.

For the first case deformation, the main technical contributions of our work are:

- To express the features of meshes, we propose novel graph construction solutions and introduce mesh descriptors for each mesh node which encode the skinning features along with poses, geometry attributes of meshes, and the relationships between meshes and skeletons. (*Solution for* ***case1-chall.1***)
- To deal with arbitrarily structured mesh graph data, our method leverages graph-attention-based (GAT) blocks for effectively learning complicated mesh features.

Specifically, in each GAT block, we extend the original GAT structure by adding a self-reinforced stream that linearly maps the individual features of each node. This stream is then concatenated with the graph-attention-based aggregation stream to form a GAT block. In this way, the GAT block can effectively compile information on complex graph features from neighboring vertices as well as information on self-features. (*Solution for* ***case1-chall.2***)

- To extract high-level features with deep layers and ensure the propagation of information on a large amount of features, we introduce "dense module" that adopts dense connectivity patterns between several GAT blocks to resolve the vanishing gradient problem and effectively improve information flow by fusing multiple levels of features. (*Solution for* ***case1-chall.3***)

- To improve the generalization ability of the network, we propose a novel multi-resolution graph network called MultiResGNet enabling the reuse of existing artist-created skinning features and the easy application to new character meshes. To improve the ability of expressing arbitrary graphs, we designed the MultiResGNet with two branches, *i.e.*, a global branch that deals with lower-resolution graphs for integrating structural information and a local branch that handles original-resolution graphs for enhancing and propagating detail features. To address the limitations of insufficient training samples, for the global branch, we designed graph pooling and corresponding unpooling operations to extract the holistic features of various characters and poses, thereby realizing better generalization and performance. (*Solution for* ***case1-chall.4***)

For the second case deformation, the main technical contributions of our work are:

- To account for complicated and irregular detailed wrinkles, we first discuss that the fit between garment and body influences the degree of wrinkles: loose clothes have smoother, sparse, and wide folds, while tight clothes have thinner, denser, and narrow folds. We therefore propose the fit parameter and transfer it through the novel GarFit parameter transfer branch. (*Solution for* ***case2-chall.1***)

- To make the learning task easy and avoid overly smooth results, we narrow the range of output, *i.e.*, vertex displacement, whose degree of freedom ranges is from negative infinity to positive infinity. We decompose the three-dimensional output vector as the combination of length and unit, where the range of length is greater than zero and the range of the unit is from -1 to 1. Based on this decomposition, we design a novel loss function to reduce the distance of the prediction to the ground truth and make training easier to converge. (*Solution for* ***case2-chall.2***)

- To improve the generalization ability for the clothing deformation model, we design a novel garment-fit-based network named GarFitNet which consists of three branches, *i.e.*, the GarFit parameter transformer branch, the unit direction prediction branch, and the length prediction branch. Specifically, the GarFit parameter transformer branch can generate more powerful feature representations so that the model can use this information to predict detailed folds in a more targeted manner. The unit prediction branch and the length prediction branch are used to respectively predict the unit and the length of output displacements. This separate prediction solution keeps the approximation error within a very small range when predicting deformations for unseen geometries, body shapes, and poses. (*Solution for* ***case2-chall.3***)

All in all, the novelty of our research lies in the design of graph-learning based frameworks together with the features of animated meshes to achieve their deformation approximation. Instead of manually setting skinning weights or fine-tuning blend shapes, we utilize existing skinning features from the training set to make predictions for new characters while allowing any number of mesh nodes and arbitrary geometric topologies. We discuss and evaluate the advantages of the proposed solutions, and compare them with existing methods and other network structures. To the best of our knowledge, our approach is the first pose-based, graph-learning-based nonlinear deformation method for data-driven characters.

1.4 Thesis Organization

The main body of our thesis is organized as below. We introduce related works of our research in Chapter 2. For different applications in animation production, the first and second case deformation methods and evaluations will be respectively introduced in Chapter 3 and 4. Finally, Chapter 5 concludes the research and the tasks for future work.

Chapter 2

Related Work

Character animation is an important part of contemporary computer games, animation films, online-shopping, and virtual reality applications. The key challenge for generating credible deformation is to meet the conflicting requirements of real-time interactivity and credibility. Existing research in the area of character deformation, ranging from geometry-based skinning, example-based approaches, physics-based approaches to learning-based methods. Recently, with the increasing interest of graph learning in recent years, applying graph neural networks to reason about irregular 3D data such as meshes has achieved success to some extent.

In this chapter, we first respectively review various techniques of character deformation for two cases. Then, we summarize the graph neural networks for 3D mesh processing. Lastly, we summarize the problems and challenges encountered in the current research and lead to our research goals and proposals.

2.1 Related Works of the 1^{st} Case Deformation

For the first case deformation, most relevant related works can be classified into traditional methods and learning-based methods.

Traditionally, character deformations can be calculated by blending the bone transformations (geometry-based skinning), or by formulating the deformation process within the simulation framework (physics-based skinning). These two types of approaches are excellent in real-time interactivity and quality credibility, but satisfying these two as-

pects at the same time is challenging. To permit more attractive deformation effects for geometry-based methods, some studies generate desired deformations by interpolating a series of examples in some poses. This example-used solution is called the example-based skinning method.

With the development of deep learning, learning-based deformation has gradually gained particular attention recently. Generalizing the learned information to the deformation of new characters in new poses while ensuring high accuracy has always been a typical difficulty for researchers.

In this section, we will respectively introduce the advantages and disadvantages of these classified methods.

2.1.1 Traditional methods

Traditional methods can be categorized into geometry-based skinning, example-based skinning, and physics-based skinning methods.

2.1.1.1 Geometry-based skinning methods

The most common way to deform a character object is to define the mesh surface as a function of the transformation of skeletal joints. Because of the simplicity and intuitive manipulation, geometry-based skinning is widely adopted in real-time applications such as games and virtual reality systems. Here, linear blend skinning and dual quaternions skinning are the two most well-known methods.

Among many proposed geometry-based deformation techniques, linear blend skinning (LBS) is the most popular method because of its directness and effectiveness. Although it has never been formally introduced in the literature, linear blending was first proposed and proven to be able to calculate the deformation of hand in animation [54].

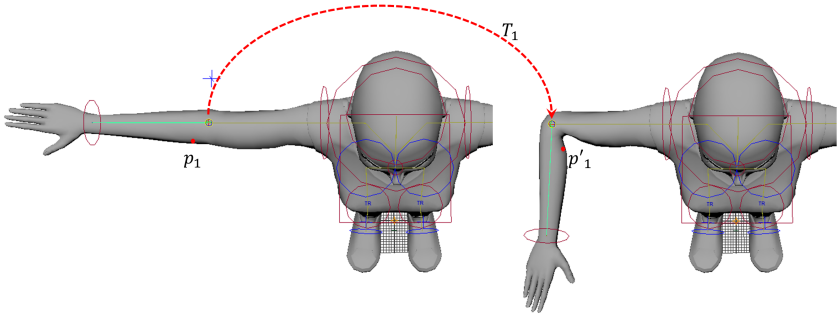


Figure 2.1: An example of illustration the process of LBS. The T_1 is the transformation containing the rotation and translation of the elbow joint.

The process of LBS is demonstrated in Figure 2.1: the surface mesh is deformed according to a given list of joint transformations. Each vertex is associated with skinning weights, which quantifies the influence of each joint on the vertex. Therefore, the basic idea of LBS is to linearly blend the transformation matrices. The whole process can be expressed in a straightforward way:

$$p'_i = \sum_{j=1}^{m} w_{i,j} T_j R_j^{-1} p_i, \tag{2.1}$$

where p_i is a vertex of mesh in rest pose, p'_i is a vertex after deformation. m indicates the total number of skeletal joints. T_j is the transformation matrix of the elbow joints and R_j^{-1} is the inverse transformation of the same bone in the rest pose R. $w_{i,j}$ is a scalar weight that describes the amount of influence of joint j to vertex i.

When the transformation of joints is not rotating largely, the deformation effect of LBS looks good. Issue arises if the joint is blending, the linear interpolation will produce the inadequate location of vertices, which results in the volume loss artefacts. The artefacts are obvious in the areas of joints as depicted in the right sub-figure of Figure 2.1.

To address the limitation mentioned in LBS, dual quaternion skinning (DQS) [33] express the joint transformations based on dual quaternions. The method is just slightly lower than LBS and becomes popular in the professional software like Maya or Blender.

In stead of using the blend matrices $\sum_{j=1}^{m} w_{i,j} T_j R_j^{-1}$ in LBS, DQS blends quaternions $\hat{q}_j$, weighted by $w_{i,j}$. The result is normalized with $\| \sum_{j=1}^{m} w_{i,j} \hat{q}_j \|$ to produce the final

dual quaternion, which is used to transform a vertex from the rest pose to the specific position:

$$p'_i = \hat{q}_j p_i \hat{q}_j{}^*, \tag{2.2}$$

where $\hat{q}_j = \frac{\sum_{j=1}^{m} w_{i,j}\hat{q}_j}{\|\sum_{j=1}^{m} w_{i,j}\hat{q}_j\|}$ is a unit dual quaternion and $\hat{q}_j{}^*$ is the conjugate of $\hat{q}_j$. Details to perform dual quaternion additions can be found in the work of [33]. The method of DQS gives encouraging results and solves the biggest volume loss problem of LBS, but produces the joint-bulging artefacts while bending.

In general, the dual quaternion skinning method is slightly complicated than linear blend skinning in terms of implementation and produces its own artefacts. Despite the drawbacks, DQS is always considered to be a good alternative over LBS since its artefacts are hardly found during the animation and can be avoided by additional manual modification. And the run-time of DQS is almost the same with LBS can be real-time. The appealing extension of DQS is the application to Disney's film Frozen [42].

In addition to LBS and DQS, there are many studies that can be classified into geometry-based skinning methods. What these methods have in common is that the resulting deformation largely relies on the skinning weights, and few works have been proposed to automatically generate skinning weights. Prior studies utilize heat diffusion [10], illumination models [77], Laplacian energy [31], and elastic energy function [35] to calculate skinning weights. After that, Multi-weight enveloping [76] tries learning weights from example poses of rigs. Dionne and de Lasa [22] propose the approach using geodesic voxel binding to compute the skinning weights, which is also capable of handling non-manifold meshes. To compensate for the unnatural deformations of naive LBS and generate more natural and plausible behaviors, helper bone rigs that can create the muscle bulging and soft tissue jiggling effects have been adopted in [57, 58]. Such helper bone rig techniques have been successfully used in practice for game productions. Despite the high flexibility of secondary dynamics, it is difficult to control deformations stably in some cases.

2.1.1.2 Example-based skinning methods

Example-based methods permit more complex skinning effects such as skin slides, muscle bulges, and wrinkling of clothing. Methods treat deformations as a shape interpolation problem that takes a series of scattered data poses as input, to obtain the desired deformation. Those examples can be manually made by the user, including sculpting the character pose into the desired shape. The example-making process is quite long and tedious, which however enables animators to control the exact desired shape. After preparing the example deformations, while animating, these created examples are interpolated to produce realistic transitions for new poses that are not in the set.

One of the first example-based approaches is pose space deformation (PSD) [44] which is also widely adopted in big animation companies. For example, it is used in Walt Disney's "Bolt" for more realistic effects to Rhino the hamster. For the process of PSD method, the geometry-based method like linear blend skinning is first used as the basic deformation, and then mesh shapes are further refined at some key positions by sculpting meshes. The shape deviations between refined meshes and basic meshes are stored as a set of displacements to each pose of the skeleton. Once all example poses are defined, a radial basis function is used to interpolate the displacement vectors among the example poses. In practice, this technology enables users to gradually perfect the deformation. Whenever the shape is unsatisfactory, users can add an example in a specific pose. The challenge for users is to visualize a large pose space. As the number of examples increases, it can be overwhelming to figure out what effect each pose has on the final deformation. The parameters that define the distance between poses may also be unintuitive to set up.

On the basis of PSD, further extensions like [39, 65] are proposed with less demand of the number of example poses. Despite these methods can deal with large-scale deformations well, they cannot provide detailed deformations and require more calculations than the original work of PSD. In these methods, the number of memory increases with the number of examples, therefore they are more popular in animated feature films than in real-time applications.

In addition to sculpting shapes, other example-based approaches use scanned and

photographed data as examples. In the work of [8] , they propose a body deformation calculation method by interpolating shapes using scattered data interpolation from range scans of bodies in a variety of poses. Huang *et al*. [30] present a robust framework that can generate fine detailed as well as large-scale deformations using scanned hands. To further enrich the animation, dynamic skinning effects can be achieved in real-time by learning from physics-based simulation [69]. More recently, Le *et al*. [41] present a method for generating linear blend skinning models by using a set of example poses. The obtained model includes skeletal structure, skinning weights, joint positions, and corresponding bone transformations.

Example-based approaches can improve the deformation effect of geometry-based methods by learning from well-designed examples. However, one major problem of example-based approaches is the requirement of examples. In addition to the fact that the example poses cannot be captured on real humans or characters, creating these poses needs significant effort by artists or requires complex physical simulations of the volumetric version of the mesh. With the development, example-based methods have gradually developed into learning-based methods, which we will discuss in Section 2.4.

2.1.1.3 Physics-based skinning methods

One underlying limitation of geometry-based approaches is their struggle to generate deformations far from realism such as fatty tissues, muscle bulging, and skin contact. These effects require animators to configure the deformation in each keyframe, which is tedious and time-consuming. To this end, physics-based simulation is introduced into the skinning process, in order to enhance the credibility and realism of character animations. The physics-based approaches manage to bring skeleton-driven animation beyond pure kinematics by simulating secondary motions like soft tissues jiggling. The additional effects enrich the visual experience of animations and become essential for creating appealing characters for movie productions.

One most popular techniques for simulating soft bodies in computer animation are the force-based method. In particular, due to simplicity and efficiency, most techniques used to simulate dynamics rely on mass-spring systems. The general idea is to represent the vertices of the mesh as mass points, controlled by Newton's second law of motion,

and represent the edges as elastic links (springs). Therefore, when the length of the elastic link changes, the mesh is deformed. This happens when the relative position of the mass point changes due to external forces. The mass-spring system is based on a local description of the material, where the physical principle of such a system is simple, and the simulator is easy to implement. However, to simulate a specific material, it is important to carefully choose the parameters of the spring, such as stiffness and damping. Although these systems are easy to implement, they still suffer from instability and overshoot at long time steps. In addition, mass-spring systems are usually inaccurate because they strongly rely on topology and are not constructed based on elastic theory.

Earlier studies proposed in [16, 71] are the first to apply mass-spring systems to deformation simulations. In their methods, they use a finite difference scheme to apply Lagrangian equations of motion to simulate elastic characters with regular parameter settings. Here, the physical material properties can be described using only a few parameters that are used to model soft bodies in an accurate manner. Studies in [14,40] use layered representations which consist of a deformable volume for the tissue layer, rigidly attached to a kinematic skeleton. By choosing a volumetric mesh that aligns with the bones, methods are able to meet bone constraints rapidly. McAdams *et al.* [55] present a robust method to simulate a deformed soft tissue shape based on a hexahedral lattice, but it cannot achieve real-time performance. To make the simulation fast, position-based dynamics (PBD) [60] solvers have been proposed and widely applied in the game industry and virtual reality. Different from the force-based method that first solves the force and then performs numerical integration based on the force value, PBD first constructs constraints, then obtains position information through constraint projection, and updates the velocity value accordingly. Despite the effectiveness and rapidity of PBD, its disadvantage is that different constraints sharing the same vertex are alternately projected to different target positions, resulting in jumps between these positions. On the basis of it, Bouaziz *et al.* [12] introduce projective dynamics (PD) for implicit time integration of physical systems which overcomes the artefacts of PBD and converges in fewer iterations. More recently, based on PD, a skinning technique named projective skinning [38] is proposed to produce high-quality skin deformations and can handle local self-collisions between skins.

2.1.2 Learning-Based Deformation

To make the deformation process automatic, learning-based methods are aimed to learn mappings from input parameters to perform deformations using neural networks. Here, we would like to discuss deformation studies according to the type of neural networks: deformations based on the linear-based model or fully-connected layers (also known as multilayer perceptions, MLP), and deformations based on graph neural networks.

2.1.2.1 Linear-based or MLP-based methods

Loper *et al.* [53] present a learned skinned multi-person linear model (SMPL) of human body shape and pose-dependent shape variation. The method is vertex-based and can accurately represent a wide range of body shapes in natural human poses that significantly improves over previous methods [29]. The advantage of SMPL lies in the calculation of pose and shape blend shapes, by simply adding vertex displacements to a template mesh to generate compelling articulated 3D meshes. Specifically, for pose blend shapes, the model is trained to build the relationship between the input pose vector to output pose-dependent blend shapes (displacements). For shape blend shapes, the model is trained to mapping from shape coefficients to shape displacements.

Based on SMPL, Casas *et al.* [15] enrich the human bodies by adding soft-tissue dynamics. Dynamic blend shapes are predicted by inputting the vector which consists of previous blend shapes and poses in multiple frames. More recently, Santesteban *et al.* [67] propose SoftSMPL that also aims at producing soft-tissue dynamics. They present a novel motion descriptor to achieve a better generalization that encompasses the velocity and acceleration of body root, pose descriptor, and the velocity and acceleration of the pose descriptor. It should be noted that, the generalization here means the generalizations to new motions or new shapes (with the same vertices number and same mesh geometry) , but the generalization to different character meshes is still limited.

To make film-quality character rigs able to run in real-time with modest computing platforms such as mobile devices or game consoles, Bailey *et al.* [9] propose a deformation system and successfully implement it on an iPad. Characters such as Shifu from “Kung Fu Panda 3" and Astrid from “How to Train Your Dragon 2" can easily and

rapidly interact with the application. Due to the complexity of the rigs, it needs a large number of models to train for one specific character.

2.1.2.2 Graph-learning based methods

To address the fundamental limitation of generalization in learning-based deformations, the latest research tries to approximate deformation using graph neural networks due to their ability to handle 3D data in non-Euclidian domains. We will introduce types and algorithms of GNNs in detail in Section 2.3. Here, we would like to first discuss their applications to deformation approximations.

For production characters with non-manifold meshes and complicated skeleton structures, Liu *et al.* [51] introduce a deformation method called NeuroSkinning. The method can automatically compute skinning weights for geometry-based deformation for new characters with complicated dressing. Here, there is no limit to the number of vertices or skeletal structures of characters, the trained network can be easily applied to arbitrary charters in game production. The proposal of this research created a precedent for applying graph neural networks to character deformations. However, because the predicted skin weight is fixed and its essence is still a skeleton-based method, it is impossible to achieve additional complicated nonlinear deformations related to poses, *e.g.*, skin slide, and muscle bulges.

To automatically generate character rigs for animations, Xu *et al.* [78] present RigNet to predict skeletons with joint placement and topology as well as surface skinning weights based on the skeleton. A deep modular structure is designed, which contains several modules to complete tasks. The module of predicting the number of joints and the module of skinning weights both utilize graph neural networks. Thanks to these graph-learning-based strategies, the approach can provide a complete solution for character rigs in various scenarios. However, like the drawback of the above NeuroSkinning, it is still unable to generate pose-based realistic deformation effects.

2.2 Related Works of the 2^{nd} Case Deformation

For the second case deformation, the focus of the research is on clothing deformation approximation. In this section, we discuss the existing studies by grouping them into physics-based simulation and learning-based methods.

2.2.1 Physics-based simulation

Physics-based simulation methods use discretizations of classical mechanics to deform cloth by solving an ordinary differential equation. There are three major processes of cloth simulation: computation of internal cloth forces, collision detection, and collision response; and the total simulation cost results from the combined influence of the three processes. Several approaches have been proposed, with differences in the underlying representation, numerical solution methods, collision detection, and constraints. In the work of [32], they propose a method for approximating penalty-based contact forces in yarn-yarn collisions by computing the exact contact response. Despite the high level of realism, the computation time costs too much and cannot be applied to interactive applications. Recently, Cirio *et al.* [20] propose an efficient method to simulate knitted cloth at the yarn level. The simulation results show details with high-resolution while requiring significant runtime computational costs.

To fight with high computational cost, several studies try to use PBD to produce the cloth simulation. For example, Kim *et al.* [36] propose a long-range attachment method that applies unilateral distance constraint between free particles of the cloth to a distant attachment point on the character. This constraint provides an efficient shortcut for enforcing global inextensibility that can be readily implemented into existing game physics methods such as position-based dynamics (PBD). Also based on PBD, instead of constraining the distances on edges, Müller *et al.* [59] present a method to derive sets of positional projections for the deformation modes corresponding to the entries of the Green - St Venant strain tensor. The common point of these methods is: even if the simulation becomes fast to a certain extent, the realism is lacking for real-world applications.

To make the simulation process rapid, the sub-space technique is used in [21].

They learn a low-dimensional representation for both the cloth and the outer surface of the body by constructing two low-dimensional linear subspace models. The learned model is efficient and allows for real-time running. Kavan *et al.* [34] speed up physics-based simulation by adding details to low-resolution simulated meshes. The process starts by pre-computing a pair of coarse and fine training simulations aligned with tracking constraints using harmonic test functions and then training the upsampling operators for cloth simulation. A similar idea of adding details to low-resolution cloth meshes, research in [81] defines an algorithm to synthesize cloth wrinkles based on the deformation of low-resolution cloth and a set of example poses. Gillette *et al.* [26] introduce a real-time method for dynamic cloth wrinkle simulation. The proposed method is suitable for interactive wrinkling of cloth and fabrics for video games and hardware. By simplify physical models, Rohmer *et al.* [66] present an automated-post-processing step for generalizing cloth wrinkles based on coarse simulation. The shapes of wrinkles look quite believable, but they cannot ensure that the pattern has precise equidistant lines, nor that the surface created is developable.

2.2.2 Learning-based methods

Inspired by the success of deep learning and its application to skin deformations. More and more studies propose to learn clothing deformation as a function of the underlying body and pose. In this subsection, we will respectively introduce relevant works using MLPs and graph-based models.

