Dynamic Surface Animation using Generative Networks

João Regateiro Adrian Hilton Marco Volino
Centre for Vision, Speech and Signal Processing
University of Surrey
{j.regateiro, a.hilton, m.volino} @surrey.ac.uk

Abstract

This paper presents techniques to animate realistic human-like motion using a compressed learnt model from 4D volumetric performance capture data. Sequences of 4D dynamic geometry representing a human performing an arbitrary motion are encoded through a generative network into a compact space representation, whilst maintaining the original properties, such as, surface dynamics. An animation framework is proposed which computes an optimal motion graph using the novel capabilities of compression and generative synthesis properties of the network. This approach significantly reduces the memory space requirements, improves quality of animation, and facilitates the interpolation between motions. The framework optimises the number of transitions in the graph with respect to the shape and motion of the dynamic content. This generates a compact graph structure with low edge connectivity, and maintains realism when transitioning between motions. Finally, it demonstrates that generative networks facilitate the computation of novel poses, and provides a compact motion graph representation of captured dynamic shape enabling real-time interactive animation and interpolation of novel poses to smoothly transition between motions.

1. Introduction

Dynamic surface capture is an emerging media that keeps evolving into new concepts and areas, allowing the research community to advance in areas such as, medical imaging, computer vision, automotive and most popular the creative industry. Dynamic surfaces are produced from 4D performance capture studios which generally consist of a set of synchronised cameras that simultaneously record a performance [9, 13, 14, 30]. The generated content usually consists of sequences of 3D geometry that integrate all visual features of the scene, for example, the shape, motion and appearance. This allows replay of the performance from any viewpoint and moment in time, although it requires a huge computational effort to process and store 3D dynamic geometry.

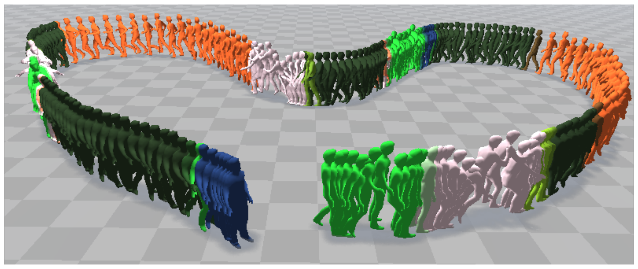
Figure 1. Character animation using a generative network to synthesise and interpolate motion sequences. The colours represent different motion sequences.

Recent advances have allowed this content to be more manageable when manipulated [8, 28], hence acquiring the potential to revolutionise digital content production. Performance capture studio content has been exploited on virtual character motion retargeting [2] through manipulation of key-points to create novel motions. Interactive character animation pipelines usually consist of two stages, modelling a parameterised 3D character and rigging an appropriated skeletal structure to match the 3D shape of the character model. This process requires a high level of skill, and manual editing to create plausible human-like motions and maintain realistic appearance.

3D characters are commonly achieved with a high-resolution 3D laser scan and skeletal structures from artists which manually adjust skeletal joint positions, the novel animations are often generated using inverse dynamic and kinematic solvers on a skeletal motion capture database [16, 25, 32, 38]. The introduction of motion graphs [1, 21, 22] and parameterised skeletal motion spaces [15, 29] allowed representation and real-time interactive control of character movement from performance capture data.

This work proposes an animation framework to generate realistic human-like motion from temporally consistent reconstructions of dynamic mesh sequences [8, 28]. The proposed approach learns an efficient compressed representation of 3D dynamic geometry through a variational autoencoder network. Finally, it builds an optimised motion graph capable of reproducing smooth and realistic motions using the learned representation of dynamic surfaces. We demonstrate that the learnt model is capable of interpolating and generating novel posed surface geometry, and can be integrated into most animation pipelines as a compact representation for real-time animation from 4D volumetric performance capture data.

The open challenges that must be addressed for widespread adoption of this technology are: (i) compact representation of 3D dynamic geometry; (ii) ability to efficiently generate and control animation in real-time with the flexibility of conventional computer graphics and the dynamic shape detail and realism of 4D performance capture data. The contributions of this paper can be summarised as follows:

- An efficient representation of 4D mesh sequences using variational autoencoders, which exploits the latent space to allow manipulation and generation of novel content.

- A motion graph framework integrated with a variational autoencoder that allows reuse, editing and synthesis of novel motions from performance capture data, allowing generation of transition between captured sequences.

2. Related Work

4D Volumetric Video: has been an active area of research [9, 13, 14, 30], that has emerged to address the increasing demands for realistic content. Recently, Collet *et al.* [13] presented a full pipeline to capture, reconstruct and replay volumetric video. It consists of a set of synchronised cameras that simultaneously capture the scene from different viewpoints.

The benefit of volumetric video is that it captures the full 4D dynamic surface geometry and appearance of a subject simultaneously. This unlocks an enormous creative potential for highly realistic animated content production based on captured performance. Recent research provides frameworks to ease the manipulation of this content [7, 8, 28, 35], allowing an artist to perform manual adjustments on 4D dynamic geometry and combine multiple sequences in a motion graph. The limitations are due to data complexity and dimensionality, making it difficult to manipulate and propagate changes across multiple sequences, and difficult to maintain the realism of captured human and clothing dynamics.

Motion Graphs: Molina-Tanco and Hilton [34] proposed the use of structured graph of skeletal motion sequences for controlling digital character animation. This was constructed using a frame-to-frame similarity metric which identifies similar poses. The similarity between frames can be represented as a directed graph, where nodes represent the pose at a time instance and edges the transitions. The concept of motion graphs has been applied to volumetric video using both structured meshes (temporally consistent) [4, 12] and unstructured meshes [27].

