Real-time Deep Dynamic Characters

MARC HABERMANN, LINGJIE LIU, Max Planck Institute for Informatics
WEIPENG XU, MICHAEL ZOLLHOEFER, Facebook Reality Labs
GERARD PONS-MOLL, AND CHRISTIAN THEOBALT, Max Planck Institute for Informatics

We propose a deep videorealistic 3D human character model displaying highly realistic shape, motion, and dynamic appearance learned in a new weakly supervised way from multi-view imagery. In contrast to previous work, our controllable 3D character displays dynamics, e.g., the swing of the skirt, dependent on skeletal body motion in an efficient data-driven way, without requiring complex physics simulation. Our character model also features a learned dynamic texture model that accounts for photo-realistic motion-dependent appearance details, as well as view-dependent lighting effects. During training, we do not need to resort to difficult dynamic 3D capture of the human; instead we can train our model entirely from multi-view video in a weakly supervised manner. To this end, we propose a parametric and differentiable character representation which allows us to model coarse and fine dynamic deformations, e.g., garment wrinkles, as explicit space-time coherent mesh geometry that is augmented with high-quality dynamic textures dependent on motion and view point. As input to the model, only an arbitrary 3D skeleton motion is required, making it directly compatible with the established 3D animation pipeline. We use a novel graph convolutional network architecture to enable motion-dependent deformation learning of body and clothing, including dynamics, and a neural generative dynamic texture model creates corresponding dynamic texture maps. We show that by merely providing new skeletal motions, our model creates motion-dependent surface deformations, physically plausible dynamic clothing deformations, as well as video-realistic surface textures at a much higher level of detail than previous state of the art approaches, and even in real-time.

CCS Concepts: • **Computing methodologies** → **Motion capture**; *Motion capture*; *Mesh geometry models*.

Additional Key Words and Phrases: human modeling, human performance capture, deep learning, non-rigid surface tracking

ACM Reference Format:
Marc Habermann, Lingjie Liu, Weipeng Xu, Michael Zollhoefer, and Gerard Pons-Moll, and Christian Theobalt. 2020. Real-time Deep Dynamic Characters. *ACM Trans. Graph.* 39, 4, Article 94 (July 2020), 16 pages. https://doi.org/10.1145/3450626.3459749

1 INTRODUCTION

Animatable and photo-realistic virtual 3D characters are of enormous importance nowadays. With the rise of computer graphics in movies, games, telepresence, and many other areas, 3D virtual characters are everywhere. Recent developments in virtual and augmented reality and the resulting immersive experience further boosted the need for virtual characters as they now become part of our real lives. However, generating realistic characters still requires manual intervention, expensive equipment, and the resulting characters are either difficult to control or not realistic. Therefore, our goal is to learn digital characters which are both realistic and easy to control and can be learned directly from a multi-view video.

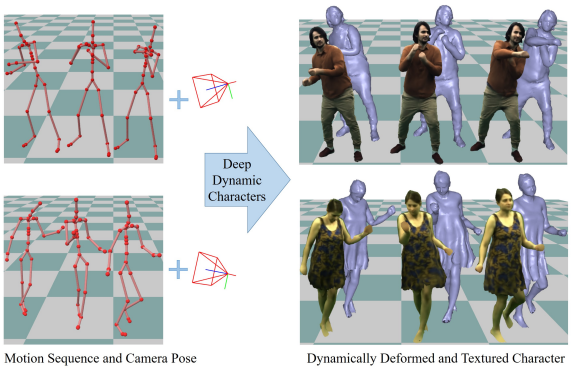


Fig. 1. Our learning-based method takes a sequence of poses and regresses the motion- and view-dependent dynamic surface deformation and texture of a person-specific template which looks video realistic.

It is a complicated process to synthesize realistic looking images of deforming characters following the conventional computer graphics pipeline. The static geometry of real humans is typically represented with a mesh obtained with 3D scanners. In order to pose or animate the mesh, a skeleton has to be attached to the geometry, i.e. rigging, and skinning techniques can then be used to deform the mesh according to the skeletal motion. While these approaches are easy to control and efficient, they lack realism as the non-rigid deformations of clothing are not modeled, e.g., the swinging of a skirt. While physics simulation can address this, it requires expert knowledge as it is hard to control. Further, these techniques are either computationally expensive or not robust to very articulated poses leading to glitches in the geometry. Finally, expensive physically based rendering techniques are needed to render realistic images of the 3D character. Those techniques are not only time consuming, but also require expert knowledge and manual parameter tuning.

To model clothing deformations, recent work combines classical skinning with a learnt mapping from skeletal motion to non-rigid deformations, and learns the model from data. One line of work learns from real data, but results either lack realism [Ma et al. 2020] or are limited to partial clothing, e.g., a T-shirt [Lähner et al. 2018]. More importantly, as they rely on ground truth registered 3D geometry, they require expensive 3D scanners and challenging template

Authors' addresses: Marc Habermann, Lingjie Liu, Max Planck Institute for Informatics, Campus E1, Stuhlsatzenhausweg 4, Saarbruecken, Saarland, Germany, 66123, mhaberma@mpi-inf.mpg.de, lliu@mpi-inf.mpg.de; Weipeng Xu, Michael Zollhoefer, Facebook Reality Labs, District Fifteen: 131 15th Street, Pittsburgh, Pennsylvania, USA, 15222, wxu@mpi-inf.mpg.de, zollhoefer@fb.de; Gerard Pons-Moll, and Christian Theobalt, Max Planck Institute for Informatics, Campus E1, Stuhlsatzenhausweg 4, Saarbruecken, Saarland, Germany, 66123, gpons@mpi-inf.mpg.de, theobalt@mpi-inf.mpg.de.

0730-0301/2020/7-ART94
https://doi.org/10.1145/3450626.3459749

registration. Another line of work tries to learn from a large database of simulated clothing [Guan et al. 2012; Patel et al. 2020]. While they can generalize across clothing categories and achieve faster run-times than physics simulations, the realism is still limited by the physics engine used for training data generation.

Furthermore, the texture dynamics are not captured by the aforementioned methods, although they are crucial to achieve photo-realism. Monocular neural rendering approaches [Chan et al. 2019; Liu et al. 2020b, 2019b] for humans learn a mapping from a CG rendering to a photo-realistic image, but their results have limited resolution and quality and struggle with consistency when changing pose and viewpoint. The most related works [Casas et al. 2014; Shysheya et al. 2019; Xu et al. 2011] are the ones leveraging multi-view imagery for creating animatable characters. However, all of them are not modeling a motion-dependent deforming geometry and/or view-dependent appearance changes.

To overcome the limitations of traditional skinning, the requirement of direct 3D supervision of recent learning based methods, as well as their lack of dynamic textures, we propose a learning based method that predicts the non-rigid character surface deformation of the full human body as well as a dynamic texture from skeletal motion *using only weak supervision in the form of multi-view images during training*. At the core of our method is a differentiable character (with neural networks parameterizing dynamic textures and non-rigid deformations) which can generate images differentiable with respect to its parameters. This allows us to train directly with multi-view image supervision using analysis by synthesis and back-propagation, instead of pre-computing 3D mesh registrations, which is difficult, tedious and prone to error. In designing differentiable characters, our key insight is to learn as much of the deformation as possible in geometry space, and produce the subtle fine details in texture space. Compared to learning geometry deformations in image space, this results in much more coherent results when changing viewpoint and pose. To this end, we propose a novel graph convolutional network architecture which takes a temporal motion encoding and predicts the surface deformation in a coarse to fine manner using our new fully differentiable character representation. The learned non-rigid deformation and dynamic texture not only account for dynamic clothing effects such as the swinging of a skirt caused by the actor's motion, or fine wrinkles appearing in the texture, but also fixes traditional skinning artifacts such as candy wrappers. Moreover, as our dynamic texture is conditioned on the camera pose, our approach can also model view-dependent effects, e.g., specular surface reflections. In summary, our contributions are:

- The first learning based real-time approach that takes a motion and camera pose as input and predicts the motion-dependent surface deformation and motion- and view-dependent texture for the full human body using direct image supervision.
- A differentiable 3D character representation which can be trained from coarse to fine (Sec. 3.1).
- A graph convolutional architecture allowing to formulate the learning problem as a graph-to-graph translation (Sec. 3.3).
- We collected a new benchmark dataset, called *DynaCap*, containing 5 actors captured with a dense multi-view system which we will make publicly available (Sec. 4.1).

Our dynamic characters can be driven either by motion capture approaches or by interactive editing of the underlying skeleton. This enables many exciting applications in gaming and movies, such as more realistic character control as the character deformation and texture will account for dynamic effects. Our qualitative and quantitative results show that our approach is clearly a step forward towards photo-realistic and animatable full body human avatars.

2 RELATED WORK

While some works focus on adapting character motions to a new geometric environment, e.g. walking on a rough terrain, [Holden et al. 2017] or create character motions from goal oriented user instructions [Starke et al. 2019], we assume the motion is given and review works that focus on animatable characters, learning based cloth deformations, and differentiable rendering.

Video-based Characters. Previous work in the field of video-based characters aims at creating photo-realistic renderings of controllable characters. Classical methods attempt to achieve this by synthesizing textures on surface meshes and/or employing image synthesis techniques in 2D space. Some works [Carranza et al. 2003; Collet et al. 2015; Hilsmann et al. 2020; Li et al. 2014; Zitnick et al. 2004] focus on achieving free-viewpoint replay from multi-view videos with or without 3D proxies, however, they are not able to produce new poses for human characters. The approach of [Stoll et al. 2010] incorporates a physically-based cloth model to reconstruct a rigged fully-animatable character in loose cloths from multi-view videos, but it can only synthesize a fixed static texture for different poses. To render the character with dynamic textures in new poses from arbitrary viewpoints, [Xu et al. 2011] propose a method that first retrieves the most similar poses and viewpoints in a pre-captured database and then applies retrieval based texture synthesis. However, their method takes several seconds per frame and thus cannot support interactive character animation. [Casas et al. 2014; Volino et al. 2014] compute a temporally coherent layered representation of appearance in texture space to achieve interactive speed, but the synthesis quality is limited due to the coarse geometric proxy. Most of the traditional methods for free-viewpoint rendering of video-based characters fall either short in terms of generalization to new poses and/or suffer from a high runtime, and/or a limited synthesis quality.

More recent works employ neural networks to close the gap between rendered virtual characters and real captured images. While some approaches have shown convincing results for the facial area [Kim et al. 2018a; Lombardi et al. 2018], creating photo-real images of the entire human is still a challenge. Most of the methods, which target synthesizing entire humans, learn an image-to-image mapping from renderings of a skeleton [Chan et al. 2019; Esser et al. 2018; Pumarola et al. 2018; Si et al. 2018], depth map [Martin-Brualla et al. 2018], dense mesh [Liu et al. 2020b, 2019b; Sarkar et al. 2020; Wang et al. 2018a] or joint position heatmaps [Aberman et al. 2019], to real images. Among these approaches, the most related work [Liu et al. 2020b] achieves better temporally-coherent dynamic textures by first learning fine scale details in texture space and then translating the rendered mesh with dynamic textures into realistic imagery. While only requiring a single camera, these methods

only demonstrate the rendering from a fixed camera position while the proposed approach works well for arbitrary view points and also models the view-dependent appearance effects. Further, these methods heavily rely on an image-to-image translation network to augment the realism, however, this refinement simply applied in 2D image space leads to missing limbs and other artifacts in their results. In contrast, our method does not require any refinement in 2D image space but explicitly generates high-quality view- and motion-dependent geometry and texture for rendering to avoid such kind of artifacts. Similar to us, Textured Neural Avatars [Shysheya et al. 2019] (TNA) also assumes multi-view imagery is given during training. However, TNA is neither able to synthesize motion- and view-dependent dynamic textures nor to predict the dense 3D surface. Our method can predict motion-dependent deformations on surface geometry as well as dynamic textures from a given pose sequence and camera view leading to video-realistic renderings.