2.2.2.1 MLP-based methods

To combat the high computational cost of physics-based simulation while realizing clothing nonlinear behaviors, Santesteban *et al.* [68] present a two-level strategy to learn deformations of garments dressed by SMPL bodies. Two-level denotes the process of generating the clothing deformation dependent on the target body's shape by using MLPs, and the process of generating detailed wrinkles dependent on both the shape and the pose by using Gate Recurrent Unit (GRU). Following this strategy, the method is able to capture nonlinear wrinkle effects. One drawback is that the method requires independent training for different garment sizes or mesh geometries.

Also in order to estimate the cloth deformations with fine details, research in TailorNet [63] uses multiple MLPs to realize the task. They decompose the deformation into a high-frequency component and a low-frequency component. While the low-frequency component is approximated from pose, shape, style parameters with one MLP model, the high-frequency component is approximated with several MLP models and their mixture. TailorNet delivers garments that retain the wrinkles from the physics-based simulations it is learned from while running faster.

To model how people wear the same garments in different sizes, Tiwari *et al.* [72] propose a SizerNet to approximate the wearing effect of a garment in different sizes. Because the dataset only consists of A-pose garments and garments, the proposed method cannot generate a variety of deformations in different poses.

Although the above-mentioned studies have achieved success in the automatic clothing deformation approximation, a common issue of these methods is that the inability of generalization to other objects with different mesh topologies. The reason for this problem is that when using this type of ordinary network, input or output values are just flattened into vectors, which enforces meshes to have the same topology with the fixed number of vertices.

However, in real applications such as games, virtual try-on, etc., the shape and topology of character objects are often various and have a different number of vertices. Therefore, the applicability of this learning method is still insufficient.

2.2.2.2 Graph-learning based methods

More recent research in [18] introduces a graph convolution network for cloth and body skin deformation approximation. They improve the graph convolution operator and define the pooling and unpooling operations on the mesh. The proposed framework can be applied to several problems including cloth upsampling, pose-to-cloth regressions, PCA coefficients to cloth deformation, and joint angles to hand-skin deformation. There also exists some limitations of this work. First, all proposed solutions, such as convolution, pooling, and unpooling, are based on triangle mesh, and they cannot be applied to other mesh topologies, like quadrilateral meshes. Furthermore, the prediction accuracy is substantially dependent on a large training set, because tasks to be regression are

extremely complex and the generalization of the network is still insufficient.

Focusing on fast cloth deformation, Vidaurre1 *et al.* [75] present a fully convolutional graph neural network (FCGNN) to predict deformations with fine-scale details. Before the learning, they initially build a parametric space for garment design that is capable of representing a large number of garment types. Then, they train two GNNs for predicting the coarse 3D draping of a garment onto the mean body shape and the final deformation refinement. These two networks have a similar network structure with a different number of layers. The proposed pipeline is able to generalize to unseen mesh topologies, garment parameters, and body shapes. However, it still suffers the weakness that pose-dependent parameters are not considered in the approach that deformations are only approximated in the rest-pose. More elaborate effects related to posture are limited to achieve.

2.3 Graph Neural Networks

Convolutional neural networks (CNNs) have achieved great success in various challenging tasks, especially for dealing with 2D images and regular 3D grids. Alternatively, there are many irregular data structures that can be represented as graphs such as point clouds, social networks, and meshes. These complicated graph data cannot be directly processed with the traditional convolution. In recent years, many attempts have been made to explore the extension of the CNNs to graph neural networks in order to reasoning about irregular 3D data.

There are many variants of GNNs which use different aggregators to gather the neighbor's information and then forward it to the next state. In the up-to-date survey on GNNs [80], GNNs are categorized into several groups based on their propagation steps: convolution, gate mechanism, skip connection, and attention mechanism.

Existing works with convolution operations on graph could be divide into spectral approaches [13, 27, 28, 47] and spatial approaches [7, 25, 56]. Specifically, for spectral networks, the convolution is defined in the Fourier domain by computing the eigendecomposition of the graph Laplacian. In all of the spectral approaches, the learned filters depend on the Laplacian eigenbasis, which depends on the graph structure, that is, a

model trained on a specific structure could not be directly applied to a graph with a different structure. For spatial networks, they directly define convolution operations on spatially close neighboring nodes. And the major challenge of this type's network is how to define the convolution operation with various sized neighborhoods and maintaining the local invariance of CNNs. Several works address this challenge by using different weight matrics for nodes with different degrees [17], by transforming to diffusion convolutional representations, and by using univariate functions.

Some studies try to use gate mechanisms in the propagation step, such as gated graph neural network (GGNN) [50] which uses the gate recurrent units in the propagation step, unrolls the recurrence for a fixed number of steps T and uses backpropagation through time in order to compute gradients. Based on the basic LSTMs architecture, the work in [70] proposes two extensions for graph data.

Training deep networks to benefit from their advantages is a common operation in CNNs. There are also attempts to unroll or stack graph neural network layers aiming to achieve better results. Some experiments [37] show that directly use deeper models could not improve the results and even perform worse. Rahimi *et al.* [64] present a Highway GCN that uses layer-wise gates similar to highway networks. By following this strategy, the performances peaks at 4 layers for the specific problem.

The attention mechanism has been successfully used in many sequence-based tasks, such as machine translation, machine reading, etc. One of the benefits of attention mechanisms is that they allow the processing of variable-sized inputs, focusing on the most relevant parts of the input to make decisions. Inspired by the attention mechanism, a graph attention network (GAT) [74] is proposed to compute the hidden representation of each node by attending over its neighbors, following a self-attention strategy. This network has several advantages: first, the computation of node and neighbors is parallel, so the efficiency becomes high. Second, by assigning arbitrary weights to neighbors, it can be applied to graph nodes of different degrees. Furthermore, it can be easily applied to inductive learning problems. Mesh deformation method in [51] utilizes GAT for skinning weight prediction. In our work, due to its efficient and easy implementation on arbitrary graphs, we extend and improve the original GAT by designing self-reinforced stream, densely connection, and applying pooling and

unpooling operations for producing nonlinear deformation effects.

Graph pooling plays an important role, as it can avoid overfitting and improve the generalization ability of networks. Because of the complicated and irregular characteristics of graph nodes, few graph pooling studies have been done. Ying *et al.* [79] propose DiffPool which learns the assignment matrix of clusters based on node features. The number of clusters needs to be decided beforehand. Gao and Ji [24] introduced TopKPool in Graph U-Net. The strategy of their pooling is to simply keep nodes with top-k scores and drop all other nodes. One drawback of this approach is poor robustness where small local changes will affect the whole pooling results. Recently, Lee *et al.* [43] present SAGPool that uses a self-attention mechanism to calculate node scores and then mask all but the top nodes. Different from their masking nodes, inspired by the method of computing attention coefficients mentioned in [74], we conduct merging neighboring nodes according to maximum attention coefficients, which has the effect for contracting edges so that both nodes features and topologies will be considered for graph processing and conveyed to deeper layers.

2.4 Summary

Finally, we take an overview of the main character deformation techniques.

For the 1^{st} case deformation, traditional methods are hard to balance the deformation accuracy and speed. Geometry-based solutions trade realism for performance, while physics-based solutions are the opposite.

To make the deformation process automatic, learning-based studies are proposed to compute deformation as a function of designed parameters. Pioneer studies [9, 53] use linear-based models or MLPs to achieve the task, but their trained models are only applicable to the specific object and cannot be generalized to new characters with different topologies or different numbers of nodes. The adoption of graph learning solves this generalization problem. However, existing research [51, 78] is only able to generate fixed skinning weights without nonlinear deformation effects in each pose step.

For the 2^{nd} case deformation, physics-based simulation approaches can bring a perfect visual effect with fine-scale fabric wrinkles. However, they are also computationally

expensive with high-end machines and usually avoided in real-time applications.

The situation of learning-based deformation for garments is about the same as the learning-based deformation for the 1^{st} case. Methods use MLPs can achieve good non-linear effects, but they require a large number of models [63] and lack the generalization ability [63, 68] to approximate the deformation for new garments. For graph-learning-based methods, due to the complexity of wrinkles deformation, existing research [75] only guarantees the deformation in one posture (A-pose) and does not have the ability to predict folds in different postures.

To conclude the learning-based deformation methods of both two cases, we list the limitations that exist in the previous learning-based work:

- Generalization (able or not). Most learning-based solutions (linear models or MLPs) are trained for a specific object or mesh topology.
- Generalization (strong or weak). Although some latest solutions (GNNs) are able to generalize to other characters or topologies, approximation relies on huge amounts of training samples (sometimes over-fitting) and results tend to be over-smooth without rich nonlinear details.
- Pose-depend effects. Recent graph-learning-based based approaches only allow for predicting fixed skinning weights or deformations in one pose.

Therefore, our work aims at addressing the above-mentioned limitations using novel graph neural networks and feature processing strategies in order to automatically generate high-quality deformations for characters with an arbitrary number of vertices in various poses.

The development of GNNs has been discussed in Section 2.3. To achieve our deformation goals, we also improve the graph convolution and design graph neural networks that can better perform the task of predicting deformation for animated characters.

Chapter 3

Deformation Approximation - 1^{st} Case

3.1 Context

3.1.1 Background of the 1^{st} case deformation

In the field of games, virtual reality, and films, the demand for diverse, high-quality, animation-ready characters is growing rapidly. For the animated characters in this situation, we have already introduced their mesh features in Section 1.2.1 (Figure 1.2).

During the process of animating this type's character, rigging is a crucial task that involves defining motion, control, and deformation systems. In principle, the first two processes are relatively easy and can be conducted automatically if using a standard skeleton structure with a fixed number of joints for the humanoid character. By contrast, the process of the developing deformation system is always labor-intensive and this has been receiving considerable attention in recent years. Traditional methods using geometry-based skinning and physic-based skinning usually require a struggle between sacrificing realism and sacrificing real-time performance. In addition, this deformation process also needs a large amount of back and forth manual modification of skinning weights and parameters setting by highly skilled animators.

To make the deformation process automatic, more recently, a number of learning-based methods address this topic by using linear-based model [53], MLPs [9] and graph neural networks [51, 78]. None of them has satisfactorily achieved the goal of producing high-quality nonlinear effects in each pose step and simultaneously allows the trained model to be generalized to characters with any number of vertices. In the next

subsection, we will discuss in detail the problems of these learning-based deformation methods.

3.1.2 Unsolved problems in state-of-art studies

State-of-art learning-based methods have been proposed and demonstrated that it is possible to learn an efficient model for deformation approximation. From the perspective of model type, most of these methods leverage linear-based model [53] or MLPs [9] that take shape and/or pose parameters (*e.g.*, flattened joint transformation matrix or rotation angle) as input, and output the predicted deformations. Despite the realism of the result, a common underlying limitation of existing learning-based approaches is the dramatic inability to generalize to new character meshes with different topologies and the different number of vertices. Furthermore, even if the training is only for one specific object, the methods usually require a large number of models to ensure high-quality predictions for the deformed object.

The reason for such limitations can be concluded as follows: first, MLP architectures (or linear-based models), which are known to be easy use in any domain, are adopted for deformation approximation. This type of architecture constrains the size of the input and output vector to a fixed number, which enforces input parameters and output meshes to have always the same dimension. Second, practical character meshes have complicated varying shapes, topologies, and the number of vertices. It is unreasonable to directly apply MLPs or common CNNs to process these 3D mesh data because they ignore spatial information and flatten vectors as input, which will lead to the loss of important local or neighboring information. Third, the architecture of MLPs requires a large number of parameters, since all input nodes are densely connected to each other.

To this end, in order to address the generalization problem and account for non-manifold meshes with various topologies, the latest studies [51, 78] introduce the graph into the deformation approximation since graphs are capable of representing spatial information of mesh vertices (nodes) and edges. Although these methods allow the application of the trained model to predict deformations for new character meshes with different mesh topologies and the number of nodes, they still only propose to predict fixed weights that do not change with postures. Also, because of the complicated of

graph data and vanilla GNNs, sometimes the prediction accuracy is not ideal and the result is unrealistic.

In summary, using GNN is feasible to estimate deformations for various characters, but generating pose-dependent deformations is still blank in the past research due to the difficulty of this task. Specifically, the difficulty of this task lies in two aspects: processing graph features of a large amount of personalized characters and poses, and designing effective graph neural networks to map the features to nonlinear offsets in each pose step. More specifically, from these two aspects, we further expand the challenges in four terms as listed in Section 1.2.1: ***case1-chall.1-4***.

3.1.3 Our proposal outline

To automatically approximate deformations for new characters based on existing well-skinned meshes and address all challenges mentioned in the last subsection, we propose a two-step outline depicted in Figure 3.1. We assume that character deformation can be regarded as a combination of two parts: the coarse deformation, and the nonlinear refinement. The part of coarse deformations on the one hand is simple and therefore can be directly computed by using linear-based skinning, although not precisely. Based on a coarse deformed mesh $M_{\text{coarse}} \in \mathbb{R}^{3\times N}$ with N vertices, we first extract its mesh graph $\mathcal{G}$, and then learn the function to produce corresponding nonlinear mesh shape corrections. The final deformed mesh $M \in \mathbb{R}^{3\times N}$ can be expressed as:

$$M = M_{\text{coarse}} + W(\mathcal{G}; \Psi), \tag{3.1}$$

where $W(\cdot)$ is the refinement through the graph-learning-based nonlinear regressor, that takes graph $\mathcal{G}$ as input and calculates shape offsets for each vertex by learning a set of parameters Ψ.

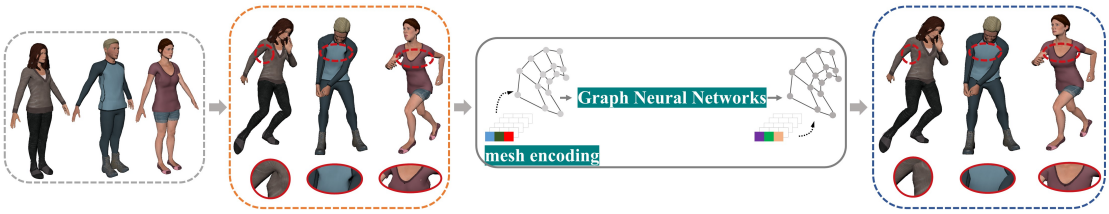


Figure 3.1: Outline of deformation approximation - 1^{st} case.

Concretely, in the process of achieving $W(\mathcal{G};\Psi)$, to account for non-manifold meshes with various topologies, we encode meshes by introducing graphs that are capable represent spatial information of mesh vertices and edges. Then, to deal with graphs, we propose graph neural networks to perform node regression of graph-structured data. In the next sections, we will describe how to encode meshes from various characters in arbitrary poses, and how to design graph neural networks to build the relationship between input graphs and output deformation refinements.

We successively proposed and published two works, *i.e.*, DenseGATs [48] and MultiResGNet [49], to solve the above two technical problems (mesh encoding and designing graph neural networks). DenseGATs is based on graph neural networks carefully designed to avoid typical problems such as gradient vanishing. MultiResGNet further improves DenseGATs from two perspectives: mesh encoding based on relative positions and introduction of the global branch that captures lower-resolution, *i.e.*, overall structural, information. The relationship between DenseGATs and MultiResGNet is shown in Figure 3.2.

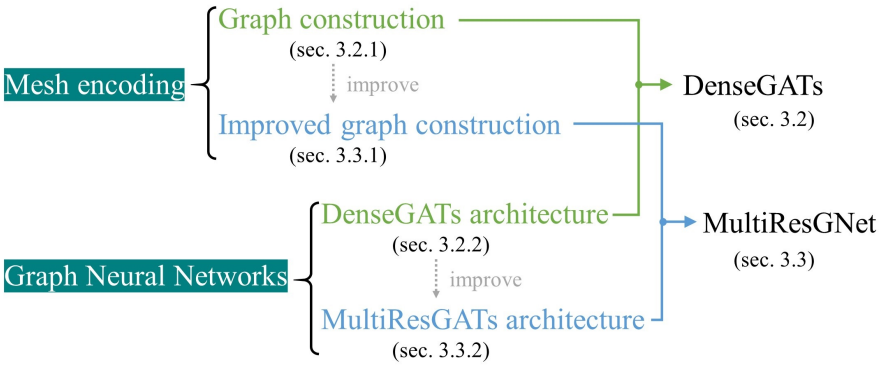


Figure 3.2: Proposed deformation approximation strategies for 1^{st} case.

3.2 Deformation with DenseGATs

3.2.1 Mesh encoding

Having the linear-deformed mesh M_{coarse} directly computed, we need to construct a parametric space that is capable of representing different mesh topologies and varying poses. Here, we consider the input of our framework to be a mesh graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{U})$

which stores features of vertices and edges. Here, $\mathcal{V} = \{1, ..., N\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denote the set of vertices (nodes) and edges respectively. $\mathbf{U} \in [0,1]^{N \times N}$ is the adjacency matrix where $\mathbf{u}(i,j) \in [0,1]$ indicates whether there is an edge between nodes i and j, $(i,j) \in \mathcal{E}$. For the node $i \in \mathcal{V}$, the set of neighboring nodes is represented by $\mathcal{N}(i)$.

For a node $v_i \in \mathcal{V}$, the attribute vector is defined as $v_i = [p_i^T, n_i^T, x_i^T]$, where $p_i \in \mathbb{R}^3$ is the vertex position, $n_i \in \mathbb{R}^3$ is the normal of the vertex. $x_i = [x_{i,1}, ..., x_{i,s}, ..., x_{i,S}] \in \mathbb{R}^S$, where $x_{i,s}$ refers to the volumetric geodesic distance [23], *i.e.*, the shortest distance from vertex v_i to joint s passing though the interior voxels. Here, all characters in our dataset have the same humanoid standard skeleton, where the joint number S is 65.

Thus, the total dimension of the node feature v_i is $6 + S$. v_i contains both mesh skinning appearance attributes which indicate the features of the vertex itself, connectivity between other vertices, and the distance attributes from the vertex to joints which implies a positional relationship between the vertex and control skeleton.

3.2.2 Deformation refinement through DenseGATs

3.2.2.1 Issues of existing graph neural networks

Graph neural networks are capable of dealing with non-Euclidean data like mesh. For our character deformation prediction, one challenge is to use prior mesh graph information to inductively generalize the graph features of mesh that have never been seen before. [74] introduces an graph-attention-based architecture to compute the hidden representation of each node by aggregating neighborhood features with different weights, without the need to know the entire graph structure upfront. This GAT model has successfully achieved or matched state-of-the-art performance across well-established node classification benchmarks, and well completed the skinning weight prediction in [51].

However, different from previous tasks such as classification and skinning weight prediction (the sum of one joint's weights is equal to one), our goal is to directly predict deformation refinements, *i.e.*, vertex deviations for diverse characters, which tend to be extremely irregular and complicated. Therefore, it is impossible to directly use the ready-made graph neural networks such as original GAT to complete the prediction

of accurate nonlinear deformation. Considering the large number of mesh vertices and complex features, it is necessary to design effective network architecture which is capable of mapping mesh graphs to corrective displacement of per mesh vertex for more complicated nonlinear effects.

3.2.2.2 GAT block

We propose a GAT block architecture shown in Figure 3.3. The GAT block consists of a graph-attention-based aggregation stream and a self-reinforced stream. The aggregation stream is used to compute the hidden representations of each node in graphs, by applying its adjacent features using a graph attention network. In addition to the neighboring nodes aggregated by graph convolution, strengthening single node features is also necessary to accurately convey information to deeper layers. Therefore, we design a self-reinforced stream to transform the node's own features and concatenate them with the output of the graph convolution. These features, along with the aggregation stream obtained features, are then fed into next GAT block.

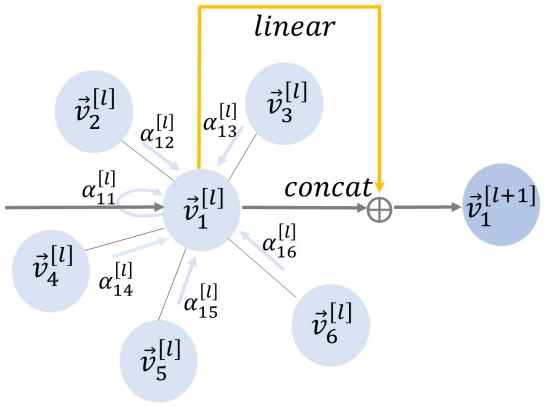


Figure 3.3: Illustration of inside of GAT block by the node 1 (feature vector is $\vec{v}_1^{[l]}$) on its neighboring nodes. The blue color indicates the process of graph-attention-based aggregation stream gathers features from neighboring nodes with different weights [74]. The yellow color indicates the process of self-reinforced stream that linearly transforms the features of the node 1. These two streams are then concatenated to produce new feature representation.

After nonlinear transformation, the node features are input into GAT blocks. Here, we use $\boldsymbol{v}^{[l]} = \left\{\vec{v}_1^{[l]}, ..., \vec{v}_i^{[l]}, ..., \vec{v}_N^{[l]}\right\}$, $\vec{v}_i^{[l]} \in \mathbb{R}^{d^{[l]}}$ and $\boldsymbol{v}^{[l+1]} = \left\{\vec{v}_1^{[l+1]}, ..., \vec{v}_i^{[l+1]}, ..., \vec{v}_N^{[l+1]}\right\}$, $\vec{v}_i^{[l+1]} \in \mathbb{R}^{d^{[l+1]}}$ to represent an input and output of one GAT block, where $d^{[l]}$ and $d^{[l+1]}$

indicate the feature dimensions after previous l and $l+1$ layers' feature transformation. The overall process of feature transformation inside one GAT block can be expressed as $\vec{v}_i^{[l+1]} = f_{\text{GATB}}(\vec{v}_i^{[l]})$.