Starck *et al.* [31] and Huang *et al.* [17] proposed to construct motion graphs for dynamic surfaces, not requiring pre-process alignment techniques. Prada *et al.* [27] instead ensures at defined transitions points, mesh alignment is performed to provide a smooth blending between frames and optical flow is used to give a ghost free texture. This overcomes the challenging problem of global mesh alignment and only considers alignment of geometry and texture where necessary. In contrast, Boukhayma *et al.* [4] and Casas *et al.* [12] leverage the correspondence of the structured mesh sequences to find pose transitions. These works demonstrate that motion graphs can be used to interpolate between related motions. Boukhayma *et al.* [4] shows an optimal graph technique to automatically interpreted structured mesh sequences. This was demonstrated to maintain realism between motion transitions via a novel transition cost function that weights its realism, and an optimised motion graph that keeps the minimum necessary transitions between frame sequences.

Learnt Deep Mesh Representations: Tan *et al.* [33] use a variational autoencoder to learn a representation of parameterised dynamic shapes. Their network trains on a preprocessed feature space of the training data, demonstrating very low reconstruction error for the ground truth shapes. Lombardi *et al.* [23] proposed a learnt model of shape and appearance conditioned on viewpoint allowing recovery of view-dependent texture detail. This network demonstrates the ability to learn 3D dynamic shapes from vertices, avoiding the need to pre-process information. This demonstrates the real-time capabilities of variational autoencoders, being able to decode shape and appearance in less then 5 milliseconds.

The method outlined in this paper overcomes the need to explicitly load full sequences of 4D dynamic shapes, which are often computationally and memory expensive [18, 19]. The use of a variational autoencoder for learning allows us to synthesise plausible novel posed meshes which have not been observed in the captured mesh sequence. During training, our network learns a latent representation of posed mesh geometry. This enables manipulation of dynamic surfaces via a low dimensional representation. This is useful for generalisation to novel motions and also for temporal upscalling. We demonstrate that this learnt representation can be used as the basis for real-time mesh animation from multiple capture mesh sequences.

3. Dynamic Surface Encoding

The following section introduces the use of generative networks to efficiently represent 4D mesh sequence from volumetric video content. Firstly, we discuss how to pre-process dynamic surfaces using state-of-the-art techniques to make it more suitable for neural networks. Secondly, we introduce two methods to compress 3D dynamic geometry using an autoencoder and a variational autoencoder. The advantage and limitations of both architectures is evaluated. This shows that the compact latent variable representation of the variational autoencoder is advantageous for mesh sequence variation. Finally, we conclude with a range of applications for the chosen approach.

3.1. Dynamic Surface Pre-processing

Dynamic surfaces represent sequences of 3D meshes of an actor or scene, preserving visual features, such as shape, motion and texture appearance of the surface. This content usually arises from performance capture studios which consist of multiple synchronised cameras focusing on a capture volume, with the ability to record a performance and convert it to a digital format described as volumetric video.

Performance capture systems tend to provide temporal inconsistent geometry for the captured performance, making it difficult to track the shape and appearance. Therefore, we use a skeleton-driven surface alignment framework [28] to provide 4D temporally consistent geometry and skeletal structures for animation control. In this work 4D reference to the known temporal correspondence of surface points across the mesh sequence [7].

This framework receives as input synchronised multiple view video from calibrated cameras and returns 3D skeletal pose and temporally consistent 4D mesh sequences. The purpose of the networks described in the following section is to learn a low dimensional latent representation of the dynamic 4D mesh shape. Therefore, it is important that our input data does not contain global translations. To achieve this we use a skeleton-based mesh alignment framework to obtain a 4D mesh sequence and align all meshes to a common origin. The input data used on the following section consists of centred temporally consistent 3D meshes.

3.2. Autoencoder Architecture

An autoencoder neural network is an unsupervised learning algorithm that learns an approximation to the input data. In this work the autoencoder consist of an encoder $P(X|z)$ and a decoder $Q(z|\tilde{X})$ so that $X \approx \tilde{X}$. The encoder maps the data X onto a latent vector z, and the decoder produces reconstructions $\tilde{X}$ of data X from the latent vector z. The architecture of the encoder consists of three fully connected layers, the first two layers use Leaky ReLu [24] activation layer and the output layer uses a linear activation layer. The encoder transforms the vertices X of mesh $M = \{X, C\}$ into a 128 dimensional latent vector z using fully connected layers. Where $X = [x_1, x_2, ..., x_N]$ is the $3N$ vector of mesh vertices and C is the constant mesh connectivity.

The decoder learns how to interpret the latent vector z to reconstruct an approximation $\tilde{X}$ of the input mesh X. The architecture of the decoder similarly to the encoder consist of three fully connected layers, although the output layer uses a tanh activation layer [33].

Training Details: We train our autoencoder for 10^4 epochs, which is optimised through validation data to avoid over-fitting with a learning rate of 0.001, where validation data is randomly sampled from the training data. We use stochastic gradient descent with a momentum of 0.9 to optimise the $L2 = ||\tilde{X} - X||^2$ loss between mesh vertices $\tilde{X}$ and the ground truth samples $X \in M$.

3.3. Variational Autoencoder Architecture

Variational autoencoders (VAE) have emerged as one of the most popular generative models, because they can be built on top of deep neural networks and provide a compact latent space. The VAE aims to maximise the probability distribution of the encoded data samples X, and allows the generation of novel data $\tilde{X}$ by sampling from the latent space. This is the advantage against using autoencoder neural network, which simply encodes the data samples X as a distinct latent vectors z in the latent space which may lead to discontinuities in the latent space. This does not allow the decoder to predict a plausible approximation of a latent vector z when this does not exist in the original encoded data.

