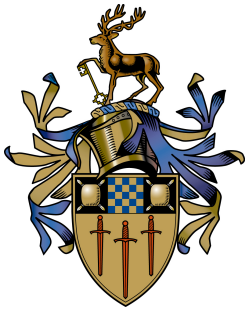
Interactive Video-Realistic Character Animation from 4D Performance Capture

Dan Casas Guix

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey



Centre for Vision, Speech and Signal Processing
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

November 2013

Summary

Recent advances in surface performance capture have demonstrated highly realistic reconstruction of human motion sequences acquired in a multi-camera studio. Free-viewpoint rendering techniques allow video-realistic replay of captured motions with interactive viewpoint control. However, current approaches do not provide methods for real-time motion and appearance manipulation, hindering the reuse of captured data. Non-sequential temporal alignment techniques enable the conversion of an unstructured set of 3D videos into temporally consistent mesh sequences with shared topology and vertex correspondence, known as 4D videos. The work presented in this thesis aims to develop methods to exploit 4D video datasets of human motion. In particular, a framework for real-time interactive control of a 4D video character created by the combination of multiple captured sequences is investigated. Our goal is to provide methods for video-realistic character animation and rendering with the flexibility and real-time interactive control associated with conventional skeleton driven animation.

An approach for parametric motion control of mesh sequences is introduced by blending multiple semantically related 4D video motions. This enables real-time interactive control using high-level parameters such as speed and direction for a walk motion. A hybrid non-linear mesh blending method is introduced. Our approach provides both realistic deformations and real-time performance, allowing parametric mesh spaces to be built at run time. A novel graph representation, referred to as 4D Parametric Motion Graph (4DPMG), is presented to encapsulate multiple independent parametric motion spaces and transition between them whilst maintaining natural non-rigid surface motions. 4DPMGs provide online path optimisation for transitions between parametric spaces of 4D video motions with low-latency, enabling responsive interactive character control with a large range of motions.

The final piece in the puzzle to enable video-realistic animations is provided by 4D Video Textures (4DVT), a new approach for free-viewpoint appearance manipulation that maintains the visual realism of