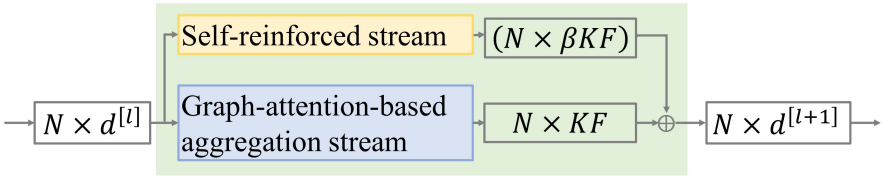


Figure 3.4: The input feature dimension into one GAT block is $d^{[l]}$, the output dimension of aggregation stream is F, the number of multi-head attention is K, and the hyper-parameters β. So the output feature dimension is $d^{[l+1]} = KF + \beta KF$.

Specifically, for the graph-attention-based aggregation streams, given a set of node features, they firstly need to be pre-processed so that they can be applied to each node by linear transformation to obtain higher dimensional expressions. The transformed features can be expressed as:

$$\vec{z}_i^{[l]} = \boldsymbol{W}\vec{v}_i^{[l]} \tag{3.2}$$

where the trainable parameter of the transformation $\boldsymbol{W}$ is a weight matrix $\boldsymbol{W} \in \mathbb{R}^{F \times d^{[l]}}$. F denotes the transformed feature dimension of a node in the aggregation stream.

Although aggregation streams could integrate features from first-order neighboring nodes, the features of the node itself are diminished in this aggregation. We designed a self-reinforced stream which is a linear transformation process for $\vec{v}_i^{[l]}$:

$$\vec{c}_i^{[l]} = \boldsymbol{H}\vec{v}_i^{[l]} \tag{3.3}$$

where $\boldsymbol{H}$ is a trainable weight matrix $\boldsymbol{H} \in \mathbb{R}^{\beta F \times d^{[l]}}$. Here, β is the hyper-parameter determining the importance of the self-reinforce stream.

For aggregation steam, as stated in [74], a shared, masked attention mechanism is performed that only computes attention coefficients with the neighboring nodes $\mathcal{N}(i)$ of a node which ensures that structural information is not lost. Furthermore, to make the results of attention coefficients across different neighborhoods comparable, they are normalized using the softmax function to obtain attention weights:

$$\alpha_{ij}^{[l]} = \frac{\exp(\text{LeakyReLU}(\vec{a}^{[l]^T}(\vec{z}_i^{[l]} \| \vec{z}_j^{[l]})))}{\sum_{k \in \mathcal{N}(i)} \exp(\text{LeakyReLU}(\vec{a}^{[l]^T}(\vec{z}_i^{[l]} \| \vec{z}_k^{[l]})))} \tag{3.4}$$

where $\alpha_{ij}^{[l]} \in \mathbb{R}$. $\vec{a}^{[l]} \in \mathbb{R}^{2F}$ indicates the weight vector and $(\cdot)^T$ represents its transposition. $\|$ is the concatenation operation that features $\vec{z}_i^{[l]}$ and features from neighboring nodes are firstly concatenated, and then this concatenation embedding result with a learnable weight vector $\vec{a}^{[l]^T}$ are executed by dot product. During the process of calculating attention coefficient, to enhance nonlinear expression, LeakyReLU is applied as the activation function.

The obtained attention weights are then linearly combined with their corresponding features to yield the output features of the aggregation steam. To improve the stability and representation ability of the model, multi-head attention mechanism K introduced in [73] is also adopted to execute transformation K times with different training parameters of aggregation operations. In addition to the aggregation steam, to enhance the expression of self-features, our designed self-reinforced steam is concatenated with the features from the graph-attention-based aggregation steam (as shown in Figure 3.3) to form the final output features of one GAT block:

$$\vec{v}_i^{[l+1]} = f_{\text{GATB}}(\vec{v}_i^{[l]}) = \sigma\Big(\|_{k=1}^{k} \underbrace{\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{[l]k} \boldsymbol{W}^k \vec{v}_j^{[l]}}_{\text{aggregation}} \| \underbrace{\vec{c}_i^{[l]}}_{\text{self-reinforced}} \Big), \tag{3.5}$$

where σ represents the nonlinear transformation tanhshrink. Since the output type is displacement which could be positive and negative, tanhshrink is able to retain negative information. Meanwhile, it can alleviate gradient vanishing problem (tanh cannot) and therefore is selected in our network. $\alpha_{ij}^{[l]k}$ and $\boldsymbol{W}^k$ respectively indicate the attention weights and input linear transformation's weight matrix computed by the k_{th} attention mechanism in the l_{th} layer. For one GAT block, the dimension of output features $d^{[l+1]}$ equals the concatenation dimension of aggregation stream and self-reinforced stream $KF + \beta KF$ (as depicted in Figure 3.4).

3.2.2.3 Dense module

Due to the complexity and significant amount of features in our mesh graph, there is a need to apply very deep networks to benefit from their advantages. However, for training deep graph convolutional networks, the problem of gradient vanishing will become more serious as the number of network layer increases. To address this problem, DenseGCNs [46] borrows the concept from DenseNet [45] by introducing dense connections to deep GCN frameworks and successfully be applied on segmentation task with specific point cloud graph structure. Inspired by it, we densely connect our GAT blocks as a dense module for further training deep layers of GAT blocks to better help process large volumes of complicated mesh data. As opposed to DeepGCNs, our dense module allows for handling unseen graph structures and efficiently computing with different importance to nodes of a same neighborhood.

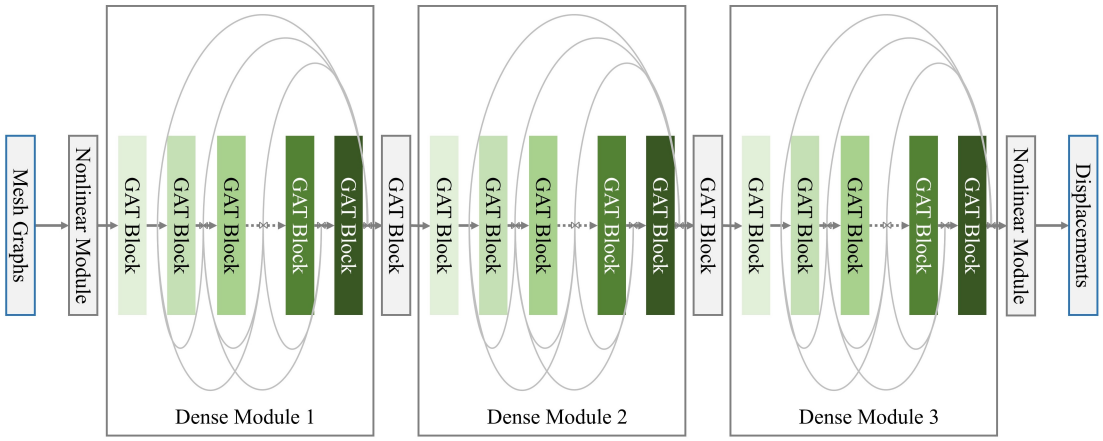


Figure 3.5: The structure of "DenseGATs". The three boxes denote our proposed "dense modules". Each dense module comprises six densely connected GAT blocks.

The overall of dense connection pattern is shown in Figure 3.5. For simplicity, we use a single function f_{GATB} to represent the transformation process of one GAT block in the rest of this paper. In a dense module, among several GAT blocks, direct connections are introduced from one block to all subsequent blocks. Suppose, there are m GAT blocks in one dense module, the layer of the first GAT block is l, thus, the $(l+m)_{th}$ layer receives features of all layers starting from the l_{th} layer. The propagation can be defined as:

$$\begin{aligned}\vec{v}_i^{[l+m]} &= f_{\text{GATB}}(\vec{v}_i^{[l+m-1]}, \theta^{[l+m-1]}) \| \vec{v}_i^{[l+m-1]} \\ &= f_{\text{GATB}}(\vec{v}_i^{[l+m-1]}, \theta^{[l+m-1]}) \| ... \| f_{\text{GATB}}(\vec{v}_i^{[l]}, \theta^{[l]}) \| \vec{v}_i^{[l]}\end{aligned} \tag{3.6}$$

where $\theta^{[\cdot]}$ represents all parameters of the transformation function f_{GATB} in different layers. The $\vec{v}_i^{[l+m]}$ is the result of fusing all the intermediate GAT block layer outputs. Since we connect GAT blocks in a dense pattern, we refer to this architecture as "dense module", and the whole network refers to "DenseGATs". Note that, because of the dense connection, for each GAT block (except the first block) in a dense module, the output features will consist of all preceding blocks' features, not only one block feature $(KF + \beta KF)$ for each node.

Between two adjacent dense modules, we adopt one GAT block as the transition. This operation plays the role of information integration.

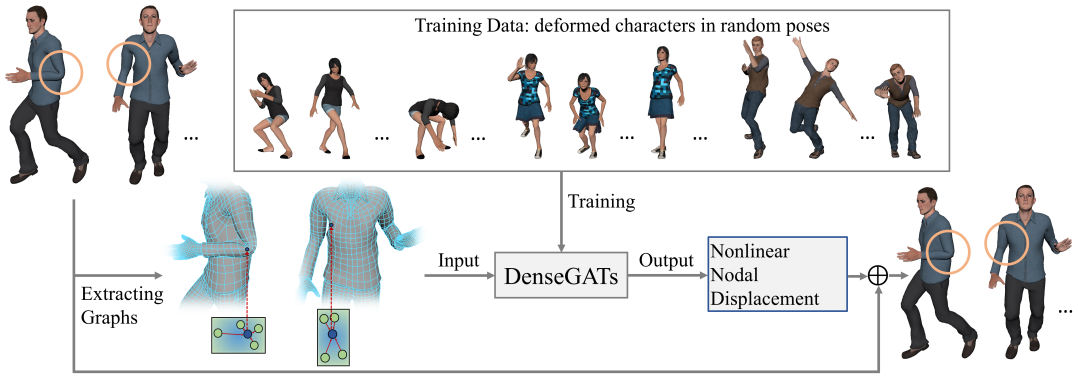


Figure 3.6: The framework of DenseGATs.

In summary, the overall framework of our DenseGATs method is shown in Figure 3.6, where the detail of DenseGATs architecture is depicted in Figure 3.5. We found that our data can reach the ideal approximation accuracy under this number of GAT blocks and modules. At runtime, our deformation model works by computing deviations for correcting nodal linear deformations to nonlinear ones. By learning skinning features in the train set, our method yields more accurate deformations for new character meshes, thereby significantly reducing the time and efforts taken for performing the skinning process.

3.2.3 Evaluation of DenseGATs

3.2.3.1 Dataset

To evaluate our proposed methods, we created dataset for training and testing.

Figure 3.7: Example characters with index 1-5 in our dataset of the 1st case. These characters have completely different representative dresses (tops, bottoms, shoes) and the same skeleton structure.

Table 3.1: Statistics for our example characters (as shown in Figure 3.7).

Character Index	Vertices	Heights	Joints
1	9035	177 cm	65
2	10220	166 cm	65
3	9303	167 cm	65
4	9005	181 cm	65
5	10320	182 cm	65

There are about 150 character models with different customizations in DenseGATs' dataset created with Adobe Fuse CC, and all of them are embedded with corresponding skeleton structures properly. To produce basic skinning data for training, the linear blend skinning method is used with eight maximum of joints influencing each vertex for rough deformation. For the ground truth data, to reduce the tedious rigging workload, the characters are firstly auto-skinned by Mixamo [3]. Then, for the inaccurate deformations with obvious artefacts in specific poses (e.g., elbow bending, twisting, etc.), manual refinements [44] were done by animators for more realistic effects. Figure 3.7 shows several test example characters with the rest pose in our dataset. These example characters with the statistics are shown in Table. 3.1 including the number of vertices, heights, and the number of joints. With all characters in our dataset, in order to save

GPU memories and training time, several character bodies without head parts are taken into account. All characters are set with a standard skeleton structure, for a total of 65 joints for the entire skeleton. To accurately predict the mesh deformation generated under any poses, we use two strategies to create the training examples. Firstly, in order to cover the range of many possible poses, we manually set each skeletal joint in a reasonable range for the rotation and scaling, and then generate poses by independently and randomly sampling in the range of all joints. This sampling method ensures that the full range of motion for each joint is contained in the training set. Furthermore, we animate our character models with motions that appear frequently in animation such as walking, jumping, running, and dancing provided by TurboSquid [6]. In total, there are about 8500 poses generated with these methods for our characters, and several examples are shown in Figure 3.8. Because the number of pose samples in the dataset is large, and it also contains many possible random poses and common motions, our approximator can accurately learn from the data and can robustly predict deformations for new poses. Then, due to a large amount of computation during the training, we randomly select one thousand consecutive vertices from each character mesh at each training epoch to ensure stable training while saving memories. To verify the effectiveness of our proposed network, we use 125 training character models and five validation character models of the dataset with about 8500 poses. The remaining 20 character models with random poses are used for testing the network.

Figure 3.8: Example poses of our character. These include typical poses (walking, running, jumping, and dancing) and random poses.

3.2.3.2 Implementation details

Based on the amount of training data available, we explored a set of parameters which can ensure the satisfied result while balancing the size of the network.

As shown in Figure 3.5, we first feed the input graph features into a nonlinear transformation module which involves two fully connected hidden layers with 32 hidden units and followed by tanh activation. Then the transformed features are fed into dense modules. The network we propose involves three dense modules, each of which consists of six GAT blocks. We found this setting is sufficient to approximate satisfactory results without excessively increasing training time. To ensure maximum information flow between GAT blocks in one module, all blocks are densely connected. For the internal structure of the GAT block, we adopt a graph-attention-based aggregation stream with the hidden features number of 8 and the multi-head number of 8. To increase the effectiveness of the interpenetration, in the self-reinforced stream, input features are processed by a linear layer with the feature size of 0.125 times the aggregation stream output feature size ($\beta = 0.125$). The final layer of one GAT block is applied a tanhshrink activation function. We refer to GAT blocks between each dense module as transition blocks which have the same parameters with GAT blocks inside the dense module. After all three dense modules, the resulting features are taken as being input to two fully connected hidden layers with 512 and 128 hidden units and followed by tanh activation. All the outputs of layers in the network are applied with 1D batch normalization.

We trained our model with nVIDIA GeForce RTX2080Ti GPU and set batch size of 8. During the training process, we trained the model using the Adam optimization method, with the initial learning rate of 1e-3. We further set the reducing learning rate with decay factor of 0.75 when the loss has stopped decreasing beyond eight epochs. The lower boundary on the learning rate of all param groups is set as 5e-7. For the loss function of our network, we choose Mean Squared Error (MSE) to minimize the distance between predicted displacement distributions and the ground truth displacement distribution.

3.2.3.3 Results of DenseGATs

To quantitatively evaluate the performance of our network, we measured the average distance error, max distance error, and minimum distance error of our predicted deformations. For each character model in the test set, we animated them using walking motions and computed the prediction vertex errors across all frames. Table. 3.2 shows the prediction errors for the test models. It can be observed that our proposed method can effectively predict the deformation for new characters with very low errors. With our network, the prediction time for one character (*e.g.*, No.2 character) in each frame is about 31ms. In Figure 3.9, we further plot the average error of the No.2 test character to visualize error distribution over the mesh for all frames of dancing and running animations. From these two plots, most of deviation distances are around 0.1cm within the allowable range of accuracy. And the number of vertices decreases exponentially with increasing distance errors. We found that the performance of predicting deformation with running animation is better than with dancing animation, where there exist several vertices in the dancing motion that have the estimated errors of more than 0.5cm. This is because the dancing animation includes some extreme poses which are beyond the range of network it was trained on.

Table 3.2: Prediction errors in "cm" between the ground truth and predicted vertices.

Character Index	mean error	max error	min error
1	0.0662	1.1639	0.0003
2	0.1045	1.5832	0.0006
3	0.0947	1.4909	0.0003
4	0.0820	1.4201	0.0003
5	0.1289	1.6966	0.0005

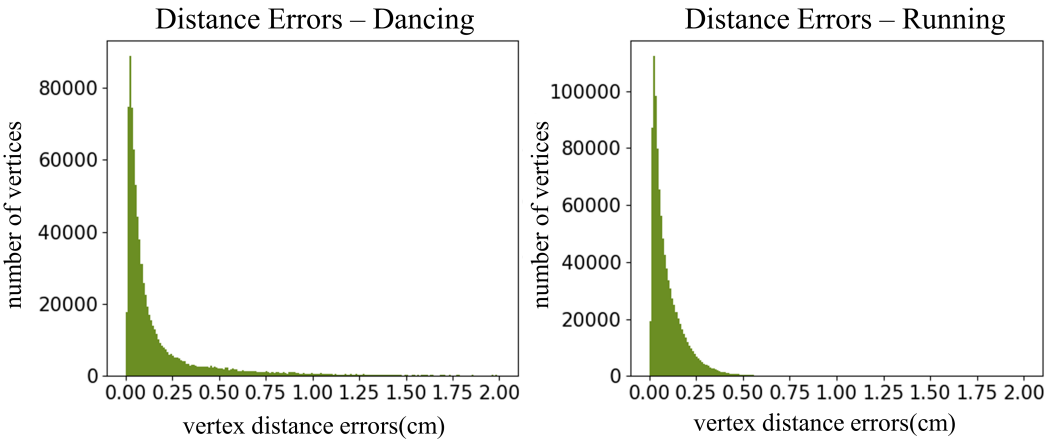


Figure 3.9: Distribution of vertex errors. The left is for dancing animation, and the right is for running animation.

To verify that our trained network is able to predict satisfactory and natural deformation results, we animate the test models with weightlifter motion as shown in Figure 3.10. We focus on some contact regions where mesh deformation volume always cannot be maintained and is highly prone to artefacts. The results approximated by our method are hardly distinguishable from the ground truth, which demonstrates that our network is able to correct the rough linear deformation to the more complex nonlinear one.

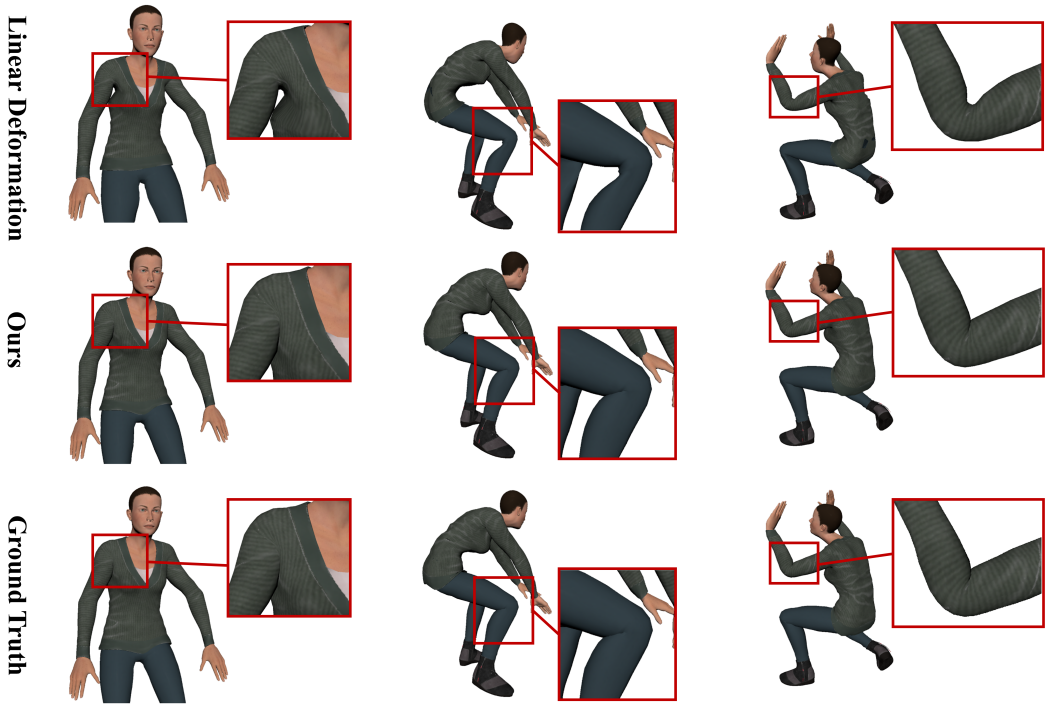


Figure 3.10: Close-up of our test character's contact regions in three frames of weightlifter motions. Based on rough linear deformation, our method corrects linear deformations to nonlinear ones by per vertex displacement correction.

We provide further intuitive deformation results with multiple poses in different animations using error color map to qualitatively evaluate our method in Figure 3.11. In Figure 3.12, we also provide a side-by-side comparison of the ground truth, the approximated deformation, and the error color map of a test character with a walking posture. With our proposed network, the approximated deformation results are visually similar to the ground truth. It can be noted that the largest deformation errors are mainly in the trousers (especially the top of trousers) of the character. The deformations in these regions are always influenced by multiple bones. In addition, the trousers worn by the test character are quite loose, whose mesh features are significantly different from those in our training set. And these reasons result in some inaccurate approximation.

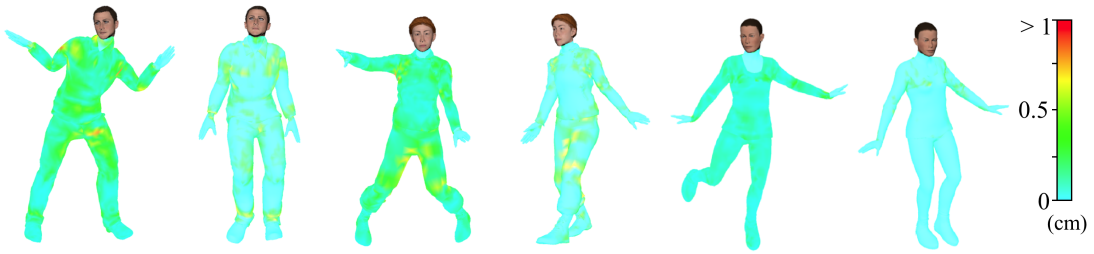


Figure 3.11: Result of deformation approximation among different characters with different poses. The vertices of the predicted mesh are colored to indicate per-vertex distance prediction errors.