Generative models can capture data dependencies by learning low-dimensional latent variables z to form a latent space $Z \in \mathbb{R}^d$, where d is the dimension. The model aims to maximise the probability of each X in training as follows,

$$p(X) = \int p(X \mid z)\, p(z) dz \tag{1}$$

Here, the latent variables z are sampled according to a probability density function $p(X)$ defined over $Z \in \mathbb{R}^d$, and the distribution $p(X|z)$ denotes the maximum likelihood estimation which allows to make dependencies of X on the latent vector z, and the $p(z)$ is the prior probability distribution of a latent vector z. In this work we chose the output of $p(X|z)$ to be a Guassian distribution

$$p(X \mid z) = \mathcal{N}(X \mid \mu(z),\ \sigma(z)^2 * I) \tag{2}$$

Where $\mu(z)$ is the mean function and $\sigma(z)$ is the co-variance function multiplied by the identity matrix I. The VAE model finds a probabilistic distribution of the trained data through an encoder and decoder architecture. The encoder is trained to map the posterior distribution of data samples X to the latent space $Z \in \mathbb{R}^d$ meanwhile forcing the latent variables z to comply with the prior distribution of $p(z)$. However, both the posterior distribution $p(z|X)$ and $p(X)$ are unknown. Therefore, VAE gives the solution that the

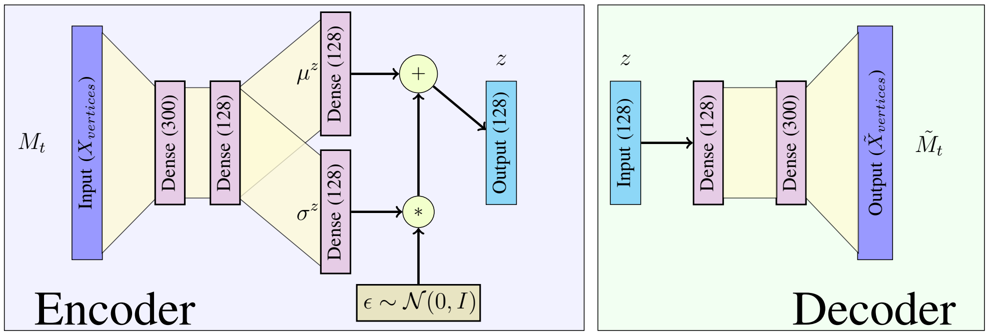

Figure 2. Variational autoencoder framework. The input and the output of the encoder and decoder respectively, are vertices. The first two dense layers of the encoder and decoder use Leaky ReLu [24].

posterior distribution is a variational distribution $Q(z|\tilde{X})$, computed by a neural network. In order to make $Q(z|\tilde{X})$ consistent with the distribution $p(z)$, the Kullback-Leiber (KL) divergence [20] is:

$$arg \min KL(Q(z \mid \tilde{X}) \mid\mid p(z)) \tag{3}$$

This network minimises the loss L of $p(X|z)$ over the whole latent space $Z \in \mathbb{R}^d$, where $\omega =$ batch size $\times$ number of vertices, as follows:

$$L = (X - Q(p(X \mid z) \mid \tilde{X})) + (\frac{KL(Q(z \mid x) \mid\mid p(z))}{\omega}) \tag{4}$$

The decoder is trained to map the latent variables z to generate data samples $\tilde{X}$ that reconstruct the optimal approximation of the true sample X from the latent vector z.

The VAE model pipeline used to train on dynamic surfaces is summarised in Figure 2.

Training Details: For the network to learn a good distribution we train our variational autoencoder for 10^4 epochs, which is optimised through validation data to avoid overfitting with a learning rate of 0.001. We set the prior probability over latent variables to be a Guassian distribution with zero mean and unit standard variation, $p(z) = N(z; 0, I)$. We use a stochastic gradient descent with a momentum of 0.9 to optimise the L2 loss between reconstructed vertices and the ground truth samples, and simultaneously the KL divergence of both probabilistic distributions.

Table 1. The table shows the lowest training loss on sequences using different latent dimensions.

	Latent Dimension			
Dataset	16	64	128	256
Dan [12]	0.442	0.413	**0.177**	0.212
Roxanne [30]	0.186	0.117	**0.087**	0.113
Thomas [6]	0.214	0.135	**0.115**	0.150

4. Motion Graphs

The following section demonstrates the necessary steps to generate motion graphs for compressed 3D dynamic geometry [3, 5, 10, 11], see Figure 4, firstly we introduce the pre-requisites for the framework to be initialised. Secondly we discuss the metrics chosen to evaluate similarity between the input data, and the cost to travel from frame to frame. Finally, a real-time motion synthesis approach to generate 4D meshes sequences with interactive animation control by concatenating and blending between the captured mesh sequences to generate novel mesh poses.

4.1. Input Data

The framework receives as input data, skeletal motion of each chosen sequence m_i, corresponding latent vectors $z_i \in S_n$, where S_n represents a sequence containing a collection of latent vectors z_i which corresponds to motion frames, and lastly we use the pre-trained decoder network $Q(z|\tilde{X})$ to interpret the latent vectors $z_i \in S_n$.