Learning Based Cloth Deformation. Synthesizing realistic cloth deformations with physics-based simulation has been extensively explored [Choi and Ko 2005; Liang et al. 2019; Narain et al. 2012; Nealen et al. 2005; Su et al. 2020; Tang et al. 2018; Tao et al. 2019]. They employ either continuum mechanics principles followed by finite element discretization, or physically consistent models. However, they are computationally expensive and often require manual parameter tuning. To address this issue, some methods [Feng et al. 2010; Guan et al. 2012; Hahn et al. 2014; Kim and Vendrovsky 2008; Wang et al. 2010; Xu et al. 2014; Zurdo et al. 2013] model cloth deformations as a function of the underlying skeletal pose and/or the shape of the person and learn the function from data.

With the development of deep learning, skinning based deformations can be improved [Bailey et al. 2018] over the traditional methods like linear blend skinning [Magnenat-Thalmann et al. 1988]. Other works go beyond skinning based deformations and incorporate deep learning for predicting cloth deformations and learn garment deformations from the body pose and/or shape. Some works [Alldieck et al. 2019, 2018a,b; Bhatnagar et al. 2019; Jin et al. 2018; Pons-Moll et al. 2017] generate per-vertex displacements over a parametric human model to capture the garment deformations. While this is an efficient representation, it only works well for tight cloths such as pants and shirts. [Gundogdu et al. 2019] use neural networks to extract garment features at varying levels of detail (i.e., point-wise, patch-wise and global features). [Patel et al. 2020] decompose the deformation into a high frequency and a low frequency component. While the low-frequency component is predicted from pose, shape and style of garment geometry with an MLP, the high-frequency component is generated with a mixture of shape-style specific pose models. Related to that Choi et al. [2020] predicts the geometry of the naked human from coarse to fine given the skeletal pose. Santesteban et al. [2019] separate the global coarse garment fit, due to body shape, from local detailed garment wrinkles, due to both body pose dynamics and shape. Other methods [Lähner et al. 2018; Zhang et al. 2020b] recover fine garment wrinkles for high-quality renderings or 3D modeling by augmenting a low-resolution normal map of a garment with high-frequency details using GANs. Zhi et al. [2020] also reconstructs albedo textures and refines a coarse geometry obtained from RGB-D data. Our method factors cloth deformation into low-frequency large deformations represented by an embedded graph and high-frequency fine wrinkles modeled by a per-vertex displacements, which allows for synthesizing deformations for any kind of clothing, including also loose clothes. In contrast to the above methods, our approach does not only predict geometric deformations but also a dynamic texture map which allows us to render video realistic controllable characters.

Differentiable Rendering and Neural Rendering. Differentiable rendering bridges the gap between 2D supervision and unknown 3D scene parameters which one wants to learn or optimize. Thus, differentiable rendering allows to train deep architectures, that learn 3D parameters of a scene, solely using 2D images for supervision. OpenDR [Loper and Black 2014] first introduces an approximate differentiable renderer by representing a pixel as a linear combination of neighboring pixels and calculating pixel derivatives using differential filters. [Kato et al. 2018] propose a 3D mesh renderer that is differentiable up to the visibility assumed to be constant during one gradient step. [Liu et al. 2019a] differentiate through the visibility function and replace the z-buffer-based triangle selection with a probabilistic approach which assigns each pixel to all faces of a mesh. DIB-R [Chen et al. 2019] propose to compute gradients analytically for all pixels in an image by representing foreground rasterization as a weighted interpolation of a face's vertex attributes and representing background rasterization as a distance-based aggregation of global face information. SDFDiff [Jiang et al. 2020] introduces a differentiable renderer based on ray-casting SDFs. Our implementation of differentiable rendering follows the one of [Kato et al. 2018] where we treat the surface as non transparent and thus the visibility is non-differentiable. This is preferable in our setting as treating the human body and clothing as transparent would lead to wrong surface deformations and blurry dynamic textures.

Different from differentiable rendering, neural rendering makes almost no assumptions about the physical model and uses neural networks to learn the rendering process from data to synthesize photo-realistic images. Some neural rendering methods [Aberman et al. 2019; Chan et al. 2019; Kim et al. 2018b; Liu et al. 2020b, 2019b; Ma et al. 2017, 2018; Martin-Brualla et al. 2018; Pumarola et al. 2018; Sarkar et al. 2020; Shysheya et al. 2019; Siarohin et al. 2018; Thies et al. 2019; Yoon et al. 2020] employ image-to-image translation networks [Isola et al. 2017; Wang et al. 2018a,b] to augment the quality of the rendering. However, most of these methods suffer from view and/or temporal inconsistency. To enforce view and temporal consistency, some attempts were made to learn scene representations for novel view synthesis from 2D images. Although this kind of methods achieve impressive renderings on static scenes [Liu et al. 2020a; Mildenhall et al. 2020; Sitzmann et al. 2019a,b; Zhang et al. 2020a] and dynamic scenes for playback or implicit interpolation [Li et al. 2020; Lombardi et al. 2019; Park et al. 2020; Pumarola et al. 2020; Raj et al. 2020; Sida Peng 2020; Tretschk et al. 2020; Wang et al. 2020; Xian et al. 2020; Zhang et al. 2020a] and face [Gafni et al. 2020], it is not straightforward to extend these methods to synthesize full body human images with explicit pose control. Instead, our approach can achieve video-realistic renderings of the full human body with motion- and view-dependent dynamic textures for arbitrary body poses *and* camera views.

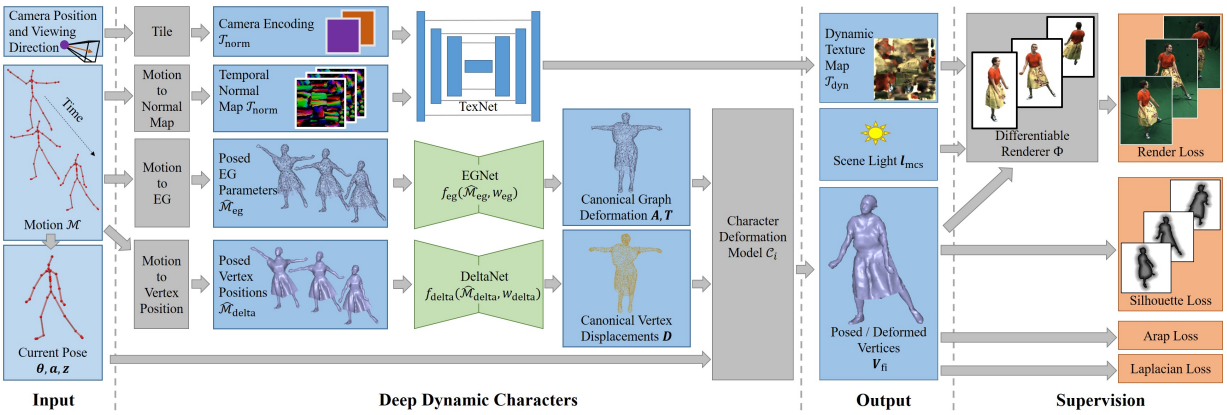


Fig. 2. Overview. Our method takes a motion sequence as input. We convert the pose information into task specific representations making the regression task easier for our network as input and output share the same representation. Then our two networks regress the motion-dependent coarse and fine deformations in the canonical pose. Given pose and deformations, our deformation layer outputs the posed and deformed character. Further, our TexNet regresses a motion- and view-dependent dynamic texture map. The regressed geometry as well as texture are weakly supervised based on multi-view 2D images.

3 METHOD

Given multi-view images for training, our goal is to learn a poseable 3D character with dense deforming geometry of the full body and view- and motion-dependent textures that can be driven just by posing a skeleton and defining a camera view. We propose a weakly supervised learning method with only multi-view 2D supervision in order to remove the need of detailed 3D ground truth geometry and 3D annotations. Once trained, our network takes the current pose and a frame window of past motions of a moving person as input and outputs the motion-dependent geometry and texture, as shown in Fig. 2. Note that our deformed geometry captures not only per-bone rigid transformations via classical skinning but also non-rigid deformations of clothing dependent on the current pose as well as the velocity and acceleration derived from the past motions. In the following, we first introduce our deformable character model (Sec. 3.1) and the data acquisition process (Sec. 3.2). To regress the non-rigid deformations, we proceed in a coarse-to-fine manner. First, we regress the deformation as rotations and translations of a coarse embedded graph (Sec. 3.3) only using multi-view foreground images as supervision signal. As a result, we obtain a posed and deformed character that already matches the silhouettes of the multi-view images. Next, we define a differentiable rendering layer which allows us to optimize the scene lighting which accounts for white balance shift and directional light changes (Sec 3.4). Finally, our second network regresses per-vertex displacements to account for finer wrinkles and deformations that cannot be captured by the embedded deformation. This layer can be trained using again the foreground masks, but in addition we also supervise it with a dense rendering loss using the previously optimized scene lighting (Sec. 3.5). Last, our dynamic texture network takes a view and motion encoding in texture space and outputs a dynamic texture (Sec. 3.6) to further enhance the realism of our 3D character. Similar to before, the texture network is weakly supervised using our differentiable renderer. Note that none of our components requires ground truth 3D geometry and can be entirely trained weakly supervised.

3.1 Character Deformation Model

Acquisition. Our method is person-specific and requires a 3D template model of the actor. We first scan the actor in T-pose using a 3D scanner [Treedys 2020]. Next, we use a commercial multi-view stereo reconstruction software [Photo Scan 2016] to reconstruct the 3D mesh with a static texture $\mathcal{T}_{\text{st}}$ and downsample the reconstructed mesh to a resolution of around 5000 vertices. Like [Habermann et al. 2019, 2020], we manually segment the mesh using the common human parsing labels and define per-vertex rigidity weights s_i to model different degrees of deformation for different materials, where low rigidity weights allow more non-rigid deformations and vice versa, e.g., skin has higher rigidity weights than clothing.

Skeleton. The template mesh is manually rigged to a skeleton. Here, the skeleton is parameterized as the set $\mathcal{S} = \{\boldsymbol{\theta}, \boldsymbol{\alpha}, \mathbf{z}\}$ with joint angles $\boldsymbol{\theta} \in \mathbb{R}^{57}$, global rotation $\boldsymbol{\alpha} \in \mathbb{R}^3$, and global translation $\mathbf{z} \in \mathbb{R}^3$, where skinning weights are automatically computed using Blender [Blender 2020]. This allows us to deform the mesh for a given pose by using dual quaternion skinning [Kavan et al. 2007].

Embedded Deformation. As discussed before, the traditional skinning process alone is hardly able to model non-rigid deformations such as the swinging of a skirt. To address this issue, we model the non-rigid deformations in the canonical pose from coarse to fine *before* applying dual quaternion skinning. On the coarse level, large deformations are captured with the embedded deformation representation [Sorkine and Alexa 2007; Sumner et al. 2007] which requires a small amount of parameters. We construct an embedded graph $\mathcal{G}$ consisting of K nodes (K is around 500 in our experiments) by downsampling the mesh. The embedded graph $\mathcal{G}$ is parameterized with $\mathbf{A} \in \mathbb{R}^{K\times 3}$ and $\mathbf{T} \in \mathbb{R}^{K\times 3}$, where each row k of $\mathbf{A}$ and $\mathbf{T}$ is the local rotation $\mathbf{a}_k \in \mathbb{R}^3$ in the form of Euler angles and local translation $\mathbf{t}_k \in \mathbb{R}^3$ of node k with respect to the initial position $\mathbf{g}_k$ of node k. The connectivity of the graph node k can be derived from the connectivity of the downsampled mesh and is denoted as $\mathcal{N}_{\text{n}}(k)$. To deform the original mesh with the embedded graph,

the movement of each vertex on the original mesh is calculated as a linear combination of the movements of all the nodes of the embedded graph. Here, the weights $w_{i,k} \in \mathbb{R}$ for vertex i and node k are computed based on the geodesic distance between the vertex i and the vertex on the original mesh that has the smallest Euclidean distance to the node k, where the weight is set to zero if the distance exceeds a certain threshold. We denote the set of nodes that finally influences the movement of the vertex i as $\mathcal{N}_{\text{vn}}(i)$.

Vertex Displacements. On the fine level, in addition to the embedded graph, which models large deformations, we use vertex displacements to recover fine-scale deformations, where a displacement $\mathbf{d}_i \in \mathbb{R}^3$ is assigned to each vertex i. Although regressing so many parameters is not an easy task, the training of the vertex displacement can still be achieved since the embedded graph captures most of deformations on a coarse level. Thus, the regressed displacements, the network has to learn, are rather small.