Figure 3.12: Comparison of ground truth (left), prediction results via DenseGATs (center), and color map (right) indicating per-vertex distance error.

We compare our method with classical deformation methods LBS and DQS. As shown in Figure 3.13, the deformation with LBS method shows the obvious artefact of volume loss when twisting the elbow. Unnatural deformations are always found in the areas near the shoulder, elbow, waist and the bottom of the pants. It is hard to assign reliable weights with direct LBS method especially for those areas influenced by multiple bones. For DQS result, bulging artifacts near the joints are very obvious which still need artistic corrections. Our method, in contrast, given the inaccurate LBS deformation, can accurately approximate most displacements of vertices and correct them to nonlinear ones with no noticeable errors.

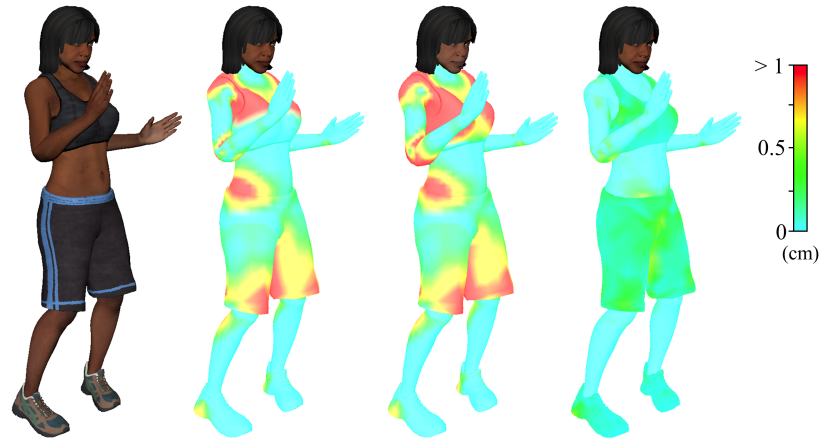


Figure 3.13: Comparison of ground truth, color map of LBS, DQS, and our prediction.

Table 3.3: Evaluation of predicted distance errors (cm) using different network architectures.

Network	mean error	max error	min error
DenseGATs (Ours)	0.0961	1.2551	0.0004
Ori.GAT	0.1693	2.1643	0.0009
NeuroSkinning network	0.1378	1.7689	0.0006
Ours w/o self-reinforcement	0.1153	1.3986	0.0002
Ours w/o dense connection	0.1425	1.4852	0.0006

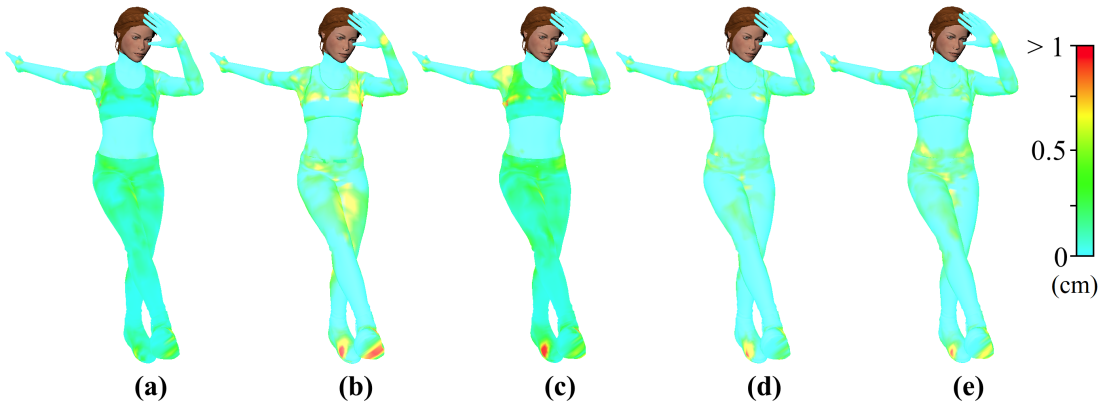


Figure 3.14: Comparison of different network architectures. From left to right: (a) ours, (b) original GAT, (c) NeuroSkinning, (d) ours (no self-reinforcement stream), (e) ours (no dense connection).

To verify the effectiveness of our proposed network, we also conducted experiments

on character models in the test set with hip-hop dancing motion to compare the performance with different network structures like those shown in Table 3.3. We first compare with the original GAT [74] network which contains four graph convolutional layers. Next, we evaluated the network described in NeuroSkinning [51] and followed the same experimental settings. Based on our proposed structure, we respectively removed the designed self-reinforced stream in each GAT block and dense connections between each GAT block. The prediction results are summarized in Table 3.3 and an example frame of the animation is shown in Figure 3.14. It could be observed that the original GAT network sometimes fails to represent complicated graph information and produces highest errors due to the limited representation of the shallow convolutional layers. The NeuroSkinning network has the same problem as the original GAT because there are only three graph convolutional layers in the whole structure. For DenseGATs structure without self-reinforced stream and the structure of multiple GAT blocks without dense connection, features can be well learned and the obvious approximation errors can be improved to a certain extent, but they still cause some undesirable deformation in several joint regions. In contrast, with our proposed network, the best prediction result can be obtained with improvement rates of about 16.7% and 32.6% compared with the DenseGATs without self-reinforced stream and without dense connection. This suggests that our network has better generalization ability owing to its self-reinforcement step and dense propagation step.

3.2.3.4 Conclusion and discussion

We have presented a DenseGATs network that leverages existing well-designed skinning features of characters to accurately predict deformation for new characters. To the best of our knowledge, our method is the first deformation approximation solution based on graph learning to perform nonlinear refinement of different characters in various poses. Our GAT blocks and dense modules allow for efficient utilization and transmission of self and adjacent information throughout the network. Experimental results demonstrate that through these strategies, the skinning features from other characters can be reused and the network is able to generate realistic nonlinear deformation results that are very close to ground truth. The high-quality deformation predicted with our method can be

directly used for masses of similar characters in films, thus reducing the efforts made by artists to re-rig for new characters each time.

Despite the success of addressing all challenges as mentioned in Section 1.2.1: ***case1-chall.1-4***, large-scale character datasets with a significant number of poses are normally required for training to learn such automatic deformation tasks for the 1^{st} case. Therefore, it is necessary to further figure out ways to further improve the generalization ability of the model, so that ***case1-chall.4*** can be solved more thoroughly.

There is still room for improvement in two aspects. First, when expressing deformations through graph node features, using the vertices position as features cannot provide the spatial invariance. For example, this representation will result in graphs with different spatial positions (but with the same pose) corresponding to the same output deformations, which will greatly increase the requirements for the number of poses in training. Therefore, our next first goal is to design representative graph features to concisely express character deformations in different poses independent of spatial position. Once the expressive features are obtained, next, a graph network needs to be designed for building the relationship between encoded graph features and final deformation predictions. Although the network of DenseGATs can handle graph features effectively thanks to the designed GAT block and dense module, it cannot essentially simplify different mesh graphs and extract more general features to further avoid overfitting and improve generalization ability. To this end, our next second goal is to propose a novel network structure with better generalization ability that can summarize features from existing character samples. The above two aspects inspired our next MultiResGNet research.

3.3 Deformation with MultiResGNet

3.3.1 Mesh encoding with improved graph features

To make graph nodes informative and discriminative among diverse meshes with different geometries and different transformations, it requires to assign attributes to each node. While the feature design method mentioned in the last subsection can be followed, we found that the approximated results are affected by the amount of character translations

due to the global position of the node attribute of the methods. To strengthen the feature expression, we replace the original vertex position vector p_i with the novel designed feature descriptor $r_i(\boldsymbol{\theta})$.

Concretely, to represent skinned mesh features with various poses, given the pose vector $\boldsymbol{\theta} = [\boldsymbol{\omega}_0^T, ..., \boldsymbol{\omega}_s^T, ..., \boldsymbol{\omega}_S^T]$, where S is the total number of joints of a rig, and $\boldsymbol{\omega}_s \in \mathbb{R}^3$ denotes the axis-angle of joint s. We define a novel relative skinning feature descriptor as $r_i(\boldsymbol{\theta}) \in \mathbb{R}^3$. When the joint s is rotated by an angle, the corresponding rotation matrix can be expressed as:

$$G_s(\boldsymbol{\theta}) = \prod_{p \in \mathcal{A}(s)} Z(\boldsymbol{\omega_p}), \tag{3.7}$$

$$Z(\boldsymbol{\omega_p}) = \boldsymbol{I} + \sin(\|\boldsymbol{\omega}_p\|)\hat{\bar{\omega}}_p + (1 - \cos(\|\boldsymbol{\omega}_p\|))\hat{\bar{\omega}}_p^2, \tag{3.8}$$

where $p \in \mathcal{A}(s)$ is the ordered set of joint ancestors of joint s. $Z(\boldsymbol{\omega_p})$ is a local rotation matrix computed by Rodrigues formula through axis angle $\boldsymbol{\omega_p}$, skew symmetric matrix $\hat{\bar{\omega}}_p$ corresponding to vector $\boldsymbol{\omega_p}$, and identity matrix $\boldsymbol{I}$.

Then, to reflect the influence of bone movement on each mesh node, we combine the rotation matrix, linear skinning weight and rest pose position. Thus, our relative skinning feature descriptor can be defined as:

$$r_i(\boldsymbol{\theta}) = \sum_{k=s}^{S} w_{s,i} G_s(\boldsymbol{\theta}) G_s(\boldsymbol{\theta}^*)^{-1} \bar{r}_i, \tag{3.9}$$

where $G_s(\boldsymbol{\theta}^*)$ is the rotation matrix of joint s in the rest pose $\boldsymbol{\theta}^*$. $w_{s,i}$ and $\bar{r}_i$ respectively denote the skinning weight of the vertex v_i influenced by the joint s, and the rest pose position of the vertex i. Note that $r_i(\boldsymbol{\theta})$ is translation-invariant instead of the simple vertex position, because $G_s(\boldsymbol{\theta}^*)$ is independent of joint position.

In total, together with the normal of vertex $n_i \in \mathbb{R}^3$ and the distance vector $x_i = [x_{i,1}, ..., x_{i,s}, ..., x_{i,S}] \in \mathbb{R}^S$, the final feature vector of graph nodes can be expressed as: $v_i = [r_i^T(\boldsymbol{\theta}), n_i^T, x_i^T]$. The dimension of v_i is also $6+S$. Such graph feature representation has the expressive power of basic linear-based deformations that can capture the skinning features across different joint transformations (*i.e.*, $r_i(\boldsymbol{\theta})$), the whole range of mesh surface features of different shapes (*i.e.*, n_i), and the binding relations between joints and meshes (*i.e.*, x_i). The designed feature descriptors are translation-invariant and

enable the network to handle data more effectively. This improved graph construction solution is proposed in our published work of MultiResGNet [49].

3.3.2 Deformation refinement through MultiResGNet

The network of DenseGATs has achieved success on automatically approximating non-linear deformations for new animated characters. The designed operations for networks (*e.g.*, self-reinforced stream, dense connection) are workable to a certain extent that make the prediction results satisfactory, the generalization ability of the trained networks are relatively not strong enough that output predictions sometimes are sensitive to tiny changes of the features in the input.

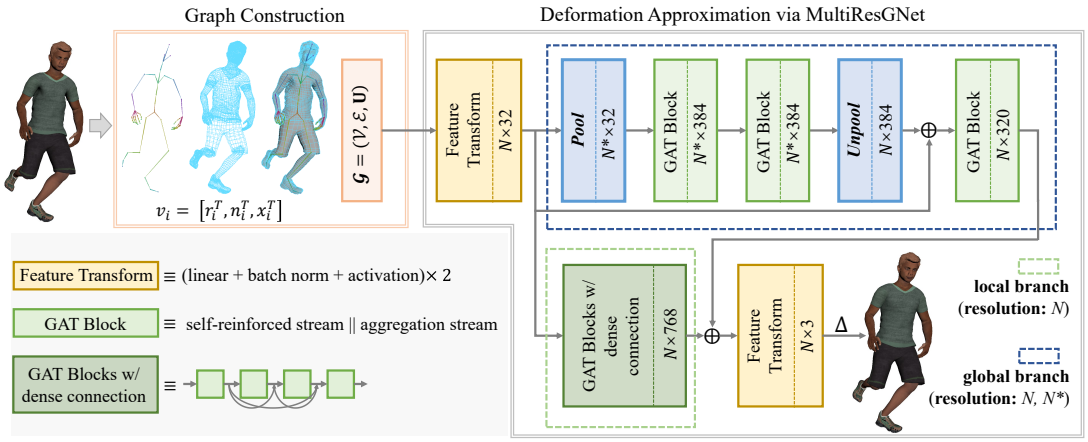


Figure 3.15: The framework of MultiResGNet.

We therefore propose a novel and effective approach to address limitations of network generalization capabilities. Our proposed model (Figure 3.15), named Multi-Resolution Graph Network (MultiResGNet), integrates multi-resolution graph features for training and inferencing. It contains a local branch and a global branch, dealing with original-resolution graphs and down-sampled lower-resolution graphs respectively. At last, processed features from two branches will be concatenated, and then pass through a feature transformation module to generate the final predictions. Here, the number of GAT blocks and pooling/unpooling operations in the two branches are designed according to the optimal computing time and accuracy of our data.

3.3.2.1 Local branch

The local branch utilizes several densely connected graph-attention-network (GAT) blocks to deal with local features. We adopt the same GAT structure with dense connection pattern as mentioned in DenseGATs [48]. Specifically, feature nodes $\vec{v}_i^{[l]}$ is processed after m GAT blocks with the dense connection pattern, resulting in $\vec{v}_i^{[l+m]}$.

Despite the outstanding performance in extracting detail features, pure single-resolution graph convolution is not enough for complicated character deformation approximation. It neglects the entire graph structure information and tends to overfit the relationship between input features and output deformations, thus sometimes generalizing poorly to new characters and poses. To address the aforementioned limitations, we additionally propose the global branch with the attention-based pooling operation to globally summarize all the node feature representations.

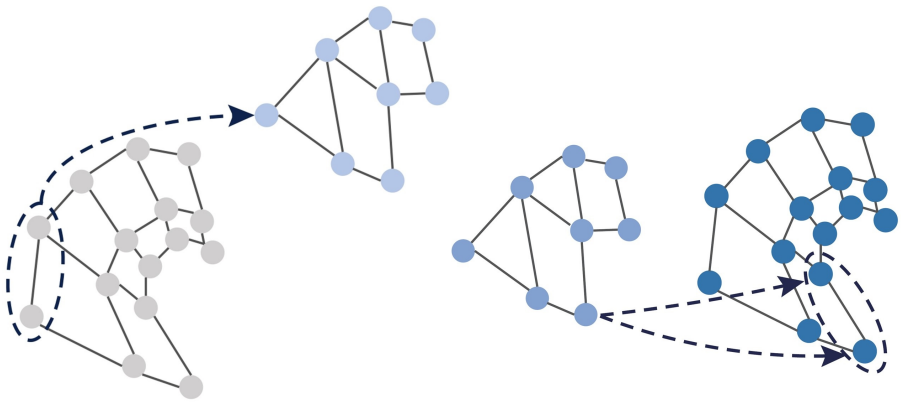
3.3.2.2 Global branch

The global branch utilizes lower-resolution graphs (with sparse vertices) to capture the overall structural information. To achieve the coarse representation of the original graph, there is a need to first define the pooling strategy. In our work, we present an edge contraction operation which follows the attention mechanism to calculate the coefficient $\alpha_{ij}^{[l]}$ of a node with its neighbors in Equation (3.4). We regard the obtained coefficient as being the edge score of each pair of nodes. By sorting all edges of their scores, the node which is adjacent to i with the highest attention coefficient can be depicted as:

$$j^* = \arg \max_{j \in \mathcal{N}(i)} \alpha_{ij}^{[l]}. \tag{3.10}$$

Iteratively, we implement edge contraction on the overall mesh. Edges with highest score are contracted and the nodes that belong to the contracted edges are ignored for other edge contractions. For example, if j^* is selected by i, they will be merged and then j will no longer participate in other node's merging. In this way, after all nodes have been traversed, that is, nodes have been merged once or the node cannot join the merge of its adjacent nodes (because all adjacent nodes have been merged with others), one pooling ends. The defined pooling operation allows both nodal features and topology features to yield hierarchical representations. Through this process, the total number of

nodes will become N^* that roughly equals to 50% of the original N as shown in Figure 3.16 left. Specifically, if all graph nodes have been merged (every two nodes becomes one node), the N^* is exactly 50% of the original N; if there remain some separate nodes which have not been merged, the N^* is larger than 50% of the original N.



Pooling Unpooling

Figure 3.16: Pooling and unpooling in action on mesh graph nodes.

The pooling operation will create newly merged nodes. Together with the edge score, we combine the features from the previous pair of nodes to obtain the new node features:

$$\vec{g}_{ij^*}^{[l+1]} = \alpha_{ij^*}^{[l]}(\vec{v}_i^{[l]} + \vec{v}_{j^*}^{[l]}). \tag{3.11}$$

In order to learn the global information, the new node features $\vec{g}_{ij^*}^{[l+1]}$ of the lower-resolution mesh graph are then fed as input for several GAT blocks. After processing with b GAT blocks, the features are transformed into $\vec{g}_{ij^*}^{[l+b+1]}$.

For the inverse of pooling, we also perform the corresponding unpooling operation. To restore the graph to its original structure, nodes are given the mapping of location information from the pooling layer. After this step, the number of nodes will be restored from N^* to N as shown in Figure 3.16 right. Moreover, the unpooled features of nodes are computed with the edge score:

$$\vec{g}_i^{[l+b+2]} = \vec{g}_{j^*}^{[l+b+2]} = \vec{g}_{ij^*}^{[l+b+1]} / \alpha_{ij^*}^{[l]}. \tag{3.12}$$

Lastly, the output features from the local branch and global branch are concatenated together. This strategy enables fusion of the fine detail information and spatial context

information. These fused features finally undergo a feature transformation module f_{trans} to enable the final prediction:

$$\Delta_i = f_{trans}(\vec{v}_i^{[l+m]} \| f_{\text{GATB}}(\vec{g}_i^{[l+b+2]})), \tag{3.13}$$

where $\Delta_i \in \mathbb{R}^3$ is the corrective displacement of the node i in the graph. While forwarding a linear-deformed mesh graph into our MultiResGNet $W(\cdot)$, it can produce a set of refined displacements Δ of mesh vertices. It should be noted that the trainable weights ($\boldsymbol{W}$, $\boldsymbol{H}$, etc.) in the network are related to the dimension of node features but are not affected by the number of mesh vertices N. Therefore, the trained networks has no constraints on the number of vertices and can be applied to meshes with different topologies. Ideal properties of such multi-resolution graph network include having a strong generalization ability of deformation approximation for unseen characters and the ability to refine details with the help of existing deformation knowledge.

3.3.3 Evaluation of MultiResGNet

3.3.3.1 Dataset

We use 75 character models by different shapes and customizations for training, five characters for validation, and 10 characters for testing with Adobe Fuse CC. All characters share the same standard skeleton with 65 joints. Each character contains linear-based deformations as the baseline, and high-quality nonlinear deformations as the learning objective. Specifically, the LBS method is used to create linear-based deformations where skinning weights are directly determined by the geodesic voxel distance without modification and a vertex is set to be influenced by no more than four joints. Here, we have tried to replace this LBS baseline with Dual Quaternion Skinning (DQS) method [33], and found that the accuracy of the approximation results has no obvious improvement. In particular, each character in our training set is animated with 250 poses, of which 150 poses are randomly generated and the remaining 100 poses are from frequent motions of walking, jumping, and running. Here, because of our translation-invariant features, the number of training poses is significantly decreased compared with DenseGATs. Although our features are not rotation-invariant (because the features are still related to the rotation angle of the bones), by randomly sampling

joints range to create various random poses, our method can ensure accurate predictions of deformations under the pose within a reasonable range. To further ensure the robustness of the approximator, data augmentation (*e.g.*, randomly rotating meshes) could be used when creating the dataset in the future.

To further evaluate our method on standard human models, we also leverage the SMPL dataset [52] which contains detailed deformations of different bodies performing animation sequences. For our training set, it includes five female subjects and three male subjects. These characters are deformed with sample poses, with the total number of 2955, which are selected from motion sequences of dancing, jumping, and butterfly kicking. For testing, we use one female and one male characters and animate them separately with dancing and front kicking motions.

3.3.3.2 Implementation details

As shown in Figure 3.15, after graph construction, the graph features are first normalized using the mean and variance for each dimension and input into a feature transformation module that contains two hidden layers with hidden neurons of (64, 32) and applied with batch normalization and tanh nonlinear activation function. Then, the transformed features are forwarded into two branches for the processing of multi-resolution graphs. To learn the overall structural features, pooling is implemented so that the number of nodes is reduced by roughly half to become N^*. The size of the feature dimension remains unchanged. The lower-resolution graph features are then processed by two GAT blocks. Each GAT block involves a graph-attention-based aggregation stream with hidden feature size of 16, multi-head number is 8, and a self-reinforced stream with hidden feature size of 128. Features will be transformed with tanhshrink activation function in the last layer of the GAT block. Unpooling operation, which is the inverse operation of pooling, is conducted to restore the original graph resolution of N. For the local branch, the parameters setting of GAT block is the same as that mentioned for the global branch. Finally, features from two branches are concatenated together and fed into the last feature transformation module with two hidden layers, where hidden neurons are (256, 64).

Training was implemented on nVIDIA GeForce RTX2080Ti GPU. During the train-

ing, we used the Adam optimization algorithm with an initial learning rate of 0.01 and set the decay factor with 0.75 to reduce the learning rate when loss stopped decreasing for eight epochs. To minimize errors between approximated and ground truth displacements, the mean squared error is used as the loss function. To ensure stable training, we randomly located a vertex and selected the surrounding 1024 consecutive vertices each time until traversing the entire meshes. The total training with our created dataset and with SMPL dataset took roughly one week, and 25 hours respectively.