The decoder network produces $\tilde{X}$ for every $z_i \in S_n$ which represents a temporally aligned 4D mesh sequence, i.e. the topology and vertex connectivity is constant across

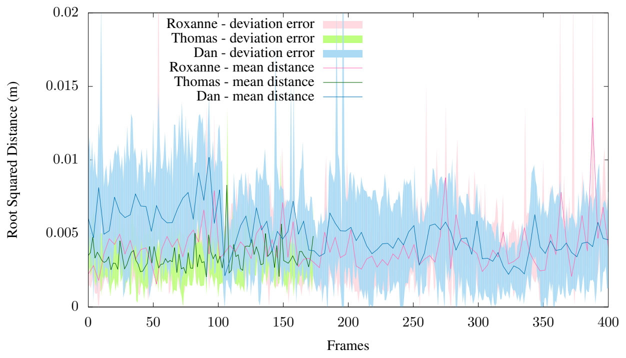

Figure 3. The graph shows the mean distance and the standard deviation from training and validation meshes against the decoder reconstruction using the lowest latent loss error, on Dan [12], Roxanne [30] and Thomas [6] datasets.

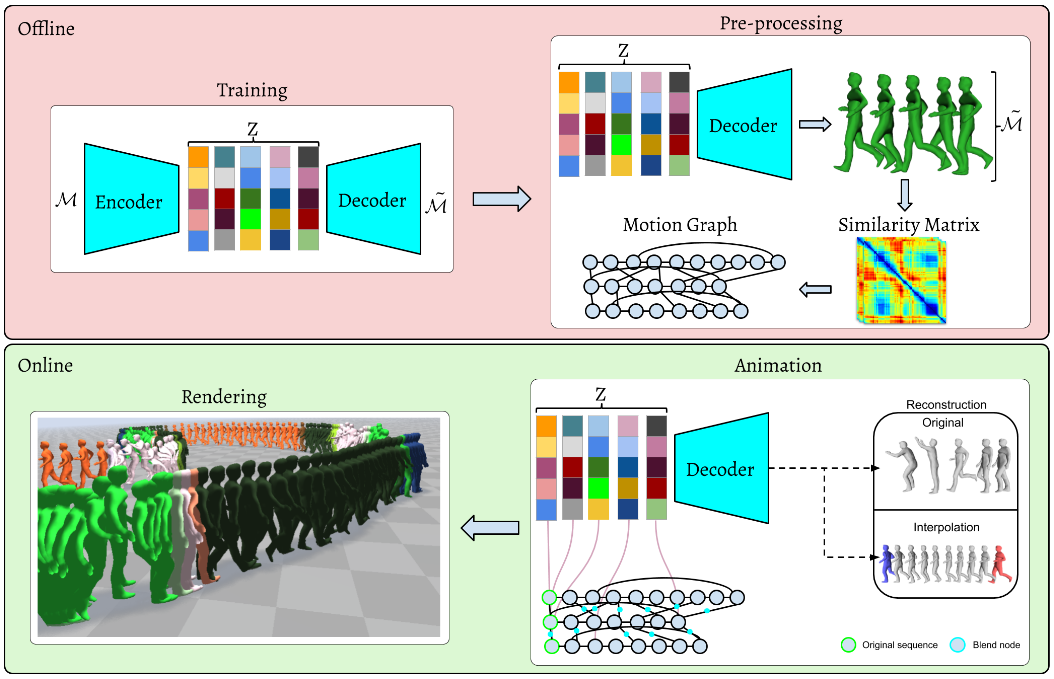

Figure 4. Motion Graph Pipeline, it consists of an offline and an online block. The offline block we train the network on 4D mesh sequences and initialise the motion graph using the trained decoder. The online block generates original and interpolated posed mesh allowing real-time rendering.

all sequences. In this work a sequence of 4D mesh sequence is a virtual representation of a human performing an arbitrary movement, not restricted in motion or clothing.

The construction of the motion graph is independent of the posed model used, allowing the framework to generalise its application to other models. Motion graphs are usually interpreted as a directed weighted graph structure built on 4D mesh sequences, where nodes represent frames containing shape and motion, and edges represent possible transitions between frames. In this work, edge weights account for both surface shape and structure and transition duration.

4.2. Pre-processing

The first step in the process is to automatically connect frames within the same sequences, and if possible create loops for cyclic motions such as the sequence can infinitely repeat, see Figure 4. Loops are generated via looking up using a similarity matrix $s(M_i^s(t_u), M_j^s(t_v))$ of each sequence, and automatically choose the minimum cost within a window of a pre-defined number of frames, see Sections 4.3 and 4.4. Transition within the same sequence should produce the most natural motion, hence the shape and motion cost should be very small. The next step is to fully connect the graph by adding all possible transition combinations between sequences to allow better path estimations to be found for all frames. This will generate a fully connected graph with appropriated edge weights using shape, motion and dynamic time warping metrics, as detailed in the following sections. Lastly, we optimise the graph by using Dijkstra's algorithm to minimise the number of transition in the final motion graph, as detailed in Section 4.5.

4.3. Shape Similarity Metric

The shape similarity matrix is computed for every pair of frames in the motion framework $s(M_i^s(t_u), M_j^t(t_v))$. For a given latent vector $z_i \in S_n$ the decoder $Q(z|\tilde{X})$ reconstructs temporally consistent geometry where vertex correspondences between the source and target meshes are known, therefore its possible to compute the shape and motion similarity between any pair of meshes. To measure similarity we use the Euclidean distances and velocities between mesh vertices: $s(M_i^s(t_u), M_j^t(t_v)) = \frac{1}{N_v}(||x_i^s - x_j^t|| + ||v_i^s - v_j^t||)$, where vertex velocity $v_i^s(t_u) = x_i^s(t_u) - x_i^s(t_u - 1)$, and N_v is the number of vertices. The matrix is then normalised as follows: $s(M_i^s(t_u), M_j^t(t_v)) = \frac{s(M_i^s(t_u), M_j^t(t_v)) - min(s(M_i^s(t_u), M_j^t(t_v)))}{max(s(M_i^s(t_u), M_j^t(t_v))) - min(s(M_i^s(t_u), M_j^t(t_v)))}$.