Character Deformation Model. Given the skeletal pose $\boldsymbol{\theta}, \boldsymbol{\alpha}, \mathbf{z}$, the embedded graph parameters $\mathbf{A}, \mathbf{T}$, and the vertex displacements $\mathbf{d}_i$, we can deform each vertex i with the function

$$C_i(\boldsymbol{\theta}, \boldsymbol{\alpha}, \mathbf{z}, \mathbf{A}, \mathbf{T}, \mathbf{d}_i) = \mathbf{v}_i \tag{1}$$

which defines our final character representation. Specifically, we first apply the embedded deformation and the per-vertex displacements to the template mesh in canonical pose, which significantly simplifies the learning of non-rigid deformations by alleviating ambiguities in the movements of mesh vertices caused by pose variations. Thus, the deformed vertex position is given as

$$\mathbf{y}_i = \mathbf{d}_i + \sum_{k \in \mathcal{N}_{\text{vn}}(i)} w_{i,k}(R(\mathbf{a}_k)(\hat{\mathbf{v}}_i - \mathbf{g}_k) + \mathbf{g}_k + \mathbf{t}_k), \tag{2}$$

where $\hat{\mathbf{v}}_i$ is the initial position of vertex i in the template mesh. $R : \mathbb{R}^3 \rightarrow SO(3)$ converts the Euler angles to a rotation matrix. We apply the skeletal pose to the deformed vertex $\mathbf{y}_i$ in canonical pose to obtain the deformed and posed vertex in the global space

$$\mathbf{v}_i = \mathbf{z} + \sum_{k \in \mathcal{N}_{\text{vn}}(i)} w_{i,k}(R_{\text{sk},k}(\boldsymbol{\theta}, \boldsymbol{\alpha})\mathbf{y}_i + t_{\text{sk},k}(\boldsymbol{\theta}, \boldsymbol{\alpha})), \tag{3}$$

where the rotation $R_{\text{sk},k}$ and the translation $t_{\text{sk},k}$ are derived from the skeletal pose using dual quaternion skinning, and $\mathbf{z}$ is the global translation of the skeleton. Note that Eq. 2 and 3 are fully differentiable with respect to pose, embedded graph and vertex displacements. Thus, gradients can be propagated in learning frameworks. The final model does not only allow us to pose the mesh via skinning but also to model non-rigid surface deformations in a coarse to fine manner via embedded deformation and vertex displacements. Further, it disentangles the pose and the surface deformation, where the later is represented in the canonical pose space.

The main difference to data-driven body models, e.g. SMPL [Loper et al. 2015], is that our character formulation allows posing, deforming, and texturing using an effective and simple equation which is differentiable to all its input parameters. SMPL and other human body models do not account for deformations (e.g. clothing) and they also do not provide a texture. Our specific formulation allows seamless integration into a learning framework and learning its parameters conditioned on skeletal motion (and camera pose) as well as adding spatial regularization from coarse to fine which is important when considering a weakly supervised setup.

3.2 Data Capture and Motion Preprocessing

For supervision, our method requires multi-view images and foreground masks of the actor performing a wide range of motions at varying speeds to sample different kinds of dynamic deformations of clothing caused by the body motion. Thus, we place the subject in a multi camera capture studio with green screen and record a sequence with $C = 120$ synchronized and calibrated 4K cameras at 25 frames per second. We apply color keying to segment the foreground and convert the foreground masks to distance transform images [Borgefors 1986]. We denote the fth frame of camera c and its corresponding distance transform image and the foreground mask as $\mathcal{I}_{c,f}$, $\mathcal{D}_{c,f}$, and $\mathcal{F}_{c,f}$, respectively.

We further track the human motions using a multi-view markerless motion capture system [TheCaptury 2020]. We denote the tracked motion of the fth frame as $\mathcal{S}_f = \{\boldsymbol{\theta}_f, \boldsymbol{\alpha}_f, \mathbf{z}_f\}$. Next, we normalize the temporal window of motions $\mathcal{M}_t = \{\mathcal{S}_f : f \in \{t - F, ..., t\}\}$ for geometry and texture generation separately as it is very hard to sample all combinations of rigid transforms and joint angle configurations during training data acquisition. The normalization for geometry generation is based on two observations: 1) The global position of the motion sequence should not influence the dynamics of the geometry; we therefore normalize the global translation of $\mathcal{S}_f$ across different temporal windows of motions while keeping relative translations between the frames in each temporal window, i.e., we set $\hat{\mathbf{z}}_t = \mathbf{0}$ and $\hat{\mathbf{z}}_{t'} = \mathbf{z}_{t'} - \mathbf{z}_t$ for $t' \in \{t - F, ..., t - 1\}$ where $\mathbf{0}$ is the zero vector in $\mathbb{R}^3$. 2) The rotation around the y axis will not affect the geometry generation as it is in parallel with the gravity direction; thus, similar to normalizing the global translation, we set $\hat{\boldsymbol{\alpha}}_{y,t} = 0$ and $\hat{\boldsymbol{\alpha}}_{y,t'} = \boldsymbol{\alpha}_{y,t'} - \boldsymbol{\alpha}_{y,t}$. We denote the temporal window of the normalized motions for geometry generation as $\hat{\mathcal{M}}_t = \{\hat{\mathcal{S}}_f : f \in \{t - F, ..., t\}\}$, where $\hat{\mathcal{S}}_f = \{\hat{\boldsymbol{\theta}}_f, \hat{\boldsymbol{\alpha}}_f, \hat{\mathbf{z}}_f\}$. For texture generation, we only normalize the global translation, but not the rotation around the y axis to get the normalized motions $\tilde{\mathcal{M}}_t$, since we would like to generate view-dependent textures where the subjects relative direction towards the light source and therefore the rotation around the y axis matters. In all our results, we set $F = 2$. For readability reasons, we assume t is fixed and drop the subscript.

3.3 Embedded Deformation Regression

3.3.1 Embedded Deformation Regression. With the skeletal pose alone, the non-rigid deformations of the skin and clothes cannot be generated. We therefore introduce an embedded graph network, *EGNet*, to produce coarse-level deformations. *EGNet* learns a mapping from the temporal window of normalized motions $\hat{\mathcal{M}}$ to the rotations $\mathbf{A}$ and translations $\mathbf{T}$ of the embedded graph defined in the canonical pose for the current frame (i.e., the last frame of the temporal window). *EGNet* learns deformations correlated to the velocity and acceleration at the current frame since it takes the pose of the current frame as well as the previous two frames as input. Directly regressing $\mathbf{A}$ and $\mathbf{T}$ from the normalized skeletal motion $\hat{\mathcal{M}}$ is challenging as the input and output are parameterized in a different way, i.e., $\hat{\mathcal{M}}$ represents skeleton joint angles while $\mathbf{A}$ and

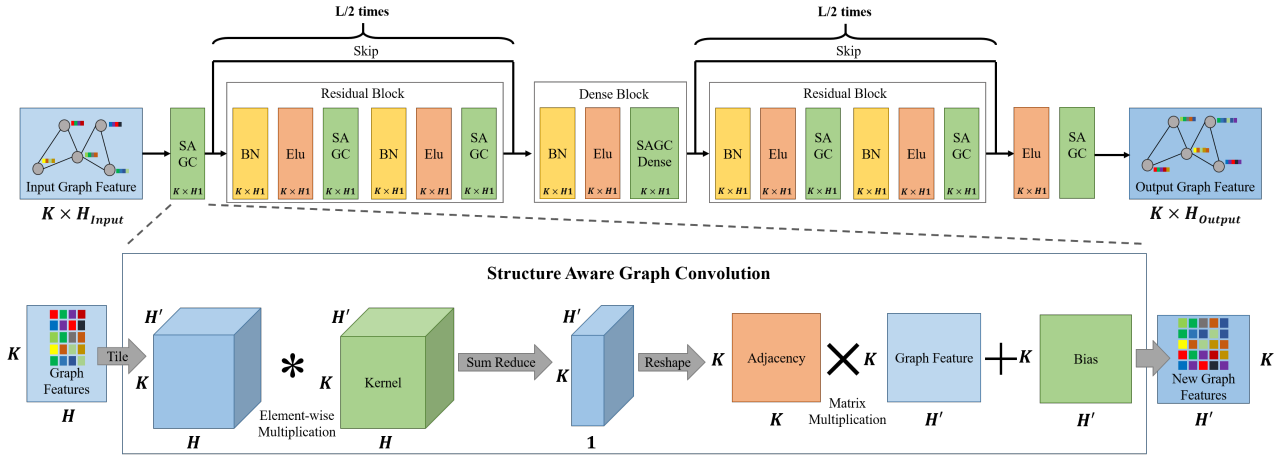


Fig. 3. Structure aware graph convolutional network (top) as well as a detailed illustration of the proposed Structure Aware Graph Convolution (bottom).

T model rotation and translation of the graph nodes. To address this issue, we formulate this regression task as a graph-to-graph translation problem rather than a skeleton-to-graph one. Specifically, we pose the embedded graph with the normalized skeletal motion $\hat{\mathcal{M}}$ using dual quaternion skinning [Kavan et al. 2007] to obtain the rotation and translation parameters $\hat{\mathcal{M}}_{\text{eg}} \in \mathbb{R}^{K\times 6(F+1)}$ of the embedded graph. Therefore, the mapping of *EGNet* can be formulated as $f_{\text{eg}}(\hat{\mathcal{M}}_{\text{eg}}, \mathbf{w}_{\text{eg}}) = (\mathbf{A}, \mathbf{T})$, which takes the posed embedded graph rotations and translations $\hat{\mathcal{M}}_{\text{eg}}$ and learnable weights $\mathbf{w}_{\text{eg}}$ as inputs and outputs the embedded deformation $(\mathbf{A}, \mathbf{T})$ in canonical pose. Using the character representation defined in Eq. 1, the posed and coarsely deformed character is defined as

$$C_i(\boldsymbol{\theta}, \boldsymbol{\alpha}, \mathbf{z}, f_{\text{eg}}(\hat{\mathcal{M}}_{\text{eg}}, \mathbf{w}_{\text{eg}}), \mathbf{0}) = \mathbf{v}_{\text{co},i}. \tag{4}$$

Here, $\boldsymbol{\theta}, \boldsymbol{\alpha}, \mathbf{z}$ are the unnormalized pose of the last frame of the motion sequence and the displacements are set to zero. Next, we explain our novel graph convolutional architecture of EGNet.

3.3.2 Structure Aware Graph Convolution. Importantly, our graph is fixed as our method is person specific. Thus, the spatial relationship between the graph nodes and their position implicitly encode a strong prior. For example, a node that is mostly attached to skin vertices will deform very different than nodes that are mainly connected to vertices of a skirt region. This implies that learnable node features require different properties depending on which node is considered. However, recent graph convolutional operators [Defferrard et al. 2017] apply the same filter on every node which contradicts the above requirements. Therefore, we aim for a graph convolutional operator that applies an individual kernel per node.

Thus, we propose a new *Structure Aware Graph Convolution (SAGC)*. To define the per node SAGC, we assume an input node feature $\mathbf{f}_k \in \mathbb{R}^H$ of size H is given and the output feature dimension is H' for a node k. Now, the output feature $\mathbf{f}'_k$ can be computed as

$$\mathbf{f}'_k = \mathbf{b}_k + \sum_{l\in\mathcal{N}_R(k)} a_{k,l}\mathbf{K}_l\mathbf{f}_l \tag{5}$$

where $\mathcal{N}_R(k)$ is the R-ring neighbourhood of the graph node k. $\mathbf{b}_k \in \mathbb{R}^{H'}$ and $\mathbf{K}_l \in \mathbb{R}^{H'\times H}$ are a trainable bias vector and kernel matrix. $a_{k,l}$ is a scalar weight that is computed as

$$a_{k,l} = \frac{r_{k,l}}{\sum_{l\in\mathcal{N}_R(k)} r_{k,l}} \tag{6}$$

where for a node k $r_{k,l}$ is the inverse ring value, e.g., for the case $l = k$ the value is R and for the direct neighbours of k the value is $R-1$. More intuitively, our operator computes a linear combination of modified features $\mathbf{K}_l\mathbf{f}_l$ of node k and neigbouring nodes l within the R-ring neighbourhood weighted by $a_{k,l}$ that has a linear falloff to obtain the new feature for node k. Importantly, each node has its own learnable kernel $\mathbf{K}_l$ and bias $\mathbf{b}_k$ weights allowing features at different locations in the graph to account for different spatial properties. As shown at the bottom of Fig. 3, the features for each node can be efficiently computed in parallel and by combining all the per node input/output features, one obtains the corresponding input/output feature matrices $\mathbf{F}_k, \mathbf{F}'_k$.