3.3.3.3 Results of MultiResGNet

To demonstrate the generalization capabilities of our method to new poses, we evaluated a character appearing in the training set and animated it with an unseen motion of playing golf. Figure 3.17 shows the per-vertex mean distance error of rough linear based deformation, LBS method, DQS method and our proposed method in 50 frames. It can be observed that through our MultiResGNet prediction, the generated nonlinear shape corrections are relatively average, with about 0.28cm improvement compared with the input rough linear-based deformation. For the deformation of LBS and DQS, even if the skinning weights are manually adjusted, due to the inherent limitation of the algorithms, the overall mean errors are still relatively large. In contrast, with our graph-learning-based approximation, the mean error of the deformation reaches to 0.09cm. Based on significantly decreased pose training samples, our approximation with lower errors demonstrates that our proposed network with novel graph descriptors has a good generalization ability for predicting nonlinear deformation with new poses.

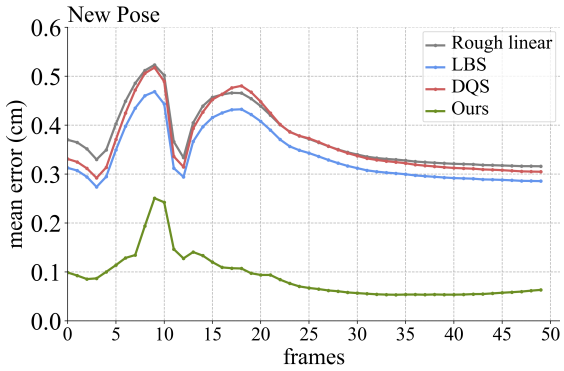


Figure 3.17: Quantitative evaluation of generalization to new poses. The test motion "playing golf" with 50 frames is applied with a character in training set. We show the mean distance error, comparing our method with the input rough deformation of linear-based deformation, LBS, and DQS.

To evaluate the generalization ability of our method to new characters, we first define the new characters as having different shapes (but the same number of vertices) or different subject models (different mesh geometries) from the characters in training set. For these two cases, we separately conducted experiments with walking and running motions to evaluate our trained networks. As shown in Figure 3.18, the per-vertex distance error of the new shape case is about 0.06cm and of the new subject case is about 0.11cm on average. Compared with the input rough linear-based deformations, our approximations can increase the average accuracies of 0.16cm and 0.13cm respectively. We found that our approximation has less errors and higher accuracy improvement for new shape deformations. It illustrates that our networks have better generalization abilities for new shapes where the geometry structure of meshes is unchanged from the character in the training set. When approximating for new subject models, the trained networks also have certain inference although the mesh geometry is completely new.

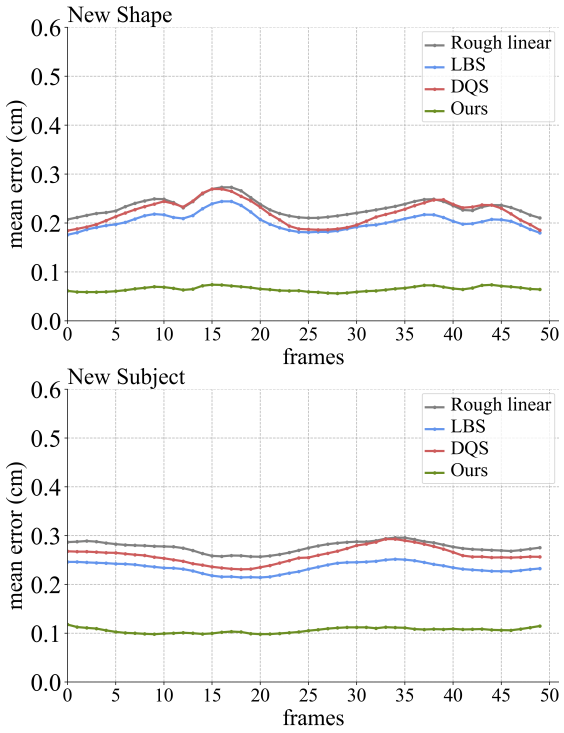


Figure 3.18: Quantitative evaluation of generalization to new characters. Top plot: Per-vertex mean error of a new character shape deformation with walking motion in 50 frames. Bottom plot: Per-vertex mean error of a new subject model deformation with running motion in 50 frames.

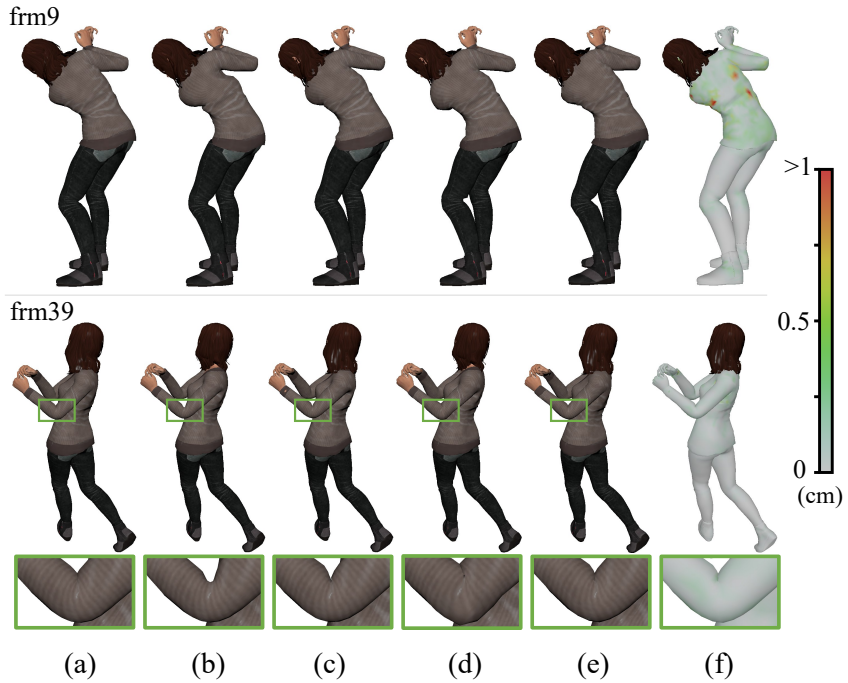


Figure 3.19: Evaluation of generalization to new poses. We show deformations of the motions of a character playing golf in the 9th and 39th frames, and side-by-side compare the ground truth (a), rough linear-based deformation (b), LBS with weight refinements (c), DQS with weight refinements (d), our prediction (e) and our approximation colormap (f). The vertices of the approximated mesh are colored to indicate the per-vertex distance error in centimetres.

To illustrate Figure 3.17 in a more intuitive way, we show the qualitative deformation results under the animation of playing golf in Figure 3.19. We focused on the representative frames which have the largest distance error (frm9) and the average error (frm39) of all frames. In the 9th frame, the largest errors tend to occur in the armpits and shoulders due to bending and substantial stretches. Since the skinning weight are automatically generated without modification, rough linear-based deformation has the significant artefacts in the right armpit. The deformations of LBS and DQS have improved this problem to a certain extent due to the manual refinements, but the visible volume loss and joint bulging (in the shoulder area) still exist in the results of LBS and DQS respectively which cause the mean errors of the entire body to be still large in Figure 3.17. Our deformation tends to produce the result that is most similar to the ground truth. The overall approximation error of upper body is higher than other body parts, this is because the large movements are concentrated on this area. However, these

errors are relatively small compared to the whole body and barely noticeable during the animation. In the 39th frame, we provide the detail of deformation results when the elbow is bent. Visually, the approximation with our method can nicely mimic the desired elbow bending behavior around the joint and give the best deformation effect than other methods. Also, the whole character can be deformed well so that is hardly distinguishable from the ground truth.

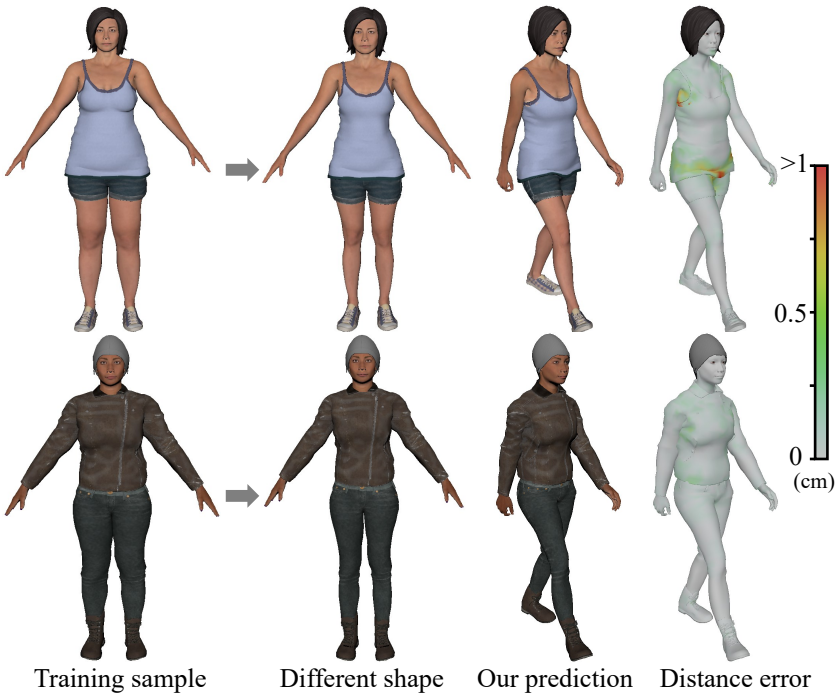


Figure 3.20: Evaluation of generalization to new shapes. Test thinner characters in walking poses. Colormaps depict the per-vertex distance error of the predicted deformation.

Figure 3.20 shows the sample frames of walking motion for two test characters with new body shapes. For the character in the lower row, the error colormap on the mesh demonstrates the outstanding generalization ability of our trained network where the approximated deformation closely matches the ground truth. The numerical error of each frame corresponds to the Figure 3.18 - generalization to new shapes. In the upper row of Figure 3.20, because of the change in the body shape and the lower edge of the camisole does not fit the body, the biggest deformation error occurs when the character takes a step. In addition, the inaccurate deformation in the right armpit area is approximated because mesh vertices in this region are affected by multiple joints and

the region is also the edge of camisole that is adjacent to the body, thus tending to be problematic.

Figure 3.21: Evaluation of generalization to new subjects. Test new subject in a running pose. Based on the rough linear-based deformation, our method corrects each vertex with a displacement so that the deformation becomes nonlinear, and no noticeable errors are found in the right colormap.

Additionally, in Figure 3.21, we visually evaluated the quality of our proposed approach on generalization to new subject models, where we compared the deformations between ground truth, linear-based method, and our approximations. The test character model is a new subject that its mesh geometries are completely different from characters in training set. As shown in the figure, the overall deformation is successfully predicted using our MultiResGNet, with better performance than the linear-based deformation. Detailed improvements are achieved especially in the area where the pants and belly contact. The prediction error distribution of the whole body is very small and relatively average, for the reason that the clothing is fitted and the running motion is a whole-body movement. Here, the hair part was not approximated by our network, so it remained the same as linear-based results.

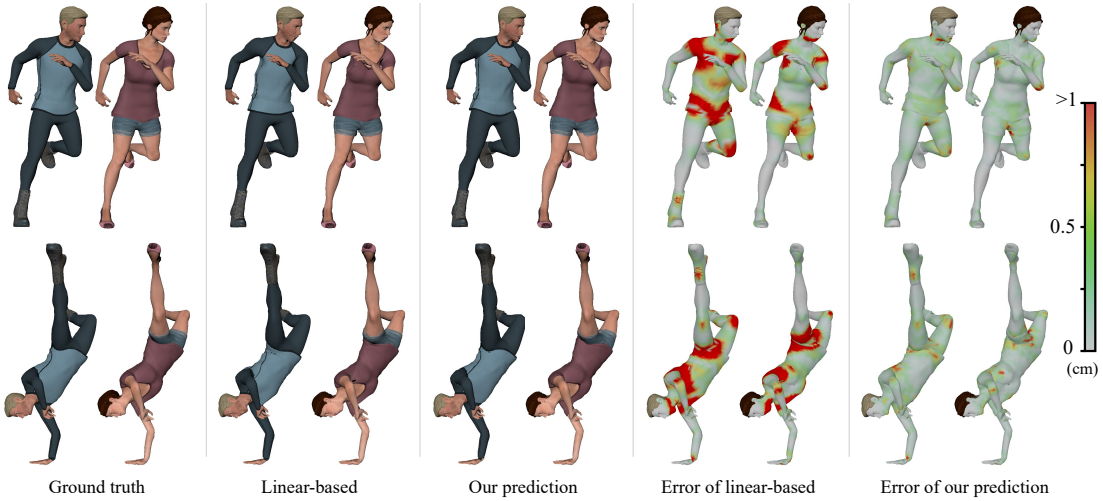
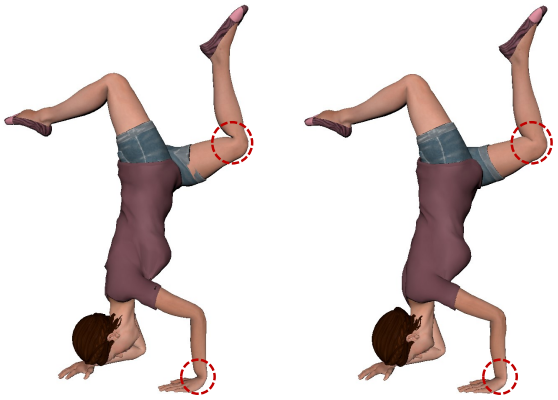


Figure 3.22: Evaluation of generalization to new characters with new poses. The dancing motion is used for testing for two new characters. We show ground truth, linear-based deformation(as input), our approximation result and colormaps of per-vertex error.

Finally, we further demonstrate the generalization ability of our network to approximate nonlinear deformations for new characters with new poses. As shown in Figure 3.22, we selected two test characters, and animated them with dancing motions. As expected, plausible and realistic deformations can be produced without particularly noticeable artefacts. The areas with large errors are concentrated in the armpits and crotch, and near the joints which are controlled by multiple bones. As our network is only trained with a small number of representative character models and poses, they are still able to produce visually plausible deformation effects that can meet the animation production needs.



(a) (b)

Figure 3.23: Deformation results with vertex position features (a) and with our relative skinning features (b).

To verify the effectiveness of our proposed graph features, we performed an ablation study to compare deformations between common position features and our newly proposed relative skinning features. The graphs containing these two different features were respectively input into the same MultiResGNet, following the training settings mentioned in Section 6.2.1. We used the inverted pose for testing, and its position is not at the origin. In Figure 3.23, the approximated deformation with our features is more natural than with position features, especially in the areas of knees, wrists, and inner thighs. Specifically, the mean error of (a) and (b) is about 0.26cm and 0.15cm respectively. Because of the translation-invariance of ours, the local predicted offsets remain unchanged even though the global position changes significantly. In this way, the trained network can learn a variety of deformations through a small number of training poses (250 poses per character), and is effective for all spatial positions.

3.3.3.4 Comparison

we conducted experiments with two types dataset, *i.e.*, our created humanoid characters with different customizations and the SMPL realistic human bodies, to measure the generalization ability of our proposed deformation method.

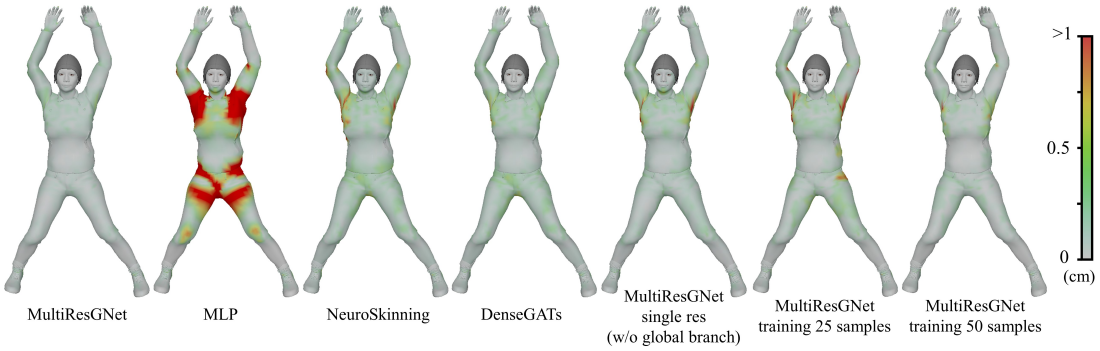


Figure 3.24: Comparison among different learning-based methods. Colormap shows the distance error to ground truth.

Comparison on our created dataset. We compared our DenseGATs and MultiResGNet with other learning-based methods. In addition, to further evaluate the generalization ability of our proposed MultiResGNet, we also compared the performance of the different network structure and the different number of training samples, where the global branch used in our method was removed and the amount of training characters was decreased to 25 and 50. As listed in Table 3.4, we respectively compared with MLP (with hidden neurons of (256, 512, 512, 128), batch normalization and tanh activation function), NeuroSkinning [51], DenseGATs [48], MultiResGNet without the global branch (with single-resolution graphs), and MultiResGNet trained with 25 and 50 samples. Since the network of NeuroSkinning also uses GAT, here we compared with its structure and predict our output (instead of the weights in its original paper). For the NeuroSkinning and DenseGATs, we adopted the same features in their original work and trained the networks with 75 characters and 8500 poses per character. For the training of MLP, we utilized our proposed features and also trained it with 75 characters and 8500 poses per character.

Table 3.4: Evaluation of predicted distance errors (cm) using different learning-based methods.

Structure	mean error	max error	min error
MultiResGNet	**0.1121**	**0.5795**	0.0038
MLP	0.3962	13.0968	**0.0017**
NeuroSkinning	0.2377	1.3316	0.0104
DenseGATs	0.1887	1.0324	0.0099
MultiResGNet single res	0.1610	1.2328	0.0101
MultiResGNet train 25 samples	0.2594	1.4826	0.0101
MultiResGNet train 50 samples	0.1728	1.3570	0.0103

To evaluate the generalization ability of our MultiResGNet method, we respectively trained it with 75 characters (original training data setting), 50 characters, and 25 characters, and each of the character was animated with 250 poses. To verify the effectiveness of the multi-resolution strategy, we removed the global branch from the original structure and used the same training amount with 75 characters and 250 poses. The qualitative results are shown in Figure 3.24. As observed, the greatest approximation errors are generated by the MLP network, which shows that the pure fully connected network cannot achieve large amounts of complicated approximations of nonlinear offsets and hardly apply the prior skinning knowledge to new character models. As graph-learning-based methods, NeuroSkinning and DenseGATs successfully predict the overall vertex displacements for characters whose translation does not change dramatically. Also, there are still undesirable effects on the areas near the armpits and belly. For the MultiResGNet without the global branch, since the graph resolution throughout the network is constant that equals to N, it can produce convincing effects in most mesh areas but also causes some obvious errors. With less training data, the approximation accuracy of our method could maintain to some degree, but deformation errors near joints are also produced. We found that setting the number of training characters to 75 allows the deformation predicted by the network achieve visually plausible effects that errors of mesh vertex deviations are within a small range. Thanks to the representative graph features and multi-resolution graph operation, our method enable easier generalization to new characters with new poses based on knowledge which is learned from existing deformations.

Comparison on the SMPL dataset. Since DenseGATs [48] and MultiResGNet both approximate deformations by estimating the nonlinear residuals based on rough deformations, and both achieved the good result in the previous experiment, next we will apply the SMPL dataset to conduct further comparison.

In Figure 3.25, we demonstrate the generalization capabilities to a new SMPL character body. In particular, we followed the training setting mentioned in Section 6.2.1 and then test the trained network using one female character (height: 169cm) with new body mass performing the dancing motion sequence. Since this task is not complicated, both two methods have achieved satisfactory results with a very low average error (DenseGATs: 0.15cm; MultiResGNet: 0.08cm) relative to the ground truth. In contrast, the MultiResGNet generates more natural deformations in the areas of base of the thigh and the chest near the upper arm that is clamped.

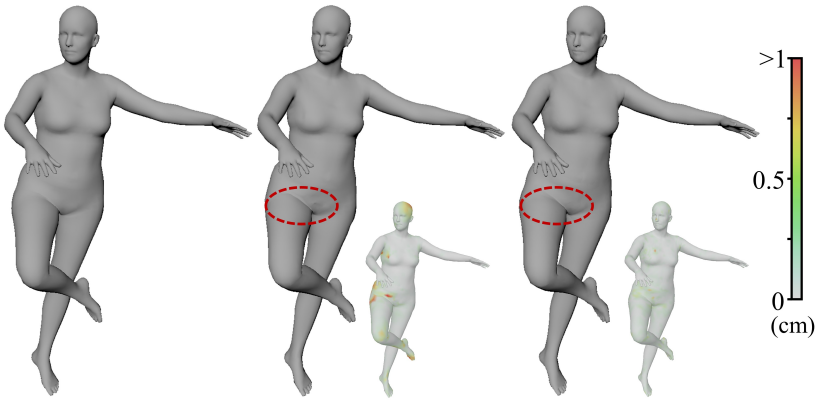


Figure 3.25: Generalization to a new SMPL character body: DenseGATs [48] and MultiResGNet.