We compute the shape and motion similarity for every pair source $M_i^s(t_u)$ and target $M_j^t(t_v)$ mesh as $s(M_i^s(t_u), M_j^t(t_v))$ for all frames $t_u \in [0, T_i], t_v \in [0, T_j]$ giving a complete shape similarity matrix $S_{ij}^{st}(u, v)$ for all frames in the framework. The pre-computed similarities S_{ij}^{st} for all frames allows us to evaluate in real-time the cost in mesh deformation between any source and target meshes.

4.4. Transition Edge Cost

An edge in the motion graph represents a transition between two frames, frames will be describe as nodes in this section. For every edge we associate a weight to qualitatively represent the similarity of the shape of the transition between nodes. Realistic transitions should require little surface deformation, hence we use a metric that take into account the optimal surface interpolation cost between any pair of nodes [4]. The cost is the weighted sum of intermediate poses between node u and node v.

Firstly, in order to smoothly blend frame u from a 3D mesh sequence to frame v from another sequence, we need to consider a blend window length b. This window represents a successive number of frames b^u, on the source sequence it begins at frame u and ends at frame $u + b^u - 1$, in the destination sequence a window b^v ending at frame v and starting at frame $v - b^v + 1$, see Figure 4. Once, the window frame is initialised between source and target sequence, it is necessary to extrapolate the frames that gradually blend both sequences, generating smooth realistic transitions. To extract the optimal frames from source and target sequence we use standard dynamic time warping (DTW) [26, 36, 37] to estimate the best temporal warps w^u and w^v respectively with respect to the metric defined in Section 4.3. The transition duration varies within a third of a second and 2 seconds following the work of [36], hence we allow the length b^u and b^v to vary between boundaries b_{min} and b_{max}.

Secondly, we choose the transitions with the minimal total surface deformation cost $C_d(u, v)$ through the path generated from the DTW algorithm, as shown in Figure 4.

$$C_d(u, v) = \min_{b^u, b^v, w^u, w^v, C_l} \sum_{t \in [0, C_l]} s(w_t^u, w_t^v) \tag{5}$$

where $s(w_t^u, w_t^v)$ is the shape similarity cost defined in Section 4.3, and C_l is the length of the path found by the DTW algorithm considered as the transition duration, see Figure 4. The optimisation above finds the following optimal parameters $(b^u, b^v, w^u, w^v, C_l)$, which are considered later for motions synthesis. Similar to Section 4.3, we define the edge weight between nodes to be the surface deformation cost $C_d(u, v)$ and its interpolated duration cost $C_l(u, v)$. The following expression summarises the definition for the edge cost between poses u and v:

$$C_e(u, v) = C_d(u, v) + \alpha\, C_l(u, v) \tag{6}$$

in the case u and v are from the same sequence we can assume that the surface deformation should be minimal, therefore if $u < v$, we assume $\tilde{C}_e(u, v) = \min[\alpha(v - u),\ C_e(u, v)]$, $u < v$ [3]. To control the tolerance between surface deformation and transition duration we add weight α, the costs C_e and $\tilde{C}_e$ mean the edge weights.

This process will create a fully connected digraph where edges are weighted for the similarity of the shape of the transition between frames, in the next step we will discuss how to prune and optimise the connectivity of the complete digraph, see Figure 4.

4.5. Motion Graph Optimisation

The last step in the framework aims to find a global optimal solution to minimise the number of transitions between nodes. Plausible transitions are achieved by selecting the minimum cost transition on the similarity matrix between sequences. From Section 4.4, a fully connected digraph was built, connecting every pair of node sequences. Therefore selecting the minima transition of every node will possibly maintain a dense connectivity in the graph. Instead following the work of [3], we implemented a global optimal strategy that extracts and maintains only the best paths between all pair of nodes. Given the fully connected digraph, we use Dijkstra algorithm on every pair of nodes to extract the shortest path between source and target. Once this process is completed we remove all edges that do not belong to the new generated paths, giving us a connected digraph containing only the necessary transition with the least transition cost. This solution will guarantee the best realism when transitioning from frames of different sequences.

4.6. Motion Synthesis

This section discusses how we convert encoded dynamic surfaces into a continuous animation of 3D mesh surfaces. As we discussed previously at this stage we have an optimal connected digraph which contains nodes and edges, where nodes are 3D mesh frames and edges represent transitions. To create continuous animation, it is necessary in real-time to find the least costly transition from source node u to target node v. From default the least costly transition is a transition within the same sequence, therefore if the motion is not interactively changed by the user the framework will infinitely play the same sequence. In the case where the user requests to change the current motion the framework processes the minimum cost $C_d(u, v)$ of the current state u to the selected motion target v, and returns the following parameters $(b^u, b^v, w^u, w^v, C_l)$, as discussed in Section 4.4. This allows us to create smooth intermediate blend frames with a transition length of C_l. The following sections will discuss how we generate temporally consistent 3D meshes and intermediate interpolated meshes.

4.6.1 Mesh Synthesis

Every node in the graph has a latent space of the original 3D mesh that allows in real-time to extrapolate the original mesh. Therefore, the decoder described in Section 3.3 reconstructs the latent representation z_i for the current node.