3.3.3 Structure Aware Graph Convolutional Network. Our structure-aware graph convolutional network (SAGCN) takes as input a graph feature matrix $\mathbf{F}_{\text{input}} \in \mathbb{R}^{K\times H_{\text{input}}}$ and outputs a new graph feature matrix $\mathbf{F}_{\text{output}} \in \mathbb{R}^{K\times H_{\text{output}}}$ (see Fig. 3). First, $\mathbf{F}_{\text{input}}$ is convolved with the SAGC operator, resulting in a feature matrix of size $K \times H_1$. Inspired by the ResNet architecture [He et al. 2016], we also use so-called residual blocks that take the feature matrix of size $K \times H_1$ and output a feature matrix of the same size. Input and output feature matrices are connected via skip connections which prevent vanishing gradients, even for very deep architectures. A residual block consists of two chains of a batch normalization, an Elu activation function, and a SAGC operation. For a very large number of graph nodes, the local features can barely spread through the entire graph. To still allow the network to share features between far nodes we propose a so-called dense block consisting of a batch normalization, an Elu activation, and a SAGC operator. Importantly, for this specific dense block we set all $a_{k,l} = 1$ which allows to share features between far nodes. In total, we use L residual blocks, half of them before and half of them after the dense block. The last layers (Elu and SAGC) resize the features to the desired output size.

We now define EGNet as a SAGCN architecture where the graph is given as the embedded graph $\mathcal{G}$. The input feature matrix is given by the normalized embedded graph rotations and translations $\hat{\mathcal{M}}_{\text{eg}}$ and the output is the deformation parameters $(\mathbf{A}, \mathbf{T})$. Thus, the input and output feature sizes are $H_{\text{input}} = 6(F+1)$ and $H_{\text{output}} = 6$, respectively. Further, we set $H_1 = 16$, $L = 8$, and $R = 3$. As $\mathcal{G}$ only contains around 500 nodes, we did not employ a dense block.

3.3.4 Weakly Supervised Losses. To train the weights $\mathbf{w}_{\text{eg}}$ of EGNet $f_{\text{eg}}(\hat{\mathcal{M}}_{\text{eg}}, \mathbf{w}_{\text{eg}})$, we only employ a weakly supervised loss on the posed and deformed vertices $\mathbf{V}_{\text{co}}$ and on the regressed embedded deformation parameters $(\mathbf{A}, \mathbf{T})$ directly as

$$\mathcal{L}_{\text{eg}}(\mathbf{V}_{\text{co}}, \mathbf{A}, \mathbf{T}) = \mathcal{L}_{\text{sil}}(\mathbf{V}_{\text{co}}) + \mathcal{L}_{\text{arap}}(\mathbf{A}, \mathbf{T}). \tag{7}$$

Here, the first term is our multi-view image-based data loss and the second term is a spatial regularizer.

Silhouette Loss. Our multi-view silhouette loss

$$\begin{aligned}\mathcal{L}_{\text{sil}}(\mathbf{V}_{\text{co}}) = &\sum_{c=1}^{C} \sum_{i \in \mathcal{B}_c} \rho_{c,i} \| \mathcal{D}_c \left(\pi_c \left(\mathbf{v}_{\text{co},i} \right) \right) \|^2 \\ &+ \sum_{c=1}^{C} \sum_{\mathbf{p} \in \{\mathbf{u} \in \mathbb{R}^2 | \mathcal{D}_c(\mathbf{u}) = 0\}} \| \pi_c \left(\mathbf{v}_{\text{co},\mathbf{p}} \right) - \mathbf{p} \|^2 \end{aligned} \tag{8}$$

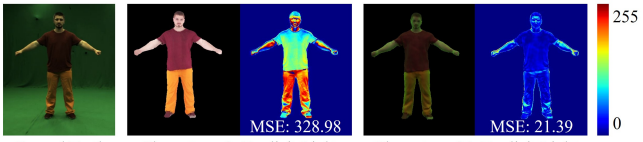
ensures that the silhouette of the projected character model aligns with the multi-view image silhouettes in an analysis-by-synthesis manner. Therefore, we employ a bi-sided loss. The first part of Eq. 8 is a model-to-data loss which enforces that the projected boundary vertices are pushed to the zero contour line of the distance transform image $\mathcal{D}_c$ for all cameras c. Here, π_c is the perspective camera projection of camera c and $\rho_{c,i}$ is a scalar weight accounting for matching image and model normals [Habermann et al. 2019]. $\mathcal{B}_c$ is the set of boundary vertices, e.g., the vertices that lie on the boundary after projecting onto camera view c. $\mathcal{B}_c$ can be efficiently computed using our differentiable renderer, that is introduced later, by rendering out the depth maps and checking if a projected vertex lies near a background pixel in the depth map. The second part of Eq. 8 is a data-to-model loss which ensures that all silhouette pixels $\{\mathbf{u} \in \mathbb{R}^2 | \mathcal{D}_c(\mathbf{u}) = 0\}$ are covered by their closest vertex $\mathbf{v}_{\text{co},\mathbf{p}}$ using the Euclidean distance in 2D image space as the distance metric.

ARAP Loss. Only using the above loss would lead to an ill-posed problem as vertices could drift along the visual hull carved by the silhouette images without receiving any penalty resulting in distorted meshes. Thus, we employ an as-rigid-as-possible regularization [Sorkine and Alexa 2007; Sumner et al. 2007] defined as

$$\mathcal{L}_{\text{arap}}(\mathbf{A}, \mathbf{T}) = \sum_{k=1}^{K} \sum_{l \in \mathcal{N}_{\text{n}}(k)} u_{k,l} \| d_{k,l}(\mathbf{A}, \mathbf{T}) \|_1 \tag{9}$$

$$d_{k,l}(\mathbf{A}, \mathbf{T}) = R(\mathbf{a}_k)(\mathbf{g}_l - \mathbf{g}_k) + \mathbf{t}_k + \mathbf{g}_k - (\mathbf{g}_l + \mathbf{t}_l).$$

We use material aware weighting factors $u_{k,l}$ [Habermann et al. 2020] computed by averaging the rigidity weights s_i of all vertices attached to node k and l. Thus, different levels of rigidity are assigned to individual surface parts, e.g., graph nodes attached to skirt vertices can deform more freely than those attached to skin vertices.



Ground Truth | Character **w/o** Explicit Light Modeling | Character **with** Explicit Light Modeling

Fig. 4. From left to right. Ground truth image. Template rendered with the static texture. Mean squared pixel error (MSE) image computed from the masked ground truth and the rendering. Template rendered with the static texture and *the optimized lighting*. MSE image computed from the ground truth and the render with optimized lighting. Note that the optimized lighting clearly lowers the error between the rendering and the ground truth which is advantageous for learning the per-vertex displacements. Further note that this is not our final appearance result. Instead, we only use the optimized lighting to improve the dense rendering loss which supervises our displacement network that will be introduced in the next section.

3.4 Lighting Estimation

So far, we can obtain the posed and coarsely deformed character $\mathbf{V}_{\text{co}}$ using EGNet and Eq. 4. What is still missing are the finer deformations which are hard to capture just with multi-view silhouette images. Thus, we aim for a dense rendering loss that takes the posed and deformed geometry along with the static texture $\mathcal{T}_{\text{st}}$, renders it into all camera views and compares it to the corresponding images. However, the lighting condition differs when capturing the scan of the subject and therefore the texture and the lighting in the multi-camera studio can vary due to different light temperatures, camera optics and sensors, and scene reflections as shown in Fig. 4. As a remedy, we propose a differentiable rendering that also accounts for the difference in lighting and explicitly optimize the lighting parameters for the multi-camera studio sequences.

Differentiable Rendering. We assume the subject has a purely Lambertian surface reflectance [Lambert 1760] and that the light sources are sufficiently far away resulting in an overall smooth lighting environment. Hence, we can use the efficient Spherical Harmonics (SH) lighting representation [Mueller 1966] which models the scene lighting only with a few coefficients. To account for view-dependent effects, each of the C cameras has its own lighting coefficients $\mathbf{l}_{\text{c}} \in \mathbb{R}^{9\times3}$ which in total sums up to $27C$ coefficients. We assume that the image has a resolution of $W \times H$. To compute the RGB color of a pixel $\mathbf{u} \in \mathcal{R}$ where $\mathcal{R} = \{(u, v) | u \in [1, W], v \in [1, H]\}$ in camera view c, we use the rendering function

$$\Phi_{c,\mathbf{u}}(\mathbf{V}, \mathcal{T}, \mathbf{l}_{\text{c}}) = a_{c,\mathbf{u}}(\mathbf{V}, \mathcal{T}) \cdot i_{c,\mathbf{u}}(\mathbf{V}, \mathbf{l}_{\text{c}}) \tag{10}$$

which takes the vertex positions $\mathbf{V}$, the texture $\mathcal{T}$, and the lighting coefficients $\mathbf{l}_{\text{c}}$ for camera c. As we assume a Lambertian reflectance model, the rendering equation simplifies to a dot product of the albedo $a_{c,\mathbf{u}}(\mathbf{V}, \mathcal{T})$ of the projected surface and the illumination $i_{c,\mathbf{u}}(\mathbf{V}, \mathbf{l}_{\text{c}})$. The albedo can be computed as

$$a_{c,\mathbf{u}}(\mathbf{V}, \mathcal{T}) = v_{c,\mathbf{u}}(\mathbf{V}) t_{c,\mathbf{u}}(\mathbf{V}, \mathcal{T}) \tag{11}$$

where $v_{c,\mathbf{u}}(\mathbf{V})$ is an indicator function that computes whether a surface is visible or not given the pixel position, camera, and surface. Like traditional rasterization [Pineda 1988], $t_{c,\mathbf{u}}(\mathbf{V}, \mathcal{T})$ computes the

Fig. 5. From left to right. Input motion. Our result without the displacements predicted by DeltaNet. Our result with the predicted displacements. Note that the displacements clearly improve the overlay as they allow capturing finer geometric details.

barycentric coordinates of the point on the triangle that is covering pixel $\mathbf{u}$, which are then used to bi-linearly sample the position in texture map space. The lighting can be computed in SH space as

$$i_{c,\mathbf{u}}(\mathbf{V}, \mathbf{l}_c) = \sum_{j=1}^{9} \mathbf{l}_{c,j} SH_j(n_{c,\mathbf{u}}(\mathbf{V})) \tag{12}$$

where $\mathbf{l}_{c,j} \in \mathbb{R}^3$ are the jth SH coefficients for each color channel and SH_j are the corresponding SH basis functions. $n_{c,\mathbf{u}}(\mathbf{V})$ computes the screen space pixel normal given the underlying geometry.

Note that the final color $\Phi_{c,\mathbf{u}}(\mathbf{V}, \mathcal{T}, \mathbf{l}_c)$ only depends on the geometry $\mathbf{V}$, the texture $\mathcal{T}$, and the lighting coefficients $\mathbf{l}_c$ assuming camera and pixel position are fixed. As all above equations (except visibility) are differentiable with respect to these variables we can backpropagate gradients through the rendering process. The visibility $v_{c,\mathbf{u}}(\mathbf{V})$ is fixed during one gradient step.

Lighting Optimization. To optimize the lighting, we assume the texture and geometry are fixed. Therefore, we set the texture to $\mathcal{T} = \mathcal{T}_{\text{st}}$ which is the static texture obtained from the scan. For the geometry, we set $\mathbf{V} = \mathbf{V}_{\text{co}}$ which is the deformed and posed vertex positions regressed by EGNet. Now, the lighting coefficients $\mathbf{l}_{\text{mcs},c}$ for camera c can be computed by minimizing

$$\mathcal{L}_{\text{light}}(\mathbf{l}_{\text{mcs},c}) = \sum_{\mathbf{u} \in \mathcal{R}} \|\Phi_{c,\mathbf{u}}(\mathbf{V}_{\text{co}}, \mathcal{T}_{\text{st}}, \mathbf{l}_{\text{mcs},c}) - \mathcal{I}_{c,\mathbf{u}}\|^2 \tag{13}$$

for all frames of the training sequence. Note that we use all frames while the lighting coefficients are the same across frames. As we cannot solve for all frames jointly, we employ a stochastic gradient descent which randomly samples 4 frames and apply 30,000 iterations. As a result, the optimal lighting coefficients $\mathbf{l}^*_{\text{mcs},c}$ are obtained and the rendering with the static texture and the optimized lighting matches the global appearance much better than a rendering which is not explicitly modeling lighting (see Fig. 4). As the lighting and the texture are now known, we can leverage the rendering function Eq. 10 to densely supervise the per-vertex displacements which we want to regress on top of the embedded deformation parameters.