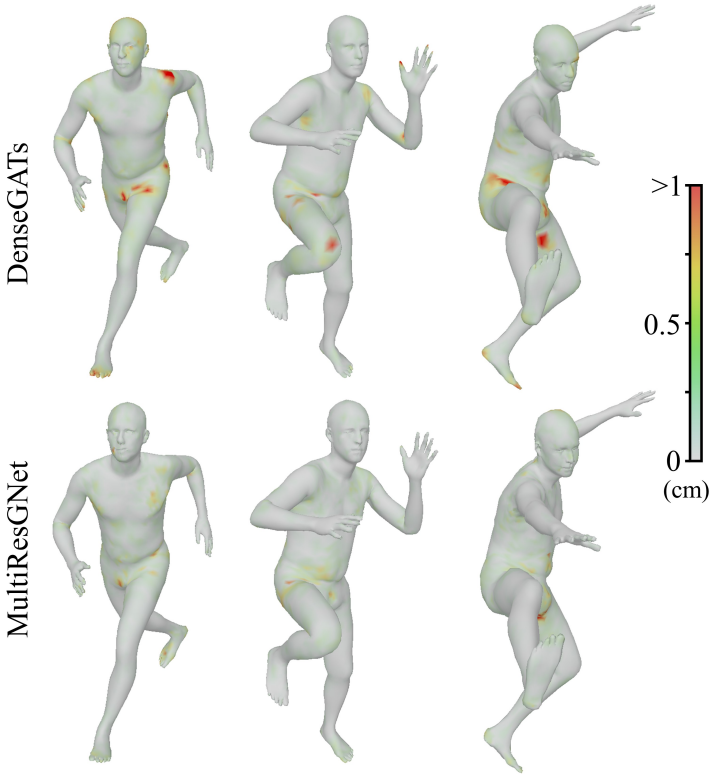


Figure 3.26: Generalization to a new SMPL character body with new front kicking poses: DenseGATs [48] and MultiResGNet.

Furthermore, with the SMPL data, we also quantitatively demonstrate the generalization capacities to the new character body and poses. Specifically, we used the trained network to predict deformations for a male subject (height: 177cm) with the front kicking motion, while both the character and the motion are unseen in the training set. As shown in Figure 3.26, the deformation results of DenseGATs tend to generate large approximation errors in regions near joints. Unlike DenseGATs, the relative skinning descriptor of ours includes rich pose knowledge which can be used to approximate new complicated poses. To this end, our proposed method could make full use of prior feature knowledge and outperforms DenseGATs for unseen cases. Numerically, the average and maximum errors of the results using our method are (0.14cm, 1.32cm), which are also better than theirs (0.23cm, 1.80cm).

3.3.3.5 Conclusion and discussion

We present a graph-learning-based method MultiResGNet for automatically approximating nonlinear deformations. Compared with DenseGATs, the advantage of our approach is that the proposed network has better generalization ability, that can use fewer training samples for accurate deformation approximation. Our novel relative skinning feature descriptor breaks through the lack of translation-invariance of using vertex position to represent pose information in the past learning-based methods. Even if it is not rotation-invariant (because it is calculated by joints rotation angles), through encoding it and the other two feature descriptors, the graph mesh attributes along with poses still can be effectively expressed in a more efficient way. Next, by using our proposed novel expressive graph features, we are able to model nonlinear offsets as a function of mesh graphs, with the help of the multi-resolution graph network. We conducted experiments to evaluate our method and the results demonstrate that the generalization abilities to new poses and new character models outperform existing methods.

3.4 Summary of the 1^{st} Case Deformation Methods

In this chapter, we have presented methods of DenseGATs and MultiResGNet for the 1^{st} case deformation approximation, that both take mesh graph as the input and use the graph-based network to output nonlinear correction offsets. The proposed methods can predict deformations of new characters with new poses, so they can be applied to animation pipelines to reduce the manual labor of animators. DenseGATs is the first method to propose the pioneering idea of using a graph-based framework to predict the nonlinear deviation under each action. Two technical contributions: GAT blocks and dense modules enable the network to effectively process huge numbers of complex graph features and make reasonable mappings. On the basis of DenseGATs, we further propose the method of MultiResGNet to improve the generalization ability of the model. By designing the novel relative skinning features and processing multi-resolution graphs with two-branch MultiResGNet, the method is able to use fewer training samples to achieve realistic deformation approximation. We showed the importance of all proposed

solutions. Experimental results illustrated that our methods achieve better performance than prior studies in deformation approximation for unseen humanoid characters and poses.

Since our DenseGATs and MultiResGNet are both learning-based methods, the quality of the deformation approximation also depends on the data in the dataset. In order to make the network easier to learn, the baseline deformation (coarse deformation) in the dataset should have the same settings, so that the model can better learn the deviation from the baseline to the final accurate deformation. In addition, currently, our dataset only contains tight-fitting characters (with bare legs or wearing tight pants), so it is better to ensure that the test character is also such situations in order to have high-quality deformation predictions. Finally, because the dimension of graph features input into the network are related to the number of skeletal joints, when using the well-trained network, the test character must have the same number of joints. Currently, the characters in our dataset have the standard humanoid skeleton with 65 joints, which can be applied to most two-limbed animated characters in animations.

Chapter 4

Deformation Approximation - 2^{nd} Case

4.1 Context and Observations

4.1.1 Background of the 2^{nd} case deformation

In application fields including virtual try-on, online shopping, and fashion design games, generating the interaction between clothes and body is an important research content in the field of computer graphics. The classic –and still nowadays prevalent– approach is based on physics-based methods [61, 62] which formulate deformations within a simulation framework. Despite the convincing effects generated by these methods, the expensive computational cost and potential instabilities obstruct their deployment in real-time applications.

As shown in Figure 1.4, the deformation in this situation tends to represent complicated wrinkles of clothing appearance. To achieve such visual effects and reduce computing time, as an alternative approach to physics-based deformation methods, learning-based solutions have recently received more attention from researchers. Their core idea is to learn a model which can mimic the deformation behaviors after training with the dataset. Studies [18, 63, 68, 72, 75] have demonstrated the possibility of realizing nonlinear deformations using trained models for clothing animation. However, two obvious limitations of these learning-based models still exist: the dramatic inability to generalize to new garments and new poses, and the inability to produce highly complex folds. In the next subsection, we will discuss the reasons for these limitations.

4.1.2 Unsolved problems in state-of-art studies

Most state-of-art studies [63,68,72] adopt MLP models to predict clothing deformation that they input a flattered vector consisting of body shape and pose parameters and output the deformed garment. Although garments predicted by this method contain plausible wrinkles, due to the limitation of MLPs architecture, the dimensions of the input and output vectors, which are related to the number of vertices, need to be fixed, enforcing training and test garments always have the same topology with the same number of vertices. Succinctly, using MLPs will suffer from the fundamental generalization problem that the trained model is only applicable to the specific garment. In addition, because of the limited ability of MLPs to understand 3D information, a large number of models are usually required to complete the deformation approximation task.

To address the above limitations, recent work [18,75] propose graph-learning-based solutions for clothing deformations. Even if they avoid the problems of MLPs and allow the generalization to new meshes, the generalization ability is still weak that the predicted results tend to be overly smooth without rich wrinkles. Moreover, due to the weak generalization, the model [75] only estimates deformations in one pose (if various poses are introduced, the task will become more complicated and the prediction accuracy cannot be guaranteed). Essentially, existing graph-learning-based methods fail to stably map large amounts of complex graph features to corresponding deformations; therefore, they overfit between input and output, and generalize poorly to new subjects and poses. Three aspects (as listed in Section 1.2.2: ***case2-chall.1-3***) have led to the above problems. First, the input parameters in previous studies do not consider the parameters that implicitly have an important influence on the fold. Second, the straightforward prediction of the clothing deformation (displacement with the body) is an extremely difficult task, resulting in overly smooth unrealistic results. Last, the network architecture is designed to directly approximate deformations as a function of graphs which results in lower performance with weak generalization ability. These keys challenges motivate the proposal of our GarFitNet.

4.1.3 Our observations and proposal outline

Different from the 1^{st} case of deformation approximation for characters with an integral mesh of the garment and the cloth, in the 2^{nd} case, the garment and the body are designed separately with their own mesh. In this situation, the garment mesh M_{coarse} cannot be directly obtained as in the 1^{st} case (just direct use linear-based skinning for the whole object who is already dressed), it requires an additional step: draping the garment on different bodies first. Through this step, a dressed character with the clothing deformation of M_{coarse} can be approximated. Then, to accurately reflect detailed deformations of garments, we learn a graph-learning-based model that produces fine-scale wrinkles $W(\mathcal{G};\Psi)$ for any given garment mesh topology. The outline including these two steps is depicted in Figure 4.1.

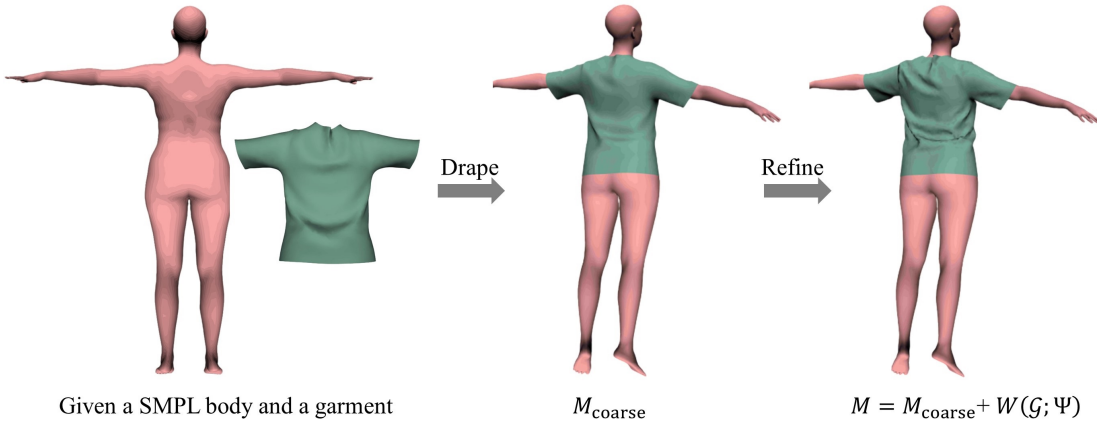


Figure 4.1: The outline of our GarFitNet.

In order to achieve complex clothing deformation, we first made observations on parameters that affect the quality of deformation. Our first observation is that the fit of the garment and body influences the degree of wrinkles. As shown in Figure 4.2, for the fixed clothing material, when the fit degree is from loose to tight, the wrinkles of garments are from smoother and sparser with wider width to detailed and denser with a narrower width. This observation inspired us to generate the fit parameter as one of the network inputs, which can better target the different suitability of clothing thereby producing more realistic deformations. We will discuss the detailed method based on this observation in the next section.

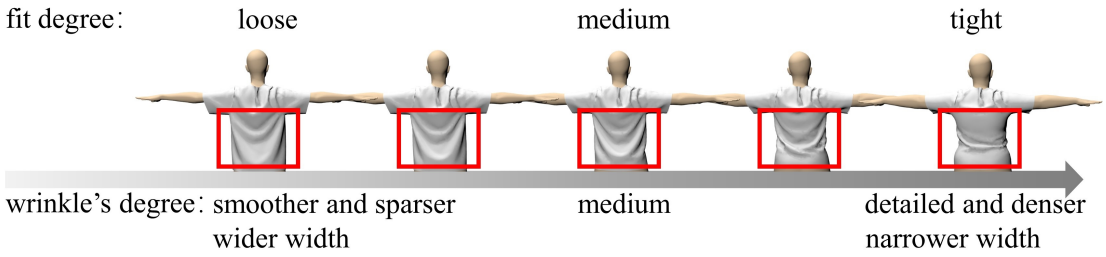


Figure 4.2: First observation: the fit of garment of body influences the wrinkles.

Another observation we made is that the task of directly outputting the deformation refinement of three-dimensional offset adjustments per vertex is extremely difficult, and the training with one graph neural network is hard to converge, resulting in the prediction results still being overly smooth. In response to this problem, past learning studies have chosen to increase the number of networks (with more than 20 MLPs) while sacrificing the generalization ability of the model [63], or reduce the difficulty of the task [75] (each garment is only predicted with one posture's deformation) to ensure the effective output. These have not fundamentally solved the issue of the high degree of freedom in the output dimension. In this work, we propose a novel decomposition method to decompose the output with a wide range $(-\infty, +\infty)$ to a narrow range which makes the learning task easy to converge therefore producing the desired deformation results with fine wrinkles for the arbitrary garment.

4.2 Coarse Deformation

In this section, we will describe the first step we address: the approximation of garment coarse deformations caused by the target body shape.

4.2.1 Fit parameter

As stated in Figure 4.2, the fit of the garment and body plays a key role in resulting deformation. In this work, we express the "fit" by generating fit variation α which is computed by using factor analysis to build the relationship between garments and bodies.

For the target body, we adopt the SMPL model [53] which represents the human body $M_b(\cdot)$ as a parametric function of shape ($\boldsymbol{\beta}$) and pose ($\boldsymbol{\theta}$):

$$M_b = W_{\text{smpl}}(\bar{M}_b, J(\boldsymbol{\beta}), \boldsymbol{\theta}, \mathbf{W}), \tag{4.1}$$

$$\bar{M}_b = T + B_s(\boldsymbol{\beta}) + B_p(\boldsymbol{\theta}), \tag{4.2}$$

where it computes a linear function $\bar{M}_b$ to add displacements caused by shape $B_s(\boldsymbol{\beta})$ and pose $B_p(\boldsymbol{\theta})$ to base mesh vertices $T \in \mathbb{R}^{N_b \times 3}$ in a t-pose, followed by learned skinning ($W_{\text{smpl}}(\cdot)$). Here, $\beta \in \mathbb{R}^{|\beta|}$, $|\beta|$ is the number of shape coefficients. $\boldsymbol{\theta} = [\boldsymbol{\omega}_0^T, ..., \boldsymbol{\omega}_s^T, ..., \boldsymbol{\omega}_S^T]$, where S is the total number of joints of a rig, and $\boldsymbol{\omega}_s \in \mathbb{R}^3$ denotes the axis-angle of joint s. $\mathbf{W}$ denotes the skinning weights of skeletal structure $J(\cdot)$.

Specifically, given a set of garment-body pairs (N_{pairs}) which consist of different SMPL body shapes and different garment sizes, at first, we compute the distance between un-posed garments $\bar{M}_g$ and bodies $\bar{M}_b$. Here, $\bar{M}_g$ and $\bar{M}_b$ respectively have N_g and N_b vertices. The distance D_i of the i^{th} garment-body pair is:

$$D_i = \|(\bar{M}_g - \boldsymbol{I}\bar{M}_b)_i\|_2, \tag{4.3}$$

where $D_i \in \mathbb{R}^{N_g}$. $\boldsymbol{I} \in \{0,1\}^{N_g \times N_b}$ is the indicator matrix to evaluates if the garment vertex is associated with the body vertex. In total, there are N_{pairs} of D_i to form the $\boldsymbol{D} \in \mathbb{R}^{N_g \times N_{\text{pairs}}}$. This $\boldsymbol{D}$ can indicate the distance between the garment and the body, so that can be used to measure whether the clothes are loose or tight. However, the dimension of $\boldsymbol{D}$ is relatively high and complex, and it is impossible for animators or users to have an intuitive experience, that is, to directly adjust a few numbers to change the fit degree. Therefore, in order to seek expressive and simple expression, we need a matrix decomposition method that studies optimal strategies for matrix dimensionality reduction. Although there are many matrix decomposition methods, the main advantage of factor analysis is that it can model the variance in every dimension of the input space independently [19]. Moreover, the different dimensional features generated from factor analysis seem more independent and semantically meaningful than methods that do not model heteroscedastic noise; therefore, we compute the fit parameter using factor analysis. Considering the speed of convergence, instead of using the expectation maximization (EM) algorithm, which is commonly used in machine

learning for computing the factor loading matrix, we apply a faster method named SVD-based likelihood optimization [11]. Factor analysis in matrix term is defined as:

$$\boldsymbol{D} - \boldsymbol{\mu} \approx \boldsymbol{L}\boldsymbol{A}, \tag{4.4}$$

where $\boldsymbol{\mu} \in \mathbb{R}^{N_g}$ is the mean vector. $\boldsymbol{L} \in \mathbb{R}^{N_g \times 3}$ and $\boldsymbol{A} \in \mathbb{R}^{3 \times N_{\text{pairs}}}$ denote the loading matrix and factors. $\boldsymbol{L}$ means the influence of each channel upon generating $\boldsymbol{A}$ in each dimension. In this way, $\boldsymbol{A}$ consists of N_{pairs} of vector $\alpha \in \mathbb{R}^3$ which provide a lower-dimensional suitableness presentation for each garment-body pair. As shown in Figure 4.3, the first component of α changes from small to large, resulting in the fit of the garment and the body changing from tight to loose.

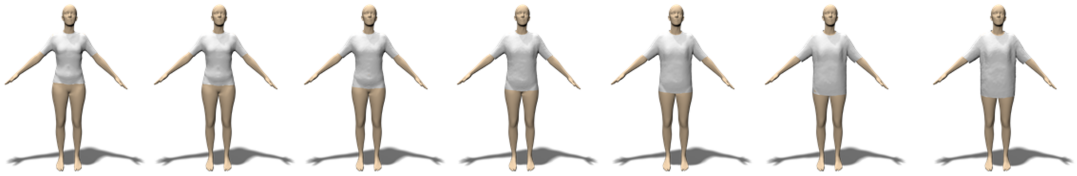


Figure 4.3: Overview of garment fit space.

4.2.2 Coarse garment deformation

Having the garment template provided by designers and parametric bodies from popular SMPL model [53], the first step is to drape the garment onto bodies with the desired fit. To produce the coarse effects M_{coarse} of the garment deformation, we seek to learn a regressor W_{coarse} that outputs a smoothed draped of the garment onto the target body shape with the fit degree α.

Two key elements are required to design the coarse regressor W_{coarse} to roughly estimate the deformation effect of the garment according to the body and the degree of fit. First, the regressor should have the generalization ability which is able to deal with arbitrary garments and bodies. Second, the regressor should be able to realize plausible performance that can infer the overall deformation based on features learned during the training. Specifically, in our work, we use the 4-layer graph-attention-based network to achieve this task (see Figure 4.4). we first build a mesh graph for body: $\mathcal{G}_{b'} = (\mathcal{V}_{b'}, \mathcal{E}_{b'}, \mathbf{U}_{b'})$ which stores features of a part of vertices $\mathcal{V}_{b'} = \{1, ..., N_{b'}\}$ ($N_{b'}$ is

a part of N_b), edges $\mathcal{E}_{b'} \subseteq \mathcal{V}_{b'} \times \mathcal{V}_{b'}$, and the adjacency representation $\mathbf{U} \in [0,1]^{N_{b'} \times N_{b'}}$. Specifically, though indicator matrix $\boldsymbol{I}$, we found the $N_{b'}$ number of vertices from body which is related to garment ($N_{b'} = N_g$), and also apply the connection to edges $\mathcal{E}_{b'}$.

Based on the proposed graph construction method in Section 3.3.1, we assign features to the graph nodes. The feature vector can be represented as: $v_{b'i} = [r^T_{b'i}(\boldsymbol{\theta}), n^T_{b'i}, \alpha]$. For $r_{b'i}(\boldsymbol{\theta})$, we multiply the skinning weight $w_{s,b'i}$, the body joint rotation matrix $G_s(\boldsymbol{\theta})$, $G_s(\boldsymbol{\theta}^*)$, and the $\bar{r}_{b'i}$ by following Equation 3.9. Then, combining with the vertex normal $n_{b'i}$, and the fit degree α, to get the final mesh node presentation $v_{b'i} \in \mathbb{R}^9$.

Next, we input the graph into our coarse regressor W_{coarse}

$$W_{\text{coarse}}(\boldsymbol{\mathcal{G}}_{b'}) = \Delta_{\text{coarse}} \tag{4.5}$$

to approximate a vector of displacement from body to coarse garment $\Delta_{\text{coarse}} \in \mathbb{R}^{3 \times N_g}$. Then, the predicted coarse garment M_{coarse} is computed by adding the vertex offsets onto the corresponding body mesh $M_{b'}$:

$$M_{\text{coarse}} = M_{b'} + \Delta_{\text{coarse}}. \tag{4.6}$$

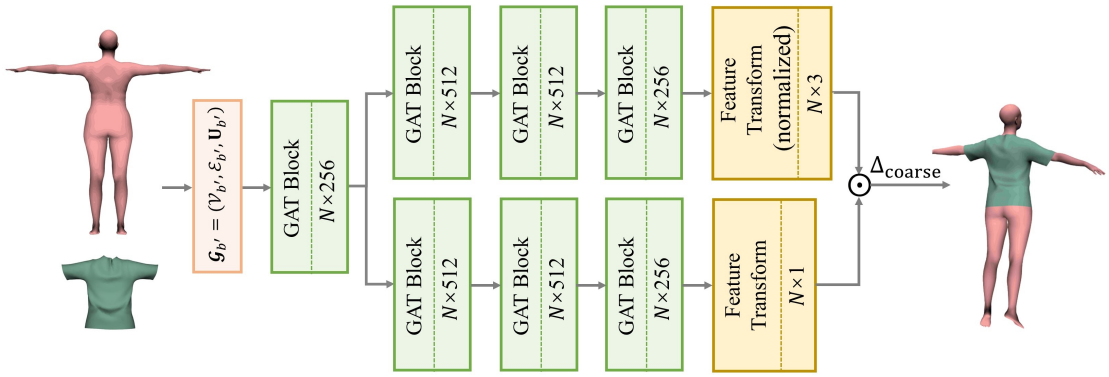


Figure 4.4: Outline of approximating the coarse deformation.

The whole process of generating the coarse deformation can be shown in Figure 4.4. Note that, we adopt the output decomposition way for accurately approximating the garment coarse deformation. The detail of the output decomposition will be described in Section 4.3.2. To train the regressor W_{coarse}, we create a dataset of ground-truth deformations of garments with different sizes using physics-based deformation [61] on

different SMPL body shapes. To obtain the coarse data, we apply a Laplacian smoothing operator to each generated garment meshes.