4.6.2 Mesh Interpolation

Intermediate blend frames are only possible because of the characteristics of the VAE described in Section 3.3. As we described previously, the VAE generates a compact latent space where samples around some area have a high level of similarity, although because of its compact properties the VAE creates boundaries in the latent space where the reconstructed 3D mesh have high similarity error. This issue is avoided because the computed graph avoids transitions where the similarity error is extremely high. For interpolating the 3D mesh we perform a linear interpolation of two given latent vectors, and reconstruct its result using the pre-trained decoder $Q(z|\tilde{X})$, see Figure 7.

5. Results and Evaluation

This section presents results and evaluation for the proposed animated dynamic surface using generative networks, presented in Section 4.5. Dan [12] and Roxanne [30] datasets are reconstructed using multi-view stereo [30] and temporally aligned with HSDSR [28] which allows for surfaces pose manipulation. Thomas dataset [6] consists of four sequences of temporally aligned meshes. Examples of character animation using dynamic surfaces are shown in Figures, 1, 5 and 6. Results demonstrated the usability of neural networks to facilitate the representation and compression (see Table 2) of highly detailed meshes.

Error Metrics: Evaluation on mesh is performed using one-sided Hausdorff distance defined as $H_B(A) = \sup_{a \in A} d(a, B)$, where $d(a, B)$ is the distance from a point a to a set B, which has shown good measurements between two 3D meshes. The variational autoencoder uses the Equation 4 in Section 3.3 as metrics to predict plausible reconstructions. The comparison was performed between the original trained data, to ensure minimum error when sampling the original sequences, and validation data to guarantee a plausible result when generating unseen mesh, see Figure 7 and 9.

Latent Dimensions: We have evaluated our model with different values for the latent space (see Table 1), according to the reconstruction loss Equation 4. The results suggest that using 128 dimensions for the latent space gives a good trade-off between compression and shape detail.

Performance: Presented results were generated using a desktop PC with an Intel Core i7-6700K CPU, 64GB of RAM and a Nvidia Geforce GTX 1080 GPU. Pre-processing performance depends on the number of sequences, the decoder performance achieves ≈ 30 frames per second (fps).

Table 2. The table illustrates the number of vertices and frames, and disk space occupied in Megabytes (MB).

Dataset	vertices	frames	original	latent space	decoder
Dan [12]	3431	1447	636.4MB	**3.7MB**	**12.6MB**
Roxanne [30]	2475	414	129.7MB	**1.1MB**	**9.2MB**
Thomas [6]	5002	212	185.5MB	**0.46MB**	**18.3MB**

Compression: Table 2 demonstrates that the learnt representation is capable of compressing the dynamic mesh geometry by $\approx 93\%$. It is required to store the decoder and the latent variables for each frame in the motion dataset.

Limitations: The proposed network suffers from discontinuities in areas where there is insufficient training data, creating wrong transitions between meshes that largely differ in pose and shape, for example, two nearby samples from the latent space do not guarantee to be similar, see Figure 8. Although this limitation is overcome with motion graphs which do not allow for such distinct transitions. Currently the network is only able to represent one character a time, an interesting extension for future work would be to encode multiple characters in a single space, or a single person wearing multiple types of clothing.

6. Conclusions

The propose generative network is capable of both providing a compact representation of multiple motion capture sequences with an order of magnitude reduction in size. The variational autoencoder also supports interpolation in the latent space to synthesise novel intermediate motions allowing smooth transitions between captured sequences. Using the decoder part of the variational autoencoder allows applications to use less memory in run-time, therefore making it more suitable for technologies with memory constraints. We have illustrated that the proposed method is able to preserve shape details and motion from various public datasets. The animation framework is independent of the network architecture, allowing for future improvements in either of the frameworks. For instance, the neural network can be improved to generate posed meshes constrained on a skeletal pose, this will allow browsing through existing motion capture data and expand the range of motion captured on the original dynamic surface sequences. The animation framework can be extended to parameterised motion, allowing increased interactivity and motion control.

Acknowledgements

This research was supported by the EPSRC Audio-Visual Media Research Platform Grant (EP/P022529/1). The authors would also like to thank Adnane Boukhayma for providing the 'Thomas' dataset used for evaluation.

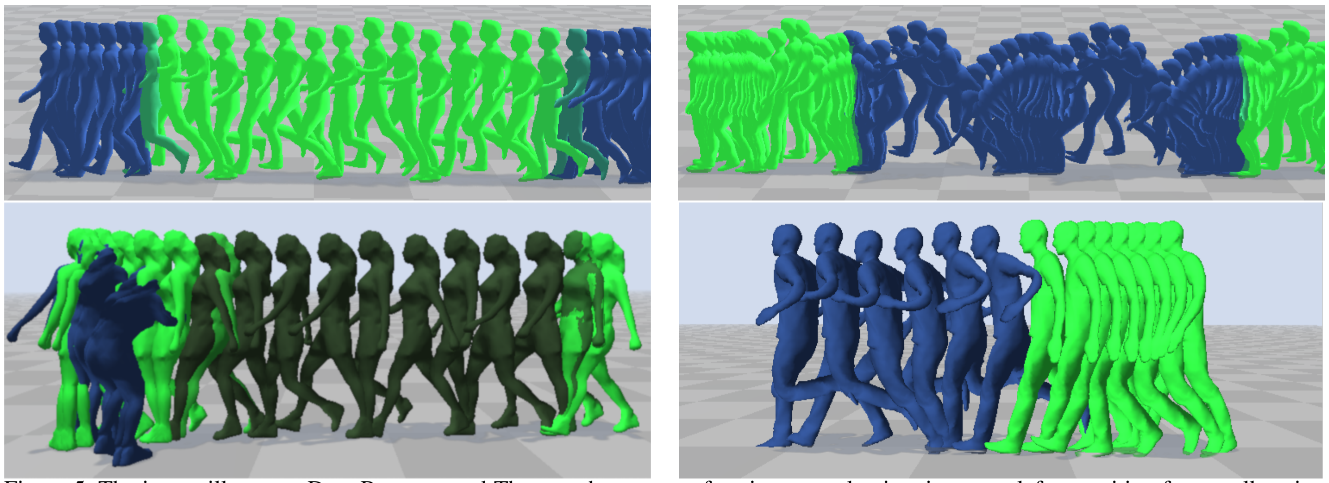

Figure 5. The image illustrates Dan, Roxanne and Thomas character performing several animations, top-left: transition from walk to jog sequence; top-right: from short to long jump sequence; bottom-left: from walk to stagger sequence; bottom-right: from walk to run sequence. The colours illustrates different sequences and transition blends.