3.5 Vertex Displacement Regression

3.5.1 Displacement Network DeltaNet. Our goal is capturing also finer deformations which our character representation models as per-vertex displacements $\mathbf{d}_i$ that were previously set to zero. Our second network, called *DeltaNet*, takes again the motion sequence and regresses the displacements $\mathbf{D} \in \mathbb{R}^{N \times 3}$ for the N vertices of the template mesh in canonical pose. Here, the ith row of $\mathbf{D}$ contains the displacement $\mathbf{d}_i$ for vertex i. Similar to the EGNet, we represent the pose in the same space as the output space of the regression task. Thus, we pose the template mesh to the respective poses from our normalized motion $\hat{\mathcal{M}}$ using dual quaternion skinning resulting in $F+1$ consecutive 3D vertex positions which we denote as $\hat{\mathcal{M}}_{\text{delta}} \in \mathbb{R}^{N \times 3(F+1)}$. We denote DeltaNet as the function $f_{\text{delta}}(\hat{\mathcal{M}}_{\text{delta}}, \mathbf{w}_{\text{delta}}) = \mathbf{D}$ where $\mathbf{w}_{\text{delta}}$ are the trainable network weights. Similarly, the displacement for a single vertex is referred to as $f_{\text{delta},i}(\hat{\mathcal{M}}_{\text{delta}}, \mathbf{w}_{\text{delta}}) = \mathbf{d}_i$ and the final posed and deformed character vertices are defined as

$$C_i(\boldsymbol{\theta}, \boldsymbol{\alpha}, \mathbf{z}, f_{\text{eg}}(\hat{\mathcal{M}}_{\text{eg}}, \mathbf{w}_{\text{eg}}), f_{\text{delta},i}(\hat{\mathcal{M}}_{\text{delta}}, \mathbf{w}_{\text{delta}})) = \mathbf{v}_{\text{fi},i}. \tag{14}$$

$\mathbf{V}_{\text{fi}} \in \mathbb{R}^{N \times 3}$ denotes the matrix that contains all the posed and deformed vertices. Again, we use our SAGCN architecture as it is able to preserve local structures better than fully connected architectures. The graph is defined by the connectivity of our template mesh and each vertex is a graph node. The input is $\hat{\mathcal{M}}_{\text{delta}}$ and therefore the input and output feature sizes are $H_{\text{input}} = 3(F+1)$ and $H_{\text{output}} = 3$, respectively. Further, we set $H_1 = 16$, $L = 8$, and $R = 3$. Different to EgNet, we employ the dense block as the mesh graph is very large and thus sharing features for very far nodes is difficult otherwise. Fig. 5 shows that adding these displacements improves the silhouette matching as the finer geometric details can be captured.

3.5.2 Weakly Supervised Losses. We weakly supervise the displacement predictions and therefore $\mathbf{V}_{\text{fi}}$ using the loss function

$$\mathcal{L}_{\text{Delta}}(\mathbf{V}_{\text{fi}}) = \mathcal{L}_{\text{chroma}}(\mathbf{V}_{\text{fi}}) + \mathcal{L}_{\text{sil}}(\mathbf{V}_{\text{fi}}) + \mathcal{L}_{\text{lap}}(\mathbf{V}_{\text{fi}}) \tag{15}$$

which is composed of two multi-view image-based data terms and a spatial regularizer. $\mathcal{L}_{\text{sil}}$ is the silhouette loss introduced in Eq. 8 but now applied to the vertices after adding the displacements to still ensure matching model and image silhouettes.

Chroma Loss. The silhouette-based loss alone can only constrain the boundary vertices of the model. But since we want to learn the displacements per vertex a denser supervision is required and therefore we employ a dense rendering loss

$$\mathcal{L}_{\text{chroma}}(\mathbf{V}_{\text{fi}}) = \sum_{c=1}^{C} \sum_{\mathbf{u} \in \mathcal{R}} \|g(\Phi_{c,\mathbf{u}}(\mathbf{V}_{\text{fi}}, \mathcal{T}_{\text{st}}, \mathbf{l}^*_{\text{mcs},c})) - g(\mathcal{I}_{c,\mathbf{u}})\|^2 \tag{16}$$

which renders the mesh $\mathbf{V}_{\text{fi}}$ into the camera view c and compares it with the ground truth image $\mathcal{I}_c$ by using our differentiable renderer proposed in the previous section. In contrast to the previous rendering loss (see Eq. 13), we apply the color transform g to both the rendered and the ground truth image. g converts RGB values into the YUV color space and only returns the UV channels. Thus, our loss is more invariant to shadow effects such as self shadows which cannot be modeled by our renderer. Instead, the loss mainly compares the chroma values of the rendering and the ground truth.

Laplacian Loss. Only using the multi-view image-based constrains can still lead to distorted geometry. Thus, we further regularize the

posed and deformed model with a Laplacian regularizer

$$\mathcal{L}_{\text{lap}}(\mathbf{V}_{\text{fi}}) = \sum_{i=1}^{N} s_i \| |\mathcal{N}_i| \left(\mathbf{v}_{\text{fi},i} - \mathbf{v}_{\text{co},i}\right) - \sum_{j \in \mathcal{N}_i} \left(\mathbf{v}_{\text{fi},j} - \mathbf{v}_{\text{co},j}\right) \|^2 \quad (17)$$

which ensures that the Laplacian of the mesh before and after adding the displacements are locally similar. Here, $\mathcal{N}_i$ is the set that contains the indices of the one ring neighbourhood of vertex i and s_i are the per-vertex spatially varying regularization weights.

3.6 Dynamic Texture Regression

To add further realism to our poseable neural character it is indispensable to have a realistic looking texture. Although our scan provides a static texture, we found that wrinkles are baked in and thus look unrealistic for certain poses and further it cannot account for view-dependent effects. Therefore, our goal is to also regress a motion and view point dependent texture $\mathcal{T}_{\text{dyn}} \in \mathbb{R}^{1024\times1024\times3}$.

As explained in Sec. 3.2, we use our normalized motion $\tilde{\mathcal{M}}$ as a conditioning input. Regressing textures just from these joint angles is difficult as the input and output are in different spaces. Thus, we pose the mesh according to the poses in $\tilde{\mathcal{M}}$ and render the global normals into a texture map. By stacking the normal maps for each of the F+1 poses in $\tilde{\mathcal{M}}$, we obtain $\mathcal{T}_{\text{norm}} \in \mathbb{R}^{1024\times1024\times3(F+1)}$ where we use a texture size of 1024×1024. As textural appearance does not only depend on poses but also on the positioning of the subject with respect to the camera, we further encode the camera position and orientation into texture space denoted as $\mathcal{T}_{\text{cam}} \in \mathbb{R}^{1024\times1024\times6}$ where each pixel contains the position and orientation of the camera. By concatenating $\mathcal{T}_{\text{norm}}$ and $\mathcal{T}_{\text{cam}}$, we obtain $\mathcal{T}_{\text{input}} \in \mathbb{R}^{1024\times1024\times3(F+1)+6}$ which is our final input to the texture regression network.

Our texture network, *TexNet*, is based on the UNet architecture [Isola et al. 2016] which we adapted to handle input and output dimensions of size 1024 × 1024. It takes the input texture encoding $\mathcal{T}_{\text{input}}$ and outputs the dynamic texture $\mathcal{T}_{\text{dyn}}$. Note that due to our input representation, the network can learn motion-dependent texture effects as well as view-dependent effects.

Photometric Loss. To supervise $\mathcal{T}_{\text{dyn}}$ for camera c', we impose a texture loss

$$\mathcal{L}_{\text{texture}}(\mathcal{T}_{\text{dyn}}) = \sum_{\mathbf{u} \in \mathcal{R}} |\hat{\mathcal{F}}_{c',\mathbf{u}}(\Phi_{c',\mathbf{u}}(\mathbf{V}_{\text{fi}}, \mathcal{T}_{\text{dyn}}, \mathbf{I}_{\text{sh}}) - \mathcal{I}_{c',\mathbf{u}})| \quad (18)$$

which renders our character using the dynamic texture regressed from TexNet and the geometry from our EgNet and DefNet and compares it to the real image. Here, $\hat{\mathcal{F}}_{c,\mathbf{u}}$ is the eroded image foreground mask. We apply an erosion to avoid that background pixels are projected into the dynamic texture if the predicted geometry does not perfectly align with the image. $\mathbf{I}_{\text{sh}}$ denotes the identity lighting. In contrast to the previous rendering losses, we only supervise the network on the conditioning view and not on all camera views.

3.7 Implementation Details

In all experiments, we use the Adam optimizer [Kingma and Ba 2014]. Due to the memory limits and training time, we randomly sample 40 cameras views (if available) for all multi-view losses. The distance transform images have a resolution of 350 × 350. The rendering resolution of the differentiable renderer is 512×512 (643×470) for the training of DeltaNet and the lighting optimization and 1024 × 1024 (1285 × 940) for the training of TexNet. We train EGNet for 360,000 iterations with a batch size of 40 where we balance the silhouette and ARAP term with 100.0 and 1500.0, respectively and used a learning rate of 0.0001. This step takes 20 hours using 4 NVIDIA Quadro RTX 8000 with 48GB of memory. The lighting is optimized with a batch size of 4, a learning rate of 0.0001, and 30,000 iterations. This takes around 7 hours. For training DeltaNet, we balanced the chroma, silhouette, and Laplacian loss with 0.03775, 500.0, and 100,000.0, respectively. Again, we train for 360,000 iterations using a batch size of 8 and a learning rate of 0.0001 which takes 2 days. Finally, for training TexNet we use a batch size of 12 and a learning rate of 0.0001. We iterate 720,000 times which takes 4 days.

4 RESULTS

All our results are computed on a machine with an AMD EPYC 7502P processing unit and a Nvidia Quadro RTX 8000 graphics card. Our approach can run at 38 frames per second (fps) at inference time and therefore allows interactive applications as discussed later. For the first frame of a test sequence, we copy over the pose of the first frame as the "previous frames" of the motion window as there are no real previous frames.

4.1 Dataset

We created a new dataset, called *DynaCap*, which consists of 5 sequences containing 4 subjects wearing 5 different types of apparel, e.g., trousers and skirts (see Fig. 6). Each sequence is recorded at 25fps and is split into a training and testing recording which contain around 20,000 and 7000 frames, respectively. The training and test motions are significantly different from each other. Following common practice, we acquired separate recordings for training and testing (instead of randomly sampling from a single sequence). For each sequence, we asked the subject to perform a wide range of motions like “dancing” which was freely interpreted by the subject. We recorded with 50 to 101 synchronized and calibrated cameras at a resolution of 1285 × 940. Further, we scanned each person to acquire a 3D template, as described in Sec. 3.1, which is rigged to a skeleton. For all sequences, we estimated the skeletal motion using [TheCaptury 2020] and segmented the foreground using color keying. We will release the new dataset, as there are no other datasets available that target exactly such a setting, namely a single actor captured for a large range of motions and with such a dense camera setup. Our dataset can be particularly interesting for dynamic neural scene representation approaches and can serve as a benchmark.

In addition, we use the subjects *S1*, *S2*, and *S4* of the publicly available DeepCap dataset [Habermann et al. 2020] who wear trousers, T-shirts, skirts, and sleeves to evaluate our method also on external data which has a sparser camera setup. The dataset comes along with ground truth pose tracking, calibrated, segmented, and synchronized multi-view imagery, in addition to a rigged template mesh. Their dataset contains between 11 and 14 camera views at a resolution of 1024 × 1024 and a frame rate of 50fps.