4.3 Refinement through GarFitNet

After the first step of generating garment coarse deformations M_{coarse} due to target body shape, pose, and fit parameters, we next add fine details on the roughly deformed meshes. The reasons for designing two steps rather than directly predicting final deformations have two folds: first, it eases the learning task since it reduces the variance in data. Direct mapping from the given parameters to the final result, *i.e.*, draping garments on bodies and estimating the precise deformation, is too difficult to guarantee the effective learning of the network. The training process will unavoidably suffer from the overfitting problem, and the trained network has poor generalization ability for approximating new objects. Second, it decouples target deformations (*i.e.*, global stretching and draping effects) from fine wrinkles deformations, which we will learn in a subsequent step.

In this section, we will describe how to refine the garment from M_{coarse} to M with our garment-fit network (GarFitNet).

4.3.1 Garment mesh encoding

Starting from the coarse garment mesh M_{coarse}, we first build a mesh graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{U})$ which stores features of vertices $\mathcal{V} = \{1, ..., N_g\}$, edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and the adjacency representation $\mathbf{U} \in [0,1]^{N_g \times N_g}$. We assign features to the graph nodes: $v_i = [r_i(\boldsymbol{\theta})^T, n_i^T, x_i^T]$. Here, $r_i(\boldsymbol{\theta})$ denotes the relative skinning features of garment and can be computed by:

$$r_i(\boldsymbol{\theta}) = r_{b'i}(\boldsymbol{\theta}) + \Delta_{\text{coarse,i}}. \tag{4.7}$$

where $r_{b'i}(\boldsymbol{\theta}) \in \mathbb{R}^3$ is computed in Section 5.2.2 following Equation 3.9. $\Delta_{\text{coarse,i}} \in \mathbb{R}^3$ is per vertex displacement. Together with the normal of vertex of the garment n_i and the distance vector from the vertex of garment to joints x_i, the dimension of the final feature vector $v_i = [r_i^T(\boldsymbol{\theta}), n_i^T, x_i^T]$ is also $6 + S$, where S is the number of joints. In this way, the garment graph features can contain all the geometry and pose information

available in context. We then take the constructed graph as the input and forward it into the graph-learning-based GarFitNet.

In addition to the constructed graph, we also input parameters (β, θ, α) to our GarFitNet. We call these parameters as GarFit parameters which are able to represent the body shape, pose, and fit degree between the body to the garment. The reason for such additional input is that only the graph of the coarse garment is insufficient to yield rich details. As we discussed before, the relation between garment and body plays a key role in detail generation. Therefore, GarFit parameters which provide the body information (β, θ) and the relationship information (α) are also forwarded into the GarFitNet.

4.3.2 Output reconstruction

Most deformation approximation studies are plagued by the problem of highly nonlinear output. Since the range of vertex offset adjustment is very wide, its value can range from negative infinity to positive infinity, studies can only increase the number of MLPs while sacrificing generalization [63] or predict for one pose [75] to ensure the quality of deformation approximation. So far, there is no research to solve the problem fundamentally from the perspective of optimizing output.

In our work, we propose an output reconstruction method by decomposing the output displacement vector $\Delta_i \in \mathbb{R}^3$ into the vector length $\Delta_i' \in \mathbb{R}^+$ and the unit vector $\delta_i \in \mathbb{R}^3$. Unlike other learning-based methods which directly predict Δ_i with a wide value range of $(-\infty, +\infty)$, our method indirectly predicts the vector length Δ_i' and the unit vector δ_i with the narrow value range of $(0, +\infty)$ and $(-1, 1)$. With the two approximated items of Δ_i and δ_i, we finally multiply them together to get the final nonlinear offset vector. This decomposition step does not seem complicated, however, it plays a crucial role that greatly eases the learning process since the value range of the output variable is largely reduced.

With the decomposed output, we design the loss function based on their value range:

$$\frac{1}{N}\sum_{i=1}^{N}\left\|\Delta_i'^{GT}\odot\delta_i^{GT}-\Delta_i'\odot\delta_i\right\|^2, \tag{4.8}$$

$$\Delta_i' = f_{relu}(v_i^{[l]}), \tag{4.9}$$

$$\delta_i = f_{norm}((f_{tanh}(g_{i,0}^{[l]}), f_{tanh}(g_{i,1}^{[l]}), f_{tanh}(g_{i,2}^{[l]}))), \tag{4.10}$$

where Δ_i^{GT} and δ_i^{GT} respectively denote the ground truth length and direction unit. f_{relu} is the relu activation function which can limit the value of output features $v_i^{[l]}$ (where $v_i^{[l]} \in \mathbb{R}$) in the length prediction branch to a range greater than zero. f_{tanh} is the tanh activation function which can limit the value of each dimension $((g_{i,0}^{[l]}), (g_{i,1}^{[l]}), (g_{i,2}^{[l]}))$ of output features $g_i^{[l]}$ in the unit prediction branch within the range of -1 to 1. Additionally, f_{norm} denotes the normalization operation ($f_{norm}(\cdot) = \frac{\cdot}{\|\cdot\|}$) which can further make the final result of δ_i normalized, so that the square root of the sum of the squared each dimensional values is equal to 1. In practice, we found the training becomes much easier to converge following the decomposed output and loss function than other methods using plain outputs.

4.3.3 GarFitNet architecture

Once input the graph $\mathcal{G}$ of M_{coarse} and optimized output are defined, we then describe how to build the relationship by proposing the novel graph neural network GarFitNet (shown in Figure 4.5).

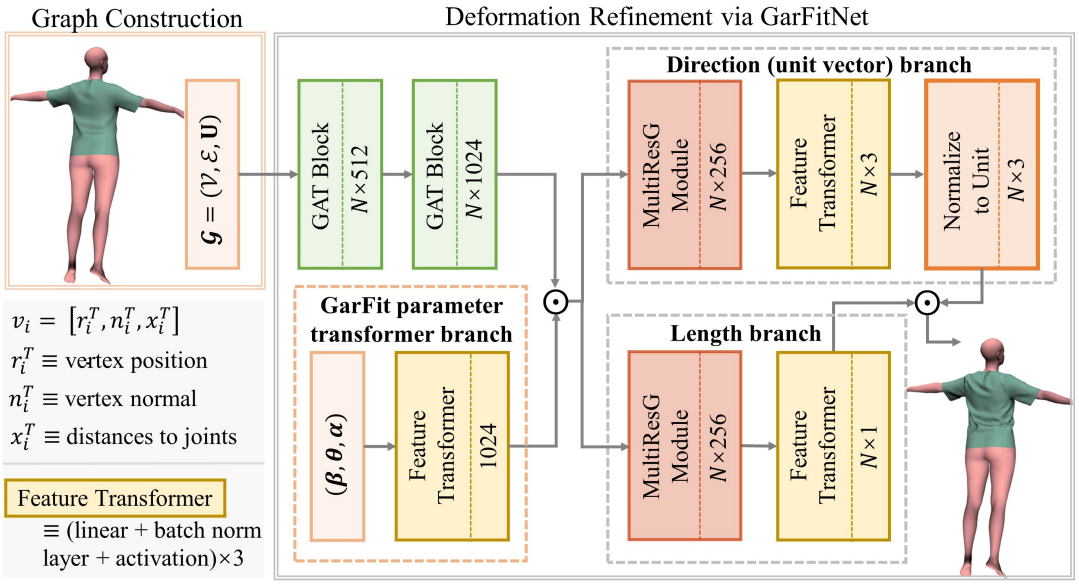


Figure 4.5: Overview of the proposed GarFitNet.

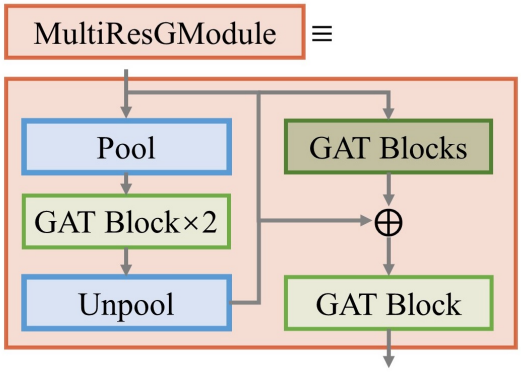


Figure 4.6: MultiResGModule is an integrated structure from MultiResGNet.

For the architecture of the designed GarFitNet, it consists of three branches, *i.e.*, the GarFit parameter transformer branch, the unit direction vector prediction branch, and the length prediction branch.

Although input garment mesh graphs have contained rich information of garment, pose, and distances to joints, it cannot express the fit degree between garment and body which we have explained (in Section 5.1) as an important factor that affects the final wrinkle effect. To this end, we design a transformer branch that is capable of transferring the GarFit parameters into high degree features as being the GarFit correction weight.

Next, we conduct Hadamard multiplication between this GarFit correction weight and the transferred graph features to obtain new features. In this way, graph features have gained greater representative capabilities with fit information after weighting with transferred GarFit parameters.

Based on the proposed output decomposition theory, we respectively design a unit direction vector prediction branch and a length prediction branch. The reason for designing them is that the value of two decomposed output elements differ greatly, as the unit value is from -1 to 1 and the length value is greater than 0.

Due to the nonlinearity of the output from these two prediction branches, we therefore apply the MultiResGNet idea to benefit from its powerful generalization and detail refinement capabilities. According to the garment features and the amount of data, we improve and integrate the MultiResGNet architecture into the MultiResGModule (shown in Figure 4.6) which has a similar structure as MultiResGNet to deal with original-resolution features and down-sampled lower resolution features respectively. The predicted unit vector and length are calculated separately in two prediction branches, where the normalized unit is additionally used in the direction prediction branch for features normalization. Finally, output features from two branches are multiplied as the final offset adjustment per vertex. Our garment after refinement can be expressed as:

$$M = M_{\text{coarse}} + \Delta, \tag{4.11}$$

$$\Delta = \Delta' \odot \delta = W(\boldsymbol{G}, \beta, \theta, \alpha; \Psi), \tag{4.12}$$

where $\Delta \in \mathbb{R}^{N_g \times 3}$ is automatically approximated by training GarFitNet regressor $W(\cdot)$. The trainable weights in the network are independent of the number of garment mesh vertices, which makes it possible to generalize to new garments with different mesh topologies. Thanks to our two observations and the designed network, various garments worn by diverse body shapes under any pose can be well predicted with realistic visual effects and highly nonlinear details.

4.4 Evaluation

4.4.1 Dataset

To evaluate our proposed method, we create the dataset for training and testing. Our dataset contains both character bodies and garments. In the training set, for different body shapes, we obtain the data from the SMPL body model [52] and generate eight types of the body shape β to represent different body weights as shown in Figure 4.7. For the garment data, we utilize the commercial 3D cloth design and simulation tool Marvelous Designer [2] to design and simulate the garments. As shown in Figure 4.8, we design six types of garments with different lengths and different topologies. In total, there are 48 garment-body pairs with the different fit degrees and garment lengths in the dataset. For pose variations, we created 1525 poses with five motion sequences, including dancing, walking, jumping, raising hands, and front kicking. To verify the effectiveness of the GarFitNet, we use three new body shapes and three new garments for testing.

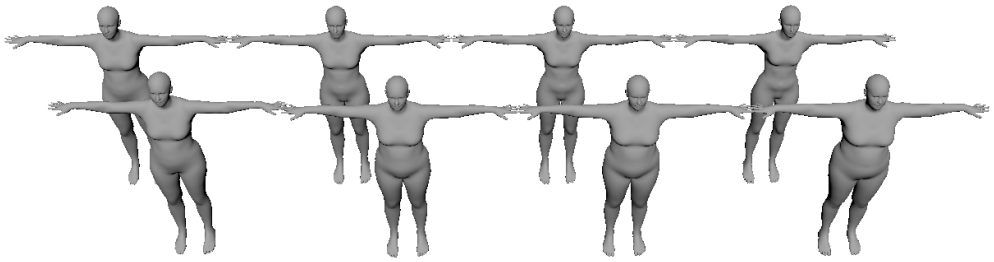
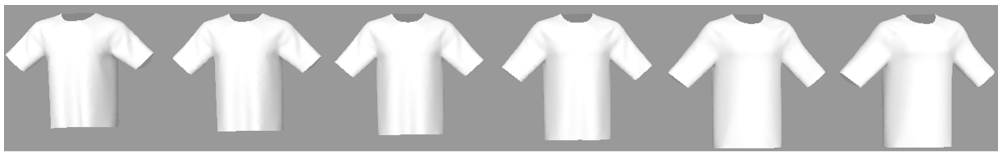


Figure 4.7: Bodies with different weights



$N = 3178$ 3311 3487 3666 3987 4002

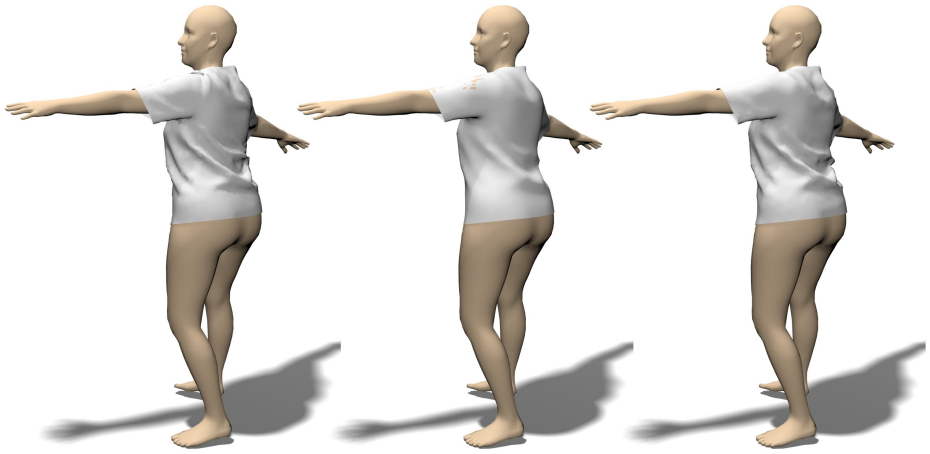
Figure 4.8: Garments with different length and they have different number of vertices.

4.4.2 Implementation details

As depicted in Figure 4.5, the graph features are first normalized and input into two GAT blocks for the feature transformation. Each GAT block involves a graph attention-based aggregation stream with the hidden feature size of 64, the multi-head number is 4, and a self-reinforced stream with hidden feature size of 256. Features after each GAT block will be transferred with the tanh activation function. For the GarFit parameter transformer branch, it consists of three hidden layers with hidden neurons of (85, 256, 512) and is applied with batch normalization and relu activation function. The dimension of transferred parameters from the transformer branch becomes 1024, and they are multiplied with output features from two GAT blocks to obtain the weighted features which contain the information of fit degree, shape, and pose. These features are then respectively forward into the direction prediction branch and the length prediction branch. For the MultiResGModule, pooling and unpooling operations play the role of globally summarize features and follow the same definition as mentioned in MultiResGNet. In the local branch of MultiResGModule, there are four GAT blocks with two multi-head and 64 hidden feature sizes of two streams. Finally, output from both the direction prediction branch and length prediction branch will forward into the feature transformation module, and follow the tanh activation function or ReLu activation function respectively. In addition, the unit vector with three-dimensional features will be normalized at last.

4.4.3 Accuracy results

Qualitatively, as shown in Figure 4.9, based on the coarse deformation, our method is able to produce as many wrinkles as ground truth data. All fine details can be well retained by using our GarFitNet despite being test on the unseen garment and pose.



(a) (b) (c)

Figure 4.9: Test on a character performing a dancing pose. (a) is the ground truth. (b) is the coarse deformation. (c) is our prediction through GarFitNet.

Next, in Figure 4.10, we provide the result of our method for a variety of body shapes, which do not exist in the training set. The fine-scale wrinkles predicted by our GarFitNet naturally match the expected behavior of the garment and have no obvious difference with ground truth data. In addition, the degree of wrinkles follows the law of our first observation that loose-fitting wear (thin body) produces sparse wrinkles with wider width whereas tight-fitting (fat body) produces denser wrinkles with a narrower width. The approximation results demonstrate that our method generalizes well to new body shapes with the new fit degree.

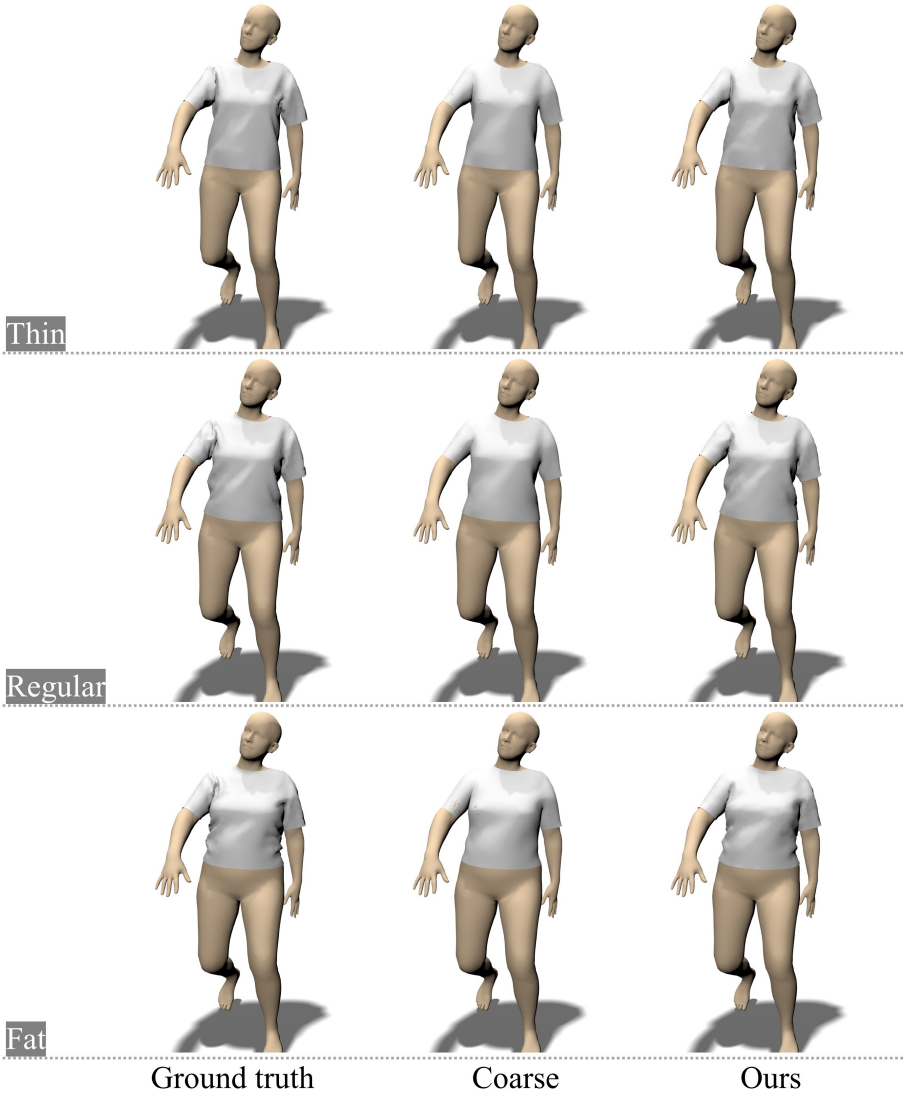


Figure 4.10: Deformations of different body shapes: thin, regular, and fat.

In Table 4.1, we report the quantitative result of garments dressed on bodies in Figure 4.10 under the walking motion. As shown in the table, the accuracy of the result after deformation refinement through GarFitNet can be improved by nearly 50% compared to the rough deformation. The deformation prediction error of garments worn by a thin body is relatively smallest because the garment folds tend to be smoother and simpler, while the error for a fat body is relatively larger due to the complexity of the folds.

Table 4.1: Mean error (cm) of per vertex of deformations in different body shapes (corresponding to Figure 4.10).

Shape types	Coarse	Refinement
Thin	0.2817	0.1426
Regular	0.3010	0.1523
Fat	0.3268	0.1831

In Figure 4.11, we show the qualitative deformation approximations of our method for various garments with different lengths. Here, we test our GarFitNet by using garments with unseen mesh topologies and vertices number at train time. The results demonstrate that our method has the generalization abilities to approximate deformations for arbitrary topology garment mesh with fine details. Then, in Table 4.2, we further provide the quantitative results of characters dressed by the three types of garments performing the jumping motion. The mean error of long garments tends to be relatively largest because the length of the garment is long and the deformation will also be affected by the movement of the legs to produce more wrinkles than in other cases.

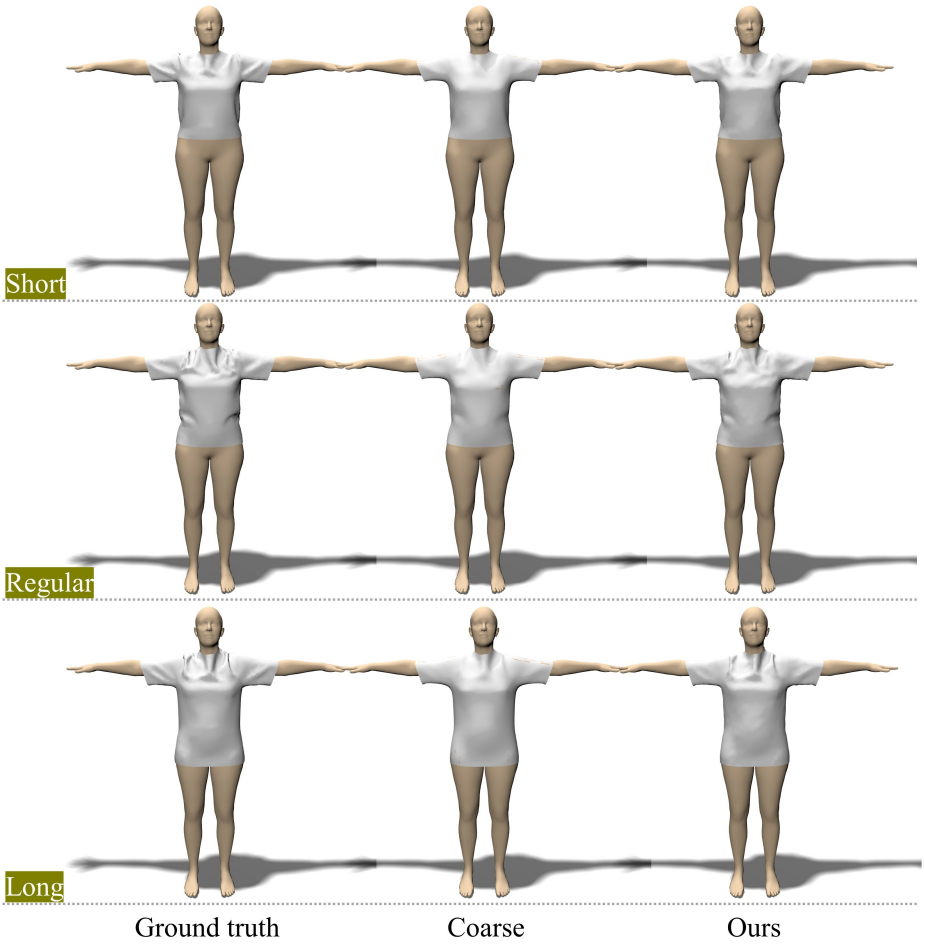


Figure 4.11: Deformations of different garment length: short, regular, and long.