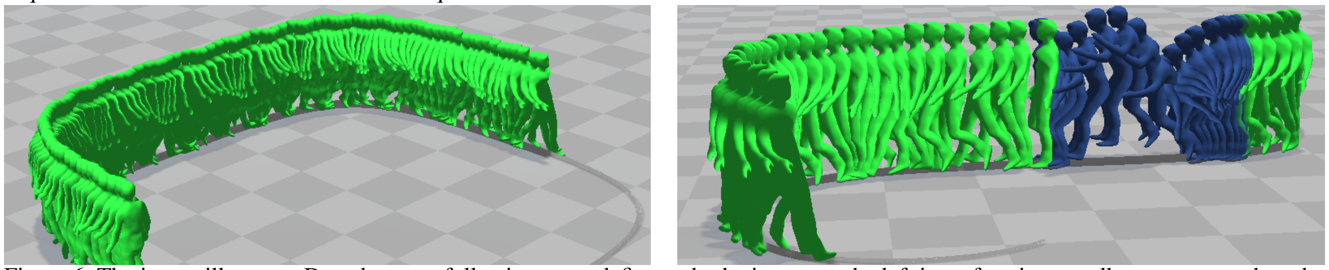

Figure 6. The image illustrates Dan character following a pre-define path, the image on the left is performing a walk sequence, and on the right a transition from walk to a long jump. The colours illustrates different sequences and transition blends.

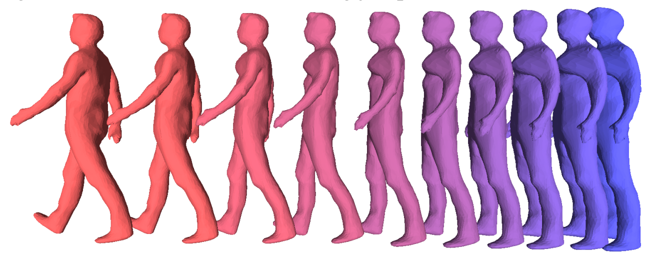

Figure 7. Interpolation in the latent space, the mesh coloured in blue represent the source and the mesh coloured in red is the target, in between we have the interpolated steps.

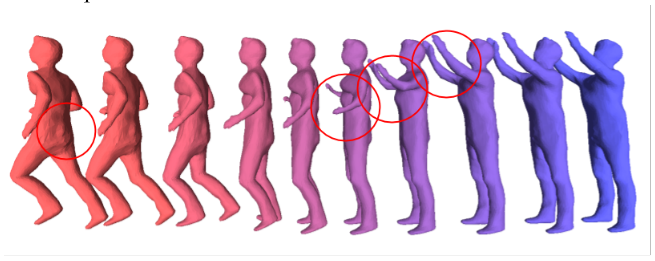

Figure 8. Latent space interpolation, the mesh coloured in blue is the source and in red the target, in between the interpolated steps. The circles represent artefacts caused from the interpolation.

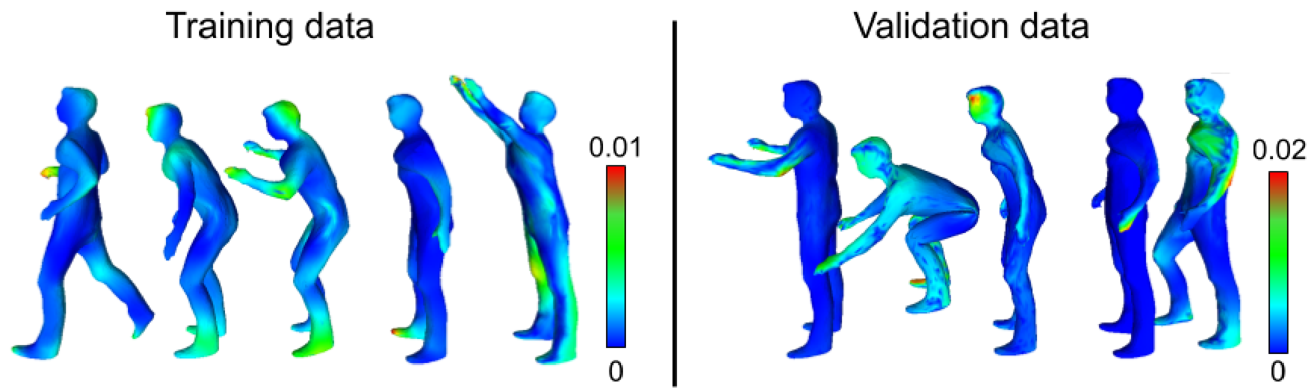

Figure 9. The left column shows the reconstruction error of the training (seen) data, the right column shows the reconstruction error for validation (unseen) data.

References

[1] O. Arikan, D. A. Forsyth, J. F. O'Brien, and J. F. O'Brien. Motion synthesis from annotations. *ACM Trans. Graph.*, 22(3):402–408, July 2003.