Fig. 6. DynaCap dataset. We recorded 5 subjects wearing different types of apparel with multiple calibrated and synchronized cameras. Further, we capture a 3D template mesh and rig it to a skeleton. For each frame, we compute the ground truth 3D skeletal pose as well as ground truth foreground segmentation.

4.2 Qualitative Results

In Fig. 7, we illustrate results for all 8 sequences showing different types of apparel. Again, note that our method learns videorealistic motion- and view-dependent dynamic surface deformation, including also deformations of loose apparel (such as the skirt and dress in Fig. 7), without requiring a physics simulation, and texture *only* from multi-view imagery and does *not* require any dense 3D data such as depth maps, point clouds or registered meshes for supervision. Our approach works not only well for tighter clothes such as pants but also for more dynamic ones like skirts. We demonstrate that our approach can create videorealistic results for *unseen* very challenging and fast motions, e.g., jumping jacks (see also supplemental video). Moreover, the texture is consistent while changing the viewpoint (images without green screen background), which shows that our view conditioned TexNet also generalizes to novel view points. The generalization comes from the fact that our networks for deformation regression as well as for texture regression focus on local configurations rather than the full 3D body motion. However, the network still allows global reasoning but this effect is dampened by the network design. Technically, this is accomplished by the local graph/image features and the graph/image convolutions. Further, note the view-dependent effects like reflections on the skin and clothing (second and third column where we keep the pose fixed and only change the view point). Given an image of the empty capture scene (images with green screen background), our approach allows augmenting the empty background image with our results to produce realistic looking images.

Fig. 8 shows that our predicted geometry (on test data) precisely overlays to the corresponding image which demonstrates that our approach generalizes well to unseen motions. Importantly, the ground truth frame showing the actor is *not* an input to our method as our method only takes the skeletal motion which we extracted from the video. We also show our textured result overlayed onto the ground truth. Our TexNet generalizes well to unseen motions, captures the motion-dependent dynamics and looks photo-realistic as ground truth and our rendered result look almost identical.

Fig. 7. Qualitative results. From left to right. Input testing pose and our posed, deformed and textured result shown from an arbitrary viewpoint. Our result for another testing pose and viewpoint. The same pose as in second column but rendered from a different view point. Note the view-dependent appearance change on the skin and clothing due to view-dependent reflections. The last two columns show testing poses but viewed from the training camera viewpoints. This allows augmenting the empty background that we captured for each camera with our result.

4.3 Comparison

Only few people in the research community have targeted creating video realistic characters from multi-view video that can be controlled to perform unseen skeleton motions. To the best of our knowledge, there are only three previous works [Casas et al. 2014; Shysheya et al. 2019; Xu et al. 2011] that also assume multi-view video data for building a controllable and textured character. However, these works do not provide their code and thus are hard to compare to. Moreover, they either do not share their data [Shysheya et al. 2019] or the publicly available sequences are too short for training our approach [Casas et al. 2014; Xu et al. 2011] and as well lack a textured template which our method assumes as given. Therefore in Tab. 1, we resort to a conceptual comparison showing the advantage of our method as well as an extensive visual comparison in the supplemental video where we recreate similar scenes.

The earlier works of [Xu et al. 2011] and [Casas et al. 2014] are both non learning-based and instead use texture retrieval to synthesize dynamic textures. In contrast to our approach, they both suffer from the fact that their geometry is either fully driven by skinning based deformations [Xu et al. 2011] or by motion graphs [Casas et al. 2014]. Thus, they cannot model motion-dependent geometric deformations and fail to model plausible dynamics of loose apparel, as our method can do it. Moreover, as they rely on retrieval-based techniques, their approaches do not generalize well to motions different from motions in the dataset. Furthermore, the retrieval is expensive to compute, making real time application impossible. In contrast,

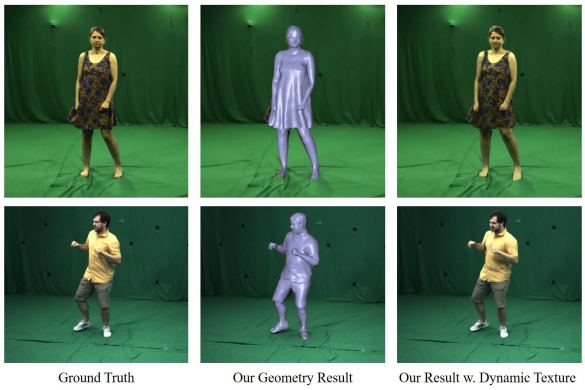


Fig. 8. Our geometry and texture networks generalize well to unseen motions as the geometry overlays nicely onto the ground truth frame and the final textured result looks almost identical to the ground truth. Importantly, our method does *not* take the ground truth frame as an input as it only takes the unseen motion from the video.

our approach leverages dedicated geometry networks (EGNet and DeltaNet) which predict motion-dependent geometry deformations for both tight and loose apparel. Further, our approach enables animation and control in real-time, and generalizes well to unseen motions (see supplemental video and Fig. 7).

More recently, Textured Neural avatars [Shysheya et al. 2019] was proposed as the first learning based approach for creating controllable and textured characters using multi-view data. In contrast to our approach, they do not model geometry explicitly but use DensePose [Güler et al. 2018] as a geometric proxy in image space. As a consequence, their approach does not provide space-time coherent geometry as well as motion-dependent surface deformation which is important in most graphics and animation settings. Moreover, they recover a static texture during training which prevents modelling motion- and view-dependent effects.

Further, our supplemental video shows that our approach is a significant step forward in terms of visual quality compared to previous methods, as they either suffer from temporal inconsistency, sudden jumps in the texture originating from the texture retrieval, and missing body parts as geometry is not modelled explicitly.

4.4 Quantitative Evaluation

4.4.1 Geometry. To evaluate our approach in terms of geometry, we leverage the challenging *S4* testing sequence (11,000 frames) of the DeepCap dataset [Habermann et al. 2020] shown in the top row of Fig. 8. We trained our model on the corresponding multi-view training sequence and used their mesh template. We follow the evaluation procedure described in the original paper. Therefore, we measure the multi-view foreground mask overlap between ground truth foreground segmentation and the foreground mask obtained from our projected and deformed model on all available views (AMVIoU) averaged over every 100th frame.

In Tab. 2, we compare to the multi-view baseline implementation of [Habermann et al. 2020], referred to as MVBL. Here, they perform optimization-based multi-view pose and surface fitting using sparse and dense image cues, e.g., 2D joint predictions and the foreground masks. Importantly, they apply this on the *testing* sequence directly whereas our method only takes the skeletal motion without even seeing the multi-view imagery. Nonetheless, our results are more accurate than MVBL. We found that their sequential optimization of pose and deformation can fall into erroneous local optima, resulting in worse overlay. In contrast, our method benefits from the randomness of the stochastic gradient descent and the shuffling of data which reduces the likelihood of getting stuck in local optima. We also compared our poseable and dynamic representation to the classical Dual Quaternion character skinning [Kavan et al. 2007] where we use the same poses as used for our approach to animate the rigged character. Skinning can merely approximate skeleton induced surface deformation, but it fails to represent dynamic clothing deformations, as we can handle them. Thus, we clearly outperform their approach as they cannot account for the surface deformation caused by the motion of the actor, e.g. swinging of a skirt.

We report the same metrics also on the training data. Even from a reconstruction perspective our method produces accurate results during training and the proposed representation is able to fit the image observations almost perfectly. Notably, there is only a small accuracy difference between training and testing performance. This confirms that our approach generalizes well to unseen motions.

4.4.2 Texture. In Tab. 3, we evaluate the realism of our motion-dependent dynamic texture on the same sequence as before (testing sequence of *S4*). We again trained on the training motion sequence of *S4* but hold out the camera 4 as a test view. We evaluate our approach on *Train Camera* 0 and *Test Camera* 4 for *Train Motions* and *Test Motions*. Therefore, we compute the mean squared image error (MSE) and the structural similarity index measure (SSIM) between the rendered model and the ground truth multi-view images averaged over every 100th frame where we masked out the background as our approach does not synthesize the background. Our method produces visually plausible results for novel motions rendered from a training view (see top row of Fig. 8 and the $4th$ and $5th$ column of the second last row of Fig. 7). But also for novel motions *and* novel camera views our approach produces video-realistic results (see $1th$, $2th$, and $3th$ column of the second last row of Fig. 7). Tab. 3 also quantitatively confirms this since all configurations of training/testing poses and camera views have a low MSE value and a high SSIM value. While there is an accuracy drop between test and train, visually the quality only decreases slightly and the absolute accuracy for each configuration is comparably high.

4.5 Ablation

4.5.1 Deformation Modules. First, we evaluate the design choices for predicting the surface deformations. Therefore, we compare the impact of the DeltaNet against only using EGNet, which we refer to as *EGNet-only*. Tab. 4 clearly shows that the additional vertex displacements improve the reconstruction accuracy as they are able to capture finer wrinkles and deformations (see also Fig. 5). While the regressed embedded deformation still performs better than a pure skinning approach, it cannot completely match the ground truth silhouettes due to the limited graph resolution causing the slightly lower accuracy compared to using the displacements.

Table 1. Conceptual comparison to previous multi-view based approaches for controllable character animation / synthesis [Casas et al. 2014; Shysheya et al. 2019; Xu et al. 2011]. Note that all previous works fall short in multiple desirable categories while our proposed approach fulfills all these requirements.

Comparison to Previous Multi-view Based Methods							
	Dyn. Geo.	Dyn. Tex.	View Dep. Effects	Controllable	Real-time	Unseen Motions	Loose Clothing
[Xu et al. 2011]	✗	✓	✓	✓	✗	✗	✗
[Casas et al. 2014]	✗	✓	✓	✓	✗	✗	✗
[Shysheya et al. 2019]	✗	✗	✗	✓	✓	✓	✗
Ours	✓	✓	✓	✓	✓	✓	✓

Table 2. Accuracy of the surface deformation. Note that we outperform the pure skinning based approach [Kavan et al. 2007] as they cannot account for dynamic cloth deformations. Our method further improves over MVBL even though this optimization based approach sees the multi-view test images. Finally, our approach performs similarly on training and testing data showing that the geometry networks generalize to unseen motions.

AMVIoU (in %) on S4 sequence	
Method	**AMVIoU↑**
MVBL [Habermann et al. 2020]	88.14
[Kavan et al. 2007]	79.45
Ours	**90.70**
Ours (Train)	94.07

Table 3. Photometric error in terms of MSE and SSIM averaged over every 100th frame. Note that our approach achieves overall low MSE results and high SSIM values. While the accuracy differs between test and train, the absolute accuracy is still comparably high and the visual quality only decreases slightly proving the generalization ability of our approach.

Photometric Error on *S4*		
Method	**MSE ↓**	**SSIM↑**
Ours (Train Motion / Train Camera)	14.79	0.99054
Ours (Train Motion / Test Camera)	31.44	0.98610
Ours (Test Motion / Train Camera)	29.00	0.98357
Ours (Test Motion / Test Camera)	43.29	0.98278

We further evaluate the impact of our SAGC over two baselines using a fully connected (FC) architecture and an unstructured graph convolutional operator [Defferrard et al. 2017] where the latter is integrated into our overall architecture and therefore just replaces the SAGC operators. We replace EGNet and DeltaNet with two fully connected networks that take the normalized motion angles as input, apply 19 fully connected layers (same depth as the proposed architecture) with nonlinear Elu activation functions, and output graph parameters and vertex displacements, respectively. As the fully connected networks have no notion of locality, they are not able to generalize well. Further, one can see that our proposed graph convolutional operation performs better than the one proposed by [Defferrard et al. 2017] because the latter shares weights across nodes while we use node specific weights which are able to encode the underlying knowledge about the individual deformation behaviours of the surface.

We also evaluate the importance of predicting the per-vertex displacements in the canonical pose space compared to predicting them in the global pose space. Note that the disentanglement of pose and deformation helps with the generalization of the network which leads to better accuracy in terms of foreground overlay.