Table 4.2: Mean error (cm) of per vertex of deformations for different garments (corresponding to Figure 4.11).

Garment types	Coarse	Refinement
Short	0.3217	0.1491
Regular	0.3575	0.1660
Long	0.3931	0.1862

4.4.4 Comparison

We conduct an ablation study to highlight the influence of the proposed GarFit parameter transformer branch and the proposed output decomposition operation. Specifically, we

trained the network by individually removing the GarFit parameter transformer branch in the GarFitNet, and remaining the output as three-dimensional displacement vector without decomposition. As shown in Figure 4.12, sub-figures in (c) and (d) have no obvious wrinkles and just have tiny improvement than coarse deformation (b) in the areas of the waist on both sides. In contrast, our proposed method with the transformer branch and the output optimization can significantly help to generate more plausible wrinkles.

(a) (b) (c) (d) (e)

Figure 4.12: Comparison between (a) ground truth, (b) coarse deformation, (c) ours without GarFit parameter transformer branch, (d) ours without output decomposition, and (e) ours followed mentioned setting.

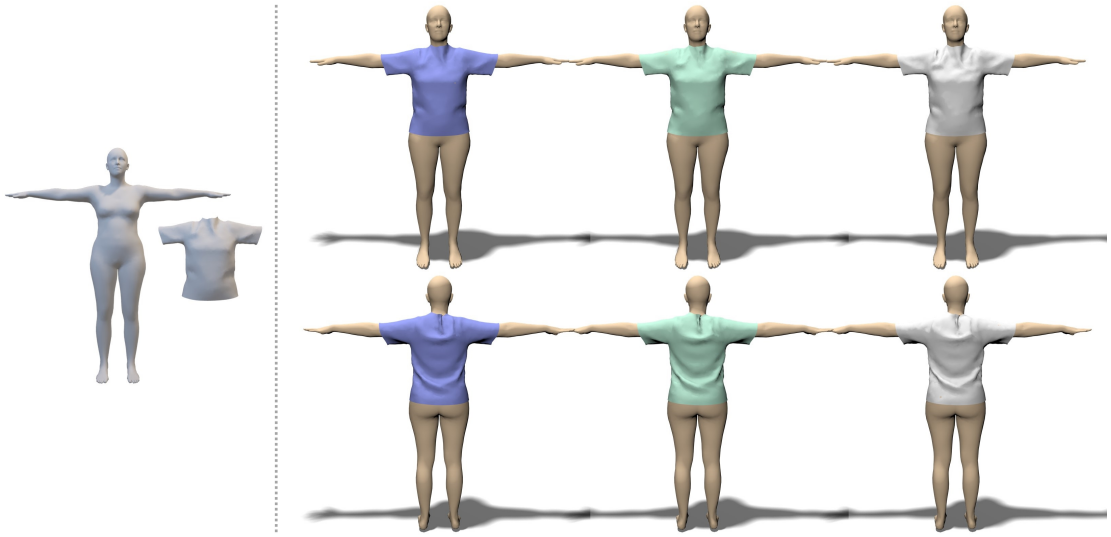
Next, we compare our method to state-of-art learning-based methods. As listed in Table 4.3, our method outperforms others that is able to use a fewer number of models to approximate deformations for arbitrary mesh geometry, pose, and fit degree. To our best knowledge, ours is the first graph-learning-based approach for garment deformation prediction with high-quality pose-fit-dependent detail effects.

Table 4.3: Comparison of our method with other state-of-art learning-based methods. Our method can achieve more functions with fewer models.

Methods	Model number	Geometry variation	Pose variation	Fit/Style variation
Santesteban *et al.* [68]	2	✗	✓	✗
TailorNet [63]	22	✗	✓	✓
FCGNN [75]	3	✓	✗	✓
Ours	2	✓	✓	✓

In Figure 4.13, we qualitatively compare our method with other approaches of FCGNN [75] and TailorNet [63]. Specifically, FCGNN (sub-figure(a) with the blue

t-shirt) also uses graph neural networks to predict the deformation of various garments dressed by arbitrary target body shape. However, it is limited to produce pose-dependent effects and is only able to predict t-pose deformations. The details of (a) are lacking especially in the areas of the collar. For the method of TailorNet (sub-figure(b) with the green t-shirt), the deformation quality is visually better than (a) with richer wrinkles. But the method cannot generalize to other mesh topologies and the training for one specific garment requires a large number of MLPs. In contrast, our method in (c) shows the realistic behaviors of the garment with complicated fine-scale wrinkles and folds. Furthermore, our GarFitNet allows for generalization to arbitrary geometry and body, and provides a richer space of pose effects.



(a) (b) (c)

Figure 4.13: Comparison between (a) FCGNN [75], (b) TailorNet [63], and (c) our prediction.

4.5 Summary of the of the 2^{nd} Case Deformation Methods

In this chapter, we have presented a novel graph-learning-based method GarFitNet for automatically generating rich detailed deformations for diverse garments worn by any body shape in any pose. We address limitations of previous work including overly smooth results and the inability of generalization through three main contributions.

To account for complicated deformations, we first propose the fit parameter that has an important impact on deformation wrinkles and take it as one of the inputs of the network. Then, to make the learning easy while ensuring high-quality output, we narrow the range of output by proposing the novel output reconstruction strategy. Also, with improved input and output, to make the network have the strong generalization ability, we introduce GarFitNet architecture with three branches. Thanks to these technical contributions, our method is superior to other learning-based approaches and is the first graph-learning-based approach to successfully produce high-quality deformations for unseen garments, bodies, and poses.

Despite the step forward in fast and accurate clothing deformation, our method also has the following weakness. First, all garments in our dataset have the same material setting and their deformations are dependent on fit, shape, and pose. Since material specification can also influence the deformation effect, if deformations with different clothing materials need to be approximated using our method, it requires training the network independently for each material. Second, we use the popular SMPL model to represent character bodies of various shapes in our research. However, the SMPL bodies do not have the appearance of muscles of different body parts. If the dressing effect of clothes wants to be perfectly closer to the effect of real people trying on it, exploring and developing a new character dataset, and then studying the deformation of clothes based on this dataset is the next interesting study.

Chapter 5

Conclusion and Future work

5.1 Conclusion

We have presented unified deformation systems for rapidly and accurately approximating nonlinear deformations for diverse characters in various poses. Compared with traditional geometry-based and physics-based methods, our methods provide an efficient way of making use of existing well-designed features of characters for unseen animated characters while ensuring high accuracy and real-time performance.

To achieve deformations for animated characters in different applications, we first declare two deformation cases to meet the quality requirements in different animation productions. For these two cases, in order to make the deformation learnable, we respectively create the dataset with various deformations of characters and garments in possible poses.

Technically, for the first case deformation, we successively propose DenseGATs and MultiResGNet to achieve nonlinear deformations for animated characters. DenseGATs pioneered the prediction of pose-dependent deformation based on graphs and make improvements on the network architecture. The proposed GAT blocks and dense modules allow for efficient utilization and transmission of self and adjacent information throughout the network. Based on DenseGATs, to make the network have better generalization ability, our MultiResGNet approach makes further improvements on both the input side and the network side. By designing expressive graph features and proposing the multi-resolution graph processing strategy, the method can yield more accurate deformation

results with fewer training samples. Next, for the second case deformation, we propose GarFitNet for animated characters emphasizing more clothing details. Our work successfully addresses the limitations in previous work involving overly smooth results and the inability of generalization. The proposal of fit parameter, output reconstruction, and three-branch GarFitNet architecture enables us to achieve high-quality deformations with rich wrinkle effects for unseen garments, bodies, and poses.

All in all, our DenseGATs, MultiResGNet, and GarFitNet provide end-to-end deformation approximation systems, which allow the application to the game, film productions, and virtual try-on. Using our systems, artists can directly produce a large collection of characters with complicated deformation effects through the trained network, thus avoiding troublesome manual processing.

5.2 Limitations

Although our presented approaches have a strong generalization ability and yield high accuracy prediction results, they also have a few drawbacks. For the 1^{st} deformation case, first, the learned network is dependent on the setting of linear-based deformation, such as the skinning method, the number of bone influences, the binding distance, etc. Thus, when generating the nonlinear deformation for a new character, the setting of linear-based deformation should be modified to keep it consistent with the training data. Secondly, in some cases where characters wear tight clothing, the produced deformation results may have collision artefacts between the body and clothing. Thirdly, the training samples we used mostly follow the body closely. For extremely loose clothing and other garment types like skirts, the deformation would largely depend on the diversity of training samples. If the test character is dramatically different from the training characters, the trained model may fail to predict the accurate deformation. Additionally, we currently focus on the deformation for humanoid characters which have the same standard skeleton, and the dimension of input graph features is dependent on the number of skeletal joints. Ideally, two-limbed characters with the same number of joints and with the similar appearance to the training samples can use our trained network to achieve the deformation approximation. But for four-limbed animals or characters with

different joint counts, the network needs to be retrained.

For the 2^{nd} deformation case, first, while our deformations have contained garments with arbitrary geometry, the garment material-dependent parameter is not considered in our approach. While deforming new garments with different materials, using our method requires retraining a new model for the specific material. Furthermore, our method utilizes the existing SMPL bodies which are popular to represent a wide range of human bodies, and this SMPL model is limited to express the appearance of muscles. Therefore, the deformation of wearing clothes on the SMPL body by our method cannot have the effect of muscles in real characters.

5.3 Future Work

For the 1^{st} deformation case, increasing the diversity of garments and add situations where characters fit differently with garments in training samples will be meaningful for actual animation needs in the future. Second, deformation research can be explored for various types of objects (*e.g.*, animals with two or four limbs). If the deformed objects have a various number of joints, a union of joints can be defined to represent all the conditions in the dataset, so as to keep the feature dimension of each object the same.

For the 2^{nd} deformation case, a larger garment material space remains open for future research. Then, it would also be interesting to explore the additional material input and extend our method to handle deformations between multiple materials of clothing. Next, to get closer to the real visual effect, the wider human dataset which is factorized of muscle mass for different body parts can be created. Then, more realistic contact effects between bodies and clothes can be produced by using data-driven approaches.

Appendix A

We provide the supplementary video to show our result of DenseGATs, MultiResGNet, and GarFitNet: https://www.youtube.com/watch?v=giAINShfGHE

Bibliography

[1] Adobe fuse cc. https://www.adobe.com/wam/fuse.html. Accessed: 2021.

[2] Marvelous designer. https://www.marvelousdesigner.com/. Accessed: 2021.

[3] Mixamo auto-rigger. https://www.mixamo.com. Accessed: 2021.

[4] Optitex. https://optitex.com/. Accessed: 2021.

[5] Tuka3d. https://tukatech.com/tuka3d/. Accessed: 2021.

[6] Turbosquid. https://www.turbosquid.com/. Accessed: 2021.

[7] Nesreen K. Ahmed, Ryan A. Rossi, Rong Zhou, John Boaz Lee, Xiangnan Kong, Theodore L. Willke, and Hoda Eldardiry. Inductive representation learning in large attributed graphs, 2017.

[8] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. *ACM Trans. Graph.*, 21(3):612–619, July 2002.

[9] Stephen W. Bailey, Dave Otte, Paul Dilorenzo, and James F. O'Brien. Fast and deep deformation approximations. *ACM Trans. Graph.*, 37(4):119:1–119:12, July 2018.

[10] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, 26(3):72–es, July 2007.

[11] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[12] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33(4), July 2014.

[13] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs, 2013.

[14] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph.*, 21(3):586–593, July 2002.

[15] Dan Casas and Miguel A. Otaduy. Learning nonlinear soft-tissue dynamics for interactive avatars. *Proc. ACM Comput. Graph. Interact. Tech.*, 1(1), July 2018.

[16] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. *SIGGRAPH Comput. Graph.*, 23(3):243–252, July 1989.

[17] Jianlong Chang, Jie Gu, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Structure-aware convolutional neural networks. *Advances in neural information processing systems*, 31:11–20, 2018.

[18] Nuttapong Chentanez, Miles Macklin, Matthias Müller, Stefan Jeschke, and Tae-Yong Kim. Cloth and skin deformation with a triangle mesh based convolutional neural network. SCA '20, Goslar, DEU, 2020. Eurographics Association.

[19] Dennis Child. *The essentials of factor analysis*. Cassell Educational, 1990.

[20] Gabriel Cirio, Jorge Lopez-Moreno, and Miguel A. Otaduy. Efficient simulation of knitted cloth using persistent contacts. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '15, page 55–61, New York, NY, USA, 2015. Association for Computing Machinery.

[21] Edilson de Aguiar, Leonid Sigal, Adrien Treuille, and Jessica K. Hodgins. Stable spaces for real-time clothing. *ACM Trans. Graph.*, 29(4), July 2010.

[22] Olivier Dionne and Martin de Lasa. Geodesic voxel binding for production character meshes. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 173–180. ACM Press, New York, NY, USA, 2013.

[23] Olivier Dionne and Martin de Lasa. Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, page 173–180, New York, NY, USA, 2013. Association for Computing Machinery.

[24] Hongyang Gao and Shuiwang Ji. Graph U-Nets, 2019.

[25] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1416–1424, New York, NY, USA, 2018. ACM Press.

[26] Russell Gillette, Craig Peters, Nicholas Vining, Essex Edwards, and Alla Sheffer. Real-time dynamic wrinkling of coarse animated cloth. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '15, page 17–26, New York, NY, USA, 2015. Association for Computing Machinery.

[27] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, March 2011.

[28] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data, 2015.

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

[30] Haoda Huang, Ling Zhao, KangKang Yin, Yue Qi, Yizhou Yu, and Xin Tong. Controllable hand deformation from sparse examples with rich details. SCA '11, page 73–82, New York, NY, USA, 2011. Association for Computing Machinery.

[31] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4), July 2011.

[32] Jonathan M. Kaldor, Doug L. James, and Steve Marschner. Efficient yarn-based cloth with adaptive contact linearization. *ACM Trans. Graph.*, 29(4), July 2010.

[33] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. I3D '07, page 39–46, New York, NY, USA, 2007. Association for Computing Machinery.

[34] Ladislav Kavan, Dan Gerszewski, Adam W. Bargteil, and Peter-Pike Sloan. Physics-inspired upsampling for cloth simulation in games. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, New York, NY, USA, 2011. Association for Computing Machinery.

[35] Ladislav Kavan and Olga Sorkine. Elasticity-inspired deformers for character articulation. *ACM Trans. Graph.*, 31(6), November 2012.

[36] Tae-Yong Kim, Nuttapong Chentanez, and Matthias Müller-Fischer. Long range attachments - a method to simulate inextensible clothing in computer games. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, page 305–310, Goslar, DEU, 2012. Eurographics Association.

[37] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[38] Martin Komaritzan and Mario Botsch. Projective skinning. *Proc. ACM Comput. Graph. Interact. Tech.*, 1(1), July 2018.

[39] Tsuneya Kurihara and Natsuki Miyata. Modeling deformable human hands from medical images. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, page 355–363, Goslar, DEU, 2004. Eurographics Association.

[40] Caroline Larboulette, Marie-Paule Cani, and Bruno Arnaldi. Dynamic skinning: Adding real-time dynamic effects to an existing character animation. In *Proceedings of the 21st Spring Conference on Computer Graphics*, SCCG '05, page 87–93, New York, NY, USA, 2005. Association for Computing Machinery.

[41] Binh Huy Le and Zhigang Deng. Robust and accurate skeletal rigging from mesh sequences. *ACM Trans. Graph.*, 33(4):84:1–84:10, July 2014.

[42] Gene S. Lee, Andy Lin, Matt Schiller, Scott Peters, Mark McLaughlin, and Frank Hanner. Enhanced dual quaternion skinning for production use. In *ACM SIGGRAPH 2013 Talks*, SIGGRAPH '13, New York, NY, USA, 2013. Association for Computing Machinery.

[43] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling, 2019.

[44] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proc. SIGGRAPH '00*, pages 165–172. ACM Press, New York, NY, USA, 2000.

[45] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, Washington, DC, USA, July 2017. IEEE Computer Society.

[46] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. Can gcns go as deep as cnns? 2019.

[47] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proc. 32nd AAAI Conference on Artificial Intelligence*, pages 3546–3553, Palo Alto, CA, 2018. AAAI Press.

[48] Tianxing Li, Rui Shi, and Takashi Kanai. DenseGATs: A graph-attention-based network for nonlinear character deformation. In *Proc. Symposium on Interactive 3D Graphics and Games*. ACM Press, New York, NY, USA, 2020.

[49] Tianxing Li, Rui Shi, and Takashi Kanai. MultiResGNet: Approximating Nonlinear Deformation via Multi-Resolution Graphs. *Computer Graphics Forum*, 2021.

[50] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks, 2017.

[51] Lijuan Liu, Youyi Zheng, Di Tang, Yi Yuan, Changjie Fan, and Kun Zhou. Neuroskinning: Automatic skin binding for production characters with deep graph networks. *ACM Trans. Graph.*, 38(4):114:1–114:12, July 2019.

[52] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6), October 2015.

[53] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[54] Nadia Magnenat-Thalmann, Richard Laperrière, and Daniel Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics Interface '88*, pages 26–33. Canadian Information Processing Society, Toronto, Ont., Canada, 1988.

[55] Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph.*, 30(4):37:1–37:12, July 2011.

[56] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5115–5124, Washington, DC, USA, July 2017. IEEE Computer Society.

[57] Tomohiko Mukai. Building helper bone rigs from examples. In *Proc. Symposium on Interactive 3D Graphics and Games*, pages 77–84. ACM Press, New York, NY, USA, 2015.

[58] Tomohiko Mukai and Shigeru Kuriyama. Efficient dynamic skinning with low-rank helper bone controllers. *ACM Trans. Graph.*, 35(4), July 2016.

[59] Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. Strain based dynamics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '14, page 149–157, Goslar, DEU, 2015. Eurographics Association.

[60] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *J. Vis. Comun. Image Represent.*, 18(2):109–118, April 2007.

[61] Rahul Narain, Armin Samii, and James F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.*, 31(6), November 2012.

[62] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. In *Computer graphics forum*, volume 25, pages 809–836. Wiley Online Library, 2006.

[63] Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020.

[64] Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. Semi-supervised user geolocation via graph convolutional networks. *CoRR*, abs/1804.08049, 2018.

[65] T. Rhee, J. P. Lewis, and U. Neumann. Real-time weighted pose-space deformation on the gpu. *Computer Graphics Forum*, 25, 2006.

[66] Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. Animation wrinkling: Augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Trans. Graph.*, 29(6), December 2010.

[67] Igor Santesteban, Elena Garces, Miguel A. Otaduy, and Dan Casas. SoftSMPL: Data-driven modeling of nonlinear soft-tissue dynamics for parametric humans. *Computer Graphics Forum (Eurographics 2020)*, 39(2):65–75, 2020.

[68] Igor Santesteban, Miguel A. Otaduy, and Dan Casas. Learning-Based Animation of Clothing for Virtual Try-On. *Computer Graphics Forum (Proc. Eurographics)*, 2019.

[69] Xiaohan Shi, Kun Zhou, Yiying Tong, Mathieu Desbrun, Hujun Bao, and Baining Guo. Example-based dynamic skinning in real time. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, New York, NY, USA, 2008. Association for Computing Machinery.

[70] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics.

[71] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 205–214, New York, NY, USA, 1987. Association for Computing Machinery.

[72] Garvita Tiwari, Bharat Lal Bhatnagar, Tony Tung, and Gerard Pons-Moll. Sizer: A dataset and model for parsing 3d clothing and learning size sensitive 3d clothing. In *European Conference on Computer Vision (ECCV)*. Springer, August 2020.

[73] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 5998–6008, USA, 2017. Curran Associates Inc.

[74] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks, 2017.

[75] Raquel Vidaurre, Igor Santesteban, Elena Garces, and Dan Casas. Fully Convolutional Graph Neural Networks for Parametric Virtual Try-On. *Computer Graphics Forum (Proc. SCA)*, 2020.

[76] Xiaohuan Corina Wang and Cary Phillips. Multi-weight enveloping: Least-squares approximation techniques for skin animation. SCA '02, page 129–138, New York, NY, USA, 2002. Association for Computing Machinery.

[77] Rich Wareham and Joan Lasenby. Bone glow: An improved method for the assignment of weights for mesh deformation. In *Proceedings of the 5th International Conference on Articulated Motion and Deformable Objects*, AMDO '08, page 63–71, Berlin, Heidelberg, 2008. Springer-Verlag.

[78] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. RigNet: Neural rigging for articulated characters. *ACM Trans. Graph.*, 39(4), 2020.

[79] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling, 2018.

[80] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications, 2018.

[81] J. S. Zurdo, J. P. Brito, and M. A. Otaduy. Animating wrinkles by example on non-skinned cloth. *IEEE Transactions on Visualization and Computer Graphics*, 19(1):149–158, 2013.