[2] I. Baran, D. Vlasic, E. Grinspun, and J. Popović. Semantic deformation transfer. *ACM Trans. Graph.*, 28(3):36:1–36:6, July 2009.

[3] A. Boukhayma and E. Boyer. Video based animation synthesis with the essential graph. In *2015 International Conference on 3D Vision*, 2015.

[4] A. Boukhayma and E. Boyer. Controllable variation synthesis for surface motion capture. In *2017 International Conference on 3D Vision (3DV)*, 2017.

[5] A. Boukhayma and E. Boyer. Surface motion capture animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 25(6):2270–2283, June 2019.

[6] A. Boukhayma, V. Tsiminaki, J.-S. Franco, and E. Boyer. Eigen Appearance Maps of Dynamic Shapes.

[7] C. Budd, P. Huang, M. Klaudiny, and A. Hilton. Global non-rigid alignment of surface sequences. *Int. J. Comput. Vision*, 102(1-3):256–270, Mar. 2013.

[8] C. Cagniart, E. Boyer, and S. Ilic. Free-form mesh tracking: A patch-based approach. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1339–1346, 2010.

[9] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Trans. Graph.*, 22(3):569–577, July 2003.

[10] D. Casas, M. Tejera, J. Guillemaut, and A. Hilton. Interactive animation of 4d performance capture. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):762–773, May 2013.

[11] D. Casas, M. Tejera, J.-Y. Guillemaut, and A. Hilton. 4d parametric motion graphs for interactive animation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, pages 103–110. ACM, 2012.

[12] D. Casas, M. Volino, J. Collomosse, and A. Hilton. 4d video textures for interactive character appearance. *Comput. Graph. Forum*, 33(2):371–380, May 2014.

[13] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan. High-quality streamable free-viewpoint video. *ACM Trans. Graph.*, 34(4):69:1–69:13, July 2015.

[14] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. *ACM Trans. Graph.*, 27(3):98:1–98:10, Aug. 2008.

[15] R. Heck and M. Gleicher. Parametric motion graphs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, pages 129–136. ACM, 2007.

[16] C. Huang, E. Boyer, and S. Ilic. Robust human body shape and pose tracking. In *2013 International Conference on 3D Vision - 3DV 2013*, 2013.

[17] P. Huang, A. Hilton, and J. Starck. Human motion synthesis from 3D video. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1478–1485. IEEE, jun 2009.

[18] P. Huang, A. Hilton, and J. Starck. Shape similarity for 3d video sequences of people. *Int. J. Comput. Vision*, 89(2-3):362–381, Sept. 2010.

[19] P. Huang, M. Tejera, J. Collomosse, and A. Hilton. Hybrid skeletal-surface motion graphs for character animation from 4d performance capture. *ACM Trans. Graph.*, 34(2):17:1–17:14, Mar. 2015.

[20] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. dec 2013.

[21] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Trans. Graph.*, 21(3):473–482, July 2002.

[22] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. *ACM Trans. Graph.*, 21(3):491–500, July 2002.

[23] S. Lombardi, J. Saragih, T. Simon, and Y. Sheikh. Deep appearance models for face rendering. *ACM Trans. Graph.*, 37(4):68:1–68:13, July 2018.

[24] A. L. Maas. Rectifier Nonlinearities Improve Neural Network Acoustic Models, 2013.

[25] C. Malleson, A. Gilbert, M. Trumble, J. Collomosse, A. Hilton, and M. Volino. Real-time full-body motion capture from video and imus. In *2017 International Conference on 3D Vision (3DV)*, 2017.

[26] M. Müller. *Information Retrieval for Music and Motion*. Springer-Verlag, 2007.

[27] F. Prada, M. Kazhdan, M. Chuang, A. Collet, and H. Hoppe. Motion graphs for unstructured textured meshes. *ACM Trans. Graph.*, 35(4):108:1–108:14, July 2016.

[28] J. Regateiro, M. Volino, and A. Hilton. Hybrid Skeleton Driven Surface Registration for Temporally Consistent Volumetric Video. In *2018 International Conference on 3D Vision (3DV)*, pages 514–522. IEEE, sep 2018.

[29] C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, Sep. 1998.

[30] J. Starck and A. Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, May 2007.

[31] J. Starck, G. Miller, and A. Hilton. Video-based character animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, pages 49–58, New York, NY, USA, 2005. ACM.

[32] C. Stoll, N. Hasler, J. Gall, H. Seidel, and C. Theobalt. Fast articulated motion tracking using a sums of gaussians body model. In *2011 International Conference on Computer Vision*, 2011.

[33] Q. Tan, L. Gao, Y. Lai, and S. Xia. Variational autoencoders for deforming 3d mesh models. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[34] L. M. Tanco and A. Hilton. Realistic synthesis of novel human movements from a database of motion capture examples. In *Proceedings Workshop on Human Motion*, pages 137–142, Dec 2000.

[35] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.*, 27(3):97:1–97:9, Aug. 2008.

[36] J. Wang and B. Bodenheimer. Synthesis and evaluation of linear motion transitions. *ACM Trans. Graph.*, 27(1):1:1–1:15, Mar. 2008.

[37] A. Witkin and Z. Popovic. Motion warping. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 105–108. ACM, 1995.

[38] W. Xu, A. Chatterjee, M. Zollhöfer, H. Rhodin, D. Mehta, H.-P. Seidel, and C. Theobalt. Monoperfcap: Human performance capture from monocular video. *ACM Trans. Graph.*, 37(2):27:1–27:15, May 2018.