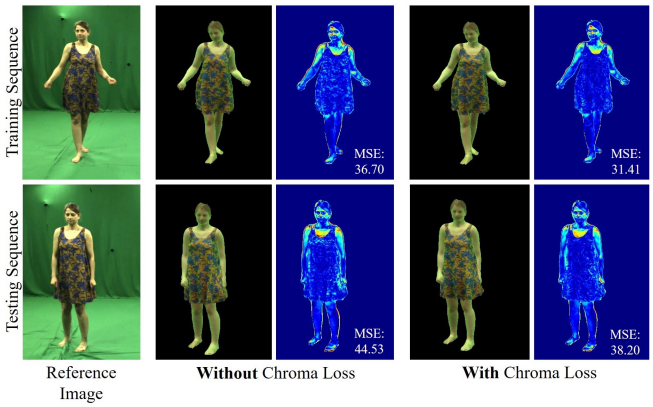


Fig. 9. Impact of the chroma loss. During training and testing, the chroma loss disambiguates drifts on the visual hull and gives more accurate results.

Finally, we evaluate the impact of the chroma loss compared to only using silhouette supervision. Note that reporting the IoU would not be meaningful as the silhouette loss alone can already ensure matching silhouettes. However, drifts along the visual hull, carved by the silhouette images, cannot be well tracked by the silhouette term alone as shown in Fig. 9. Our chroma loss penalizes these drifts, both, during training and testing leading to better results. This can be best evaluated by comparing the MSE of the deformed model with the static texture and the ground truth image as shown in Fig. 9 Here, using the chroma loss has an error of 38.20 compared to an error of 44.53 when only the silhouette loss is used during test time. This clearly shows that the chroma error can disambiguate drifts on the visual hull and thus gives more accurate results.

4.5.2 Texture Module. Next, we compare using our motion- and view-dependent texture to using a static texture rendered with and without optimized lighting. Using the static texture without optimized lighting leads to the highest error. Optimizing the light already brings the rendering globally a bit closer to the ground truth image, but still fails to represent important dynamic and view-dependent effects. By applying our dynamic texture also motion- and view-dependent texture effects can be captured, resulting in the lowest error.

4.5.3 Amount of Data. Finally, we evaluate the influence of the number of training cameras for the *OlekDesert* sequence in Tab. 5. We tested training with 5, 10, 25 and 49 cameras placed around the scene in a dome-like arrangement. We used the respective test

Table 4. Ablation study. We evaluate the design choices for the geometry networks and texture networks. Note that we beat the baselines in all aspects confirming that our design choices indeed lead to an improvement.

Ablation on the S4 sequence		
Method	**AMVIoU↑**	**MSE↓**
EGNet-only	87.89	—
Fully Connected	87.48	—
Unstructured GraphConv	83.30	—
Global Pose Space	89.81	—
Without Lighting and Dynamic Texture	—	176.99
Without Dynamic Texture	—	60.50
Ours	**90.70**	**43.29**

Table 5. Influence of the number of available training cameras. Already with few cameras our method achieves plausible results. However, adding more cameras further improves the quality of both geometry and texture.

Ablation on the OlekDesert sequence		
Method	**AMVIoU↑**	**MSE↓**
5 camera views	90.27	20.85
10 camera views	90.34	19.32
25 camera views	90.36	17.49
Ours (49 views)	**90.68**	**16.72**

motions for all reported metrics. For computing the MSE, we chose camera 46 which was not part of the training views for all experiments. Note that already 5 cameras can lead to plausible results. Interestingly, with such a sparse setup our approach still produces coherent results for unseen viewpoints as the prediction is in canonical texture space, which implicitly regularizes the predictions, leading to a better generalization ability. However, adding more cameras further improves both geometry and texture quality.

4.6 Applications

As shown in Fig. 10, our method can be used in several applications such as motion re-targeting where a source actor (blue dress girl) drives our character model (red shirt girl). Further, our method synthesizes new free-viewpoint videos of an actor only with a driving motion sequence. Moreover, we implemented an interactive interface, where the user can freely change the skeletal pose and 3D camera viewpoint and our method produces the posed, deformed, and texture geometry in real time.

4.7 Limitations

Our approach approximates clothing dynamics in a data-driven and plausible way, but actual physics-based clothing animation may still lead to further improved results. In future research, this could be handled by employing those physics-based priors in the learning process or even at inference. Further, our method cannot handle apparent topological changes such as taking off pieces of apparel. We believe the current progress in implicit representations combined with our representation could help to generate such changes even though they are radically different from the initial template mesh. We do not track the facial expression and hands. 2D face landmark trackers as well as hand trackers could be used to also track hands and face so that they can also be controlled in the deep dynamic character. Currently, the training time of the network modules is quite long. In the future, more efficient training schemes could be explored to solve this issue. Moreover, we rely on good foreground segmentation results. In consequence, our method might receive a wrong silhouette supervision when multiple people or other moving objects, which are detected as foreground, are in the training scene. Explicitly modeling multi-person scenes and using a learning based multi-person detector could help here. Finally, severely articulated poses like a hand stand, which are not within the training motion distribution, can lead to wrong deformation and texture predictions.

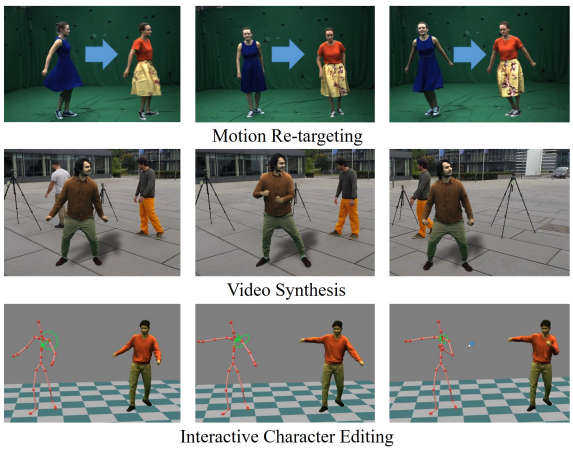


Fig. 10. Applications. Our method can be used in several applications such as motion re-targeting, neural video synthesis, and interactive character editing. Note that for all these applications, our method produces videorealistic results creating an immersive experience.

5 CONCLUSION

We presented a real-time method that allows to animate the dynamic 3D surface deformation and texture of highly realistic 3D avatars in a user-controllable way. Skeleton motion can be freely controlled and avatars can be free-viewpoint rendered from any 3D viewpoint. To this end, we propose a learning based architecture which not only regresses dynamic surface deformations but also dynamic textures. Our approach does not require any ground truth 3D supervision. Instead, we only need multi-view imagery and employ new analysis-by-synthesis losses for supervision. Our results outperform the state of the art in terms of surface detail and textural appearance and therefore the high visual quality of our animations opens up new possibilities in video-realistic character animation, controllable free-viewpoint video, and neural video synthesis.

ACKNOWLEDGMENTS

All data captures and evaluations were performed at MPII by MPII. The authors from MPII were supported by the ERC Consolidator Grant 4DRepLy (770784), the Deutsche Forschungsgemeinschaft (Project Nr. 409792180, Emmy Noether Programme, project: Real Virtual Humans) and Lise Meitner Postdoctoral Fellowship.

REFERENCES

Kfir Aberman, Mingyi Shi, Jing Liao, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. 2019. Deep Video-Based Performance Cloning. *Comput. Graph. Forum* 38, 2 (2019), 219–233. https://doi.org/10.1111/cgf.13632

Thiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar, Christian Theobalt, and Gerard Pons-Moll. 2019. Learning to Reconstruct People in Clothing from a Single RGB Camera. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1175–1186.

Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. 2018a. Detailed Human Avatars from Monocular Video. In *International Conference on 3D Vision*. 98–109. https://doi.org/10.1109/3{DV}.2018.00022

Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. 2018b. Video Based Reconstruction of 3D People Models. In *IEEE Conference on Computer Vision and Pattern Recognition*. CVPR Spotlight Paper.

Stephen W. Bailey, Dave Otte, Paul Dilorenzo, and James F. O'Brien. 2018. Fast and Deep Deformation Approximations. *ACM Transactions on Graphics* 37, 4 (Aug. 2018), 119:1–12. https://doi.org/10.1145/3197517.3201300 Presented at SIGGRAPH 2018, Los Angeles.

Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. 2019. Multi-Garment Net: Learning to Dress 3D People from Images. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE.

Blender 2020. Blender. https://www.blender.org/.

Gunilla Borgefors. 1986. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing* 34, 3 (1986), 344 – 371. https://doi.org/10.1016/S0734-189X(86)80047-0

Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. 2003. Free-viewpoint Video of Human Actors. *ACM Trans. Graph.* 22, 3 (July 2003).

Dan Casas, Marco Volino, John Collomosse, and Adrian Hilton. 2014. 4D Video Textures for Interactive Character Appearance. *Comput. Graph. Forum* 33, 2 (May 2014), 371–380. https://doi.org/10.1111/cgf.12296

Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. 2019. Everybody Dance Now. In *International Conference on Computer Vision (ICCV)*.

Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. 2019. Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In *Advances In Neural Information Processing Systems*.

Hongsuk Choi, Gyeongsik Moon, and Kyoung Mu Lee. 2020. Pose2Mesh: Graph Convolutional Network for 3D Human Pose and Mesh Recovery from a 2D Human Pose. In *European Conference on Computer Vision (ECCV)*.

Kwang-Jin Choi and H. Ko. 2005. Research problems in clothing simulation. *Comput. Aided Des.* 37 (2005), 585–592.

Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 69.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2017. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. arXiv:1606.09375 [cs.LG]

Patrick Esser, Johannes Haux, Timo Milbich, and Bj orn Ommer. 2018. Towards Learning a Realistic Rendering of Human Behavior. In *The European Conference on Computer Vision (ECCV) Workshops*.

Wei-Wen Feng, Yizhou Yu, and Byung-Uck Kim. 2010. A Deformation Transformer for Real-Time Cloth Animation. *ACM Trans. Graph.* 29, 4, Article 108 (July 2010), 9 pages. https://doi.org/10.1145/1778765.1778845

Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2020. Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction. arXiv:2012.03065 [cs.CV]

Peng Guan, Loretta Reiss, David A. Hirshberg, Alexander Weiss, and Michael J. Black. 2012. DRAPE: DRessing Any PErson. *ACM Trans. Graph.* 31, 4, Article 35 (July 2012), 10 pages. https://doi.org/10.1145/2185520.2185531

Riza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. 2018. DensePose: Dense Human Pose Estimation In The Wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Erhan Gundogdu, Victor Constantin, Amrollah Seifoddini, Minh Dang, Mathieu Salzmann, and Pascal Fua. 2019. Garnet: A Two-stream Network for Fast and Accurate 3D Cloth Draping. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE.

Marc Habermann, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. 2019. LiveCap: Real-time Human Performance Capture from Monocular Video. *ACM Trans. Graph.* (2019).

Marc Habermann, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. 2020. DeepCap: Monocular Human Performance Capture Using Weak Supervision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Fabian Hahn, Bernhard Thomaszewski, Stelian Coros, Robert W. Sumner, Forrester Cole, Mark Meyer, Tony DeRose, and Markus Gross. 2014. Subspace Clothing Simulation Using Adaptive Bases. *ACM Trans. Graph.* 33, 4, Article 105 (July 2014), 9 pages. https://doi.org/10.1145/2601097.2601160

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Anna Hilsmann, Philipp Fechteler, Wieland Morgenstern, Wolfgang Paier, Ingo Feldmann, Oliver Schreer, and Peter Eisert. 2020. Going beyond free viewpoint: creating animatable volumetric video of human performances. *IET Computer Vision* 14, 6 (Sep 2020), 350–358. https://doi.org/10.1049/iet-cvi.2019.0786

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-Functioned Neural Networks for Character Control. *ACM Trans. Graph.* 36, 4, Article 42 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073663

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2016. Image-to-Image Translation with Conditional Adversarial Networks. *CoRR* abs/1611.07004 (2016). arXiv:1611.07004 http://arxiv.org/abs/1611.07004

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *CVPR* (2017).

Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. 2020. SDFDiff: Differentiable Rendering of Signed Distance Fields for 3D Shape Optimization. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ning Jin, Yilin Zhu, Zhenglin Geng, and Ronald Fedkiw. 2018. A Pixel-Based Framework for Data-Driven Clothing. arXiv:1812.01677 [cs.CV]

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3D Mesh Renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with dual quaternions. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. ACM, 39–46.

Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Nießner, Patrick Pérez, Christian Richardt, Michael Zollöfer, and Christian Theobalt. 2018a. Deep Video Portraits. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 163.

Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Nießner, Patrick Pérez, Christian Richardt, Michael Zollöfer, and Christian Theobalt. 2018b. Deep Video Portraits. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 163.

Tae-Yong Kim and Eugene Vendrovsky. 2008. DrivenShape: A Data-Driven Approach for Shape Deformation. In *ACM SIGGRAPH 2008 Talks* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 69, 1 pages. https://doi.org/10.1145/1401032.1401121

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (12 2014).

Zorah Lähner, Daniel Cremers, and Tony Tung. 2018. DeepWrinkles: Accurate and Realistic Clothing Modeling. *ArXiv* abs/1808.03417 (2018).

J. H. Lambert. 1760. Photometria sive de mensure de gratibus luminis, colorum umbrae. In *Photometria sive de mensure de gratibus luminis, colorum umbrae, Eberhard Klett.*

Guannan Li, Yebin Liu, and Qionghai Dai. 2014. Free-viewpoint Video Relighting from Multi-view Sequence Under General Illumination. *Mach. Vision Appl.* 25, 7 (Oct. 2014), 1737–1746. https://doi.org/10.1007/s00138-013-0559-0

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. 2020. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *https://arxiv.org/abs/2011.13084* (2020).

Junbang Liang, Ming C. Lin, and Vladlen Koltun. 2019. Differentiable Cloth Simulation for Inverse Problems. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020a. Neural Sparse Voxel Fields. *NeurIPS* (2020).

Lingjie Liu, Weipeng Xu, Marc Habermann, Michael Zollhoefer, Florian Bernard, Hyeongwoo Kim, Wenping Wang, and Christian Theobalt. 2020b. Neural Human Video Rendering by Learning Dynamic Textures and Rendering-to-Video Translation. arXiv:2001.04947 [cs.GR]

Lingjie Liu, Weipeng Xu, Michael Zollhöfer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. 2019b. Neural Rendering and Reenactment of Human Actor Videos. *ACM Trans. Graph.* 38, 5, Article 139 (Oct. 2019), 14 pages. https://doi.org/10.1145/3333002

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019a. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. *The IEEE International Conference on Computer Vision (ICCV)* (Oct 2019).

Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. 2018. Deep appearance models for face rendering. *ACM Transactions on Graphics* 37, 4 (Aug 2018), 1–13. https://doi.org/10.1145/3197517.3201401

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 65.

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

Matthew M. Loper and Michael J. Black. 2014. OpenDR: An Approximate Differentiable Renderer. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 154–169.

Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. 2017. Pose guided person image generation. In *Advances in Neural Information Processing Systems*. 405–415.

Liqian Ma, Qianru Sun, Stamatios Georgoulis, Luc van Gool, Bernt Schiele, and Mario Fritz. 2018. Disentangled Person Image Generation. *Computer Vision and Pattern Recognition (CVPR)* (2018).

Qianli Ma, Jinlong Yang, Anurag Ranjan, Sergi Pujades, Gerard Pons-Moll, Siyu Tang, and Michael Black. 2020. Learning to Dress 3D People in Generative Clothing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

N. Magnenat-Thalmann, A. Laperri'ere, and D. Thalmann. 1988. Joint-Dependent Local Deformations for Hand Animation and Object Grasping. In *Proceedings of Graphics Interface '88* (Edmonton, Alberta, Canada) *(GI '88)*. Canadian Man-Computer Communications Society, Toronto, Ontario, Canada, 26–33. http://graphicsinterface.org/wp-content/uploads/gi1988-4.pdf

Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, and Sean Fanello. 2018. LookinGood: Enhancing Performance Capture with Real-Time Neural Re-Rendering. *ACM Trans. Graph.* 37, 6, Article 255 (Dec. 2018), 14 pages. https://doi.org/10.1145/3272127.3275099

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Claus Mueller. 1966. Spherical Harmonics. In *Spherical Harmonics, Springer*.

Rahul Narain, Armin Samii, and James F. O'Brien. 2012. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Transactions on Graphics* 31, 6 (Nov. 2012), 147:1–10. http://graphics.berkeley.edu/papers/Narain-AAR-2012-11/ Proceedings of ACM SIGGRAPH Asia 2012, Singapore.

A. Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and M. Carlson. 2005. Physically based deformable models in computer graphics. *Eurographics: State of the Art Report* (01 2005), 71–94.

Keunhong Park, Utkarsh Sinha, Jonathan Barron, Sofien Bouaziz, Dan Goldman, Steven Seitz, and Ricardo Martin-Brualla. 2020. Deformable Neural Radiance Fields. *https://arxiv.org/abs/2011.12948* (2020).

Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. 2020. 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.

Photo Scan 2016. PhotoScan. http://www.agisoft.com.

Juan Pineda. 1988. A Parallel Algorithm for Polygon Rasterization. *SIGGRAPH Comput. Graph.* 22, 4 (June 1988), 17–20. https://doi.org/10.1145/378456.378457

Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael Black. 2017. ClothCap: Seamless 4D Clothing Capture and Retargeting. *ACM Transactions on Graphics, (Proc. SIGGRAPH)* 36, 4 (2017). http://dx.doi.org/10.1145/3072959.3073711

Albert Pumarola, Antonio Agudo, Alberto Sanfeliu, and Francesc Moreno-Noguer. 2018. Unsupervised Person Image Synthesis in Arbitrary Poses. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2020. D-NeRF: Neural Radiance Fields for Dynamic Scenes. *https://arxiv.org/abs/2011.13961* (2020).

Amit Raj, Michael Zollhoefer, Tomas Simon, Jason Saragih, Shunsuke Saito, James Hays, and Stephen Lombardi. 2020. PVA: Pixel-aligned Volumetric Avatars. In *arXiv:2101.02697*.

Igor Santesteban, Miguel A. Otaduy, and Dan Casas. 2019. Learning-Based Animation of Clothing for Virtual Try-On. *Comput. Graph. Forum* 38 (2019), 355–366.

Kripasindhu Sarkar, Dushyant Mehta, Weipeng Xu, Vladislav Golyanik, and Christian Theobalt. 2020. Neural Re-Rendering of Humans from a Single Image. In *European Conference on Computer Vision (ECCV)*.

Aliaksandra Shysheya, Egor Zakharov, Kara-Ali Aliev, Renat Bashirov, Egor Burkov, Karim Iskakov, Aleksei Ivakhnenko, Yury Malkov, Igor Pasechnik, Dmitry Ulyanov, Alexander Vakhitov, and Victor Lempitsky. 2019. Textured Neural Avatars. arXiv:1905.08776 [cs.CV]

Chenyang Si, Wei Wang, Liang Wang, and Tieniu Tan. 2018. Multistage Adversarial Losses for Pose-Based Human Image Synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Aliaksandr Siarohin, Enver Sangineto, Stephane Lathuiliere, and Nicu Sebe. 2018. Deformable GANs for Pose-based Human Image Generation. In *CVPR 2018*.

Yinghao Xu Qianqian Wang Qing Shuai Hujun Bao Xiaowei Zhou Sida Peng, Yuanqing Zhang. 2020. Neural Body: Implicit Neural Representations with Structured Latent Codes for Novel View Synthesis of Dynamic Humans. *arXiv preprint arXiv:2012.15838* (2020).

Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. 2019a. DeepVoxels: Learning Persistent 3D Feature Embeddings. In *Computer Vision and Pattern Recognition (CVPR)*.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. 2019b. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In *Advances in Neural Information Processing Systems*.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible Surface Modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Barcelona, Spain) *(SGP '07)*. Eurographics Association.

S. Starke, H. Zhang, T. Komura, and J. Saito. 2019. Neural state machine for character-scene interactions. *ACM Transactions on Graphics (TOG)* 38 (2019), 1 – 14.

Carsten Stoll, Juergen Gall, Edilson de Aguiar, Sebastian Thrun, and Christian Theobalt. 2010. Video-Based Reconstruction of Animatable Human Characters. *ACM Trans. Graph.* 29, 6, Article 139 (Dec. 2010), 10 pages. https://doi.org/10.1145/1882261.1866161

Zhaoqi Su, Weilin Wan, Tao Yu, Lingjie Liu, Lu Fang, Wenping Wang, and Yebin Liu. 2020. MulayCap: Multi-layer Human Performance Capture Using A Monocular Video Camera. *IEEE Transactions on Visualization and Computer Graphics* (2020), 1–1. https://doi.org/10.1109/tvcg.2020.3027763

Robert W. Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded Deformation for Shape Manipulation. *ACM Trans. Graph.* 26, 3 (July 2007).

Min Tang, tongtong wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. 2018. I-Cloth: Incremental Collision Handling for GPU-Based Interactive Cloth Simulation. *ACM Trans. Graph.* 37, 6, Article 204 (Dec. 2018), 10 pages. https://doi.org/10.1145/3272127.3275005

Yu Tao, Zerong Zheng, Yuan Zhong, Jianhui Zhao, Dai Quionhai, Gerard Pons-Moll, and Yebin Liu. 2019. SimulCap : Single-View Human Performance Capture with Cloth Simulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

TheCaptury 2020. The Captury. http://www.thecaptury.com/.

Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred neural rendering: image synthesis using neural textures. *ACM Transactions on Graphics* 38 (2019).

Treedys 2020. Treedys. https://www.treedys.com/.

Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. 2020. Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Deforming Scene from Monocular Video. *https://arxiv.org/abs/2012.12247* (2020).

Marco Volino, Dan Casas, John Collomosse, and Adrian Hilton. 2014. Optimal Representation of Multiple View Video. In *Proceedings of the British Machine Vision Conference*. BMVA Press.

Huamin Wang, Florian Hecht, Ravi Ramamoorthi, and James F. O'Brien. 2010. Example-Based Wrinkle Synthesis for Clothing Animation. *ACM Trans. Graph.* 29, 4, Article 107 (July 2010), 8 pages. https://doi.org/10.1145/1778765.1778844

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018a. Video-to-Video Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*. 1152––1164.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018b. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *CVPR*.

Ziyan Wang, Timur Bagautdinov, Stephen Lombardi, Tomas Simon, Jason Saragih, Jessica Hodgins, and Michael ZollhÃ¶fer. 2020. Learning Compositional Radiance Fields of Dynamic Human Heads. arXiv:2012.09955 [cs.CV]

Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. 2020. Space-time Neural Irradiance Fields for Free-Viewpoint Video. *https://arxiv.org/abs/2011.12950* (2020).

Feng Xu, Yebin Liu, Carsten Stoll, James Tompkin, Gaurav Bharaj, Qionghai Dai, Hans-Peter Seidel, Jan Kautz, and Christian Theobalt. 2011. Video-based Characters: Creating New Human Performances from a Multi-view Video Database. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. ACM, New York, NY, USA, Article 32, 10 pages. https://doi.org/10.1145/1964921.1964927

Weiwei Xu, Nobuyuki Umentani, Qianwen Chao, Jie Mao, Xiaogang Jin, and Xin Tong. 2014. Sensitivity-Optimized Rigging for Example-Based Real-Time Clothing Synthesis. *ACM Trans. Graph.* 33, 4, Article 107 (July 2014), 11 pages. https://doi.org/10.1145/2601097.2601136

Jae Shin Yoon, Lingjie Liu, Vladislav Golyanik, Kripasindhu Sarkar, Hyun Soo Park, and Christian Theobalt. 2020. Pose-Guided Human Animation from a Single Image in the Wild. arXiv:2012.03796 [cs.CV]

Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020a. NERF++: Analyzing and Improving Neural Radiance Fields. *https://arxiv.org/abs/2010.07492* (2020).

Meng Zhang, Tuanfeng Wang, Duygu Ceylan, and Niloy J. Mitra. 2020b. Deep Detail Enhancement for Any Garment. arXiv:2008.04367 [cs.GR]

Tiancheng Zhi, Christoph Lassner, Tony Tung, Carsten Stoll, Srinivasa G. Narasimhan, and Minh Vo. 2020. TexMesh: Reconstructing Detailed Human Texture and Geometry from RGB-D Video. arXiv:2008.00158 [cs.CV]

C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. 2004. High-quality video view interpolation using a layered representation. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 600–608.

J. S. Zurdo, J. P. Brito, and M. A. Otaduy. 2013. Animating Wrinkles by Example on Non-Skinned Cloth. *IEEE Transactions on Visualization and Computer Graphics* 19, 1 (2013), 149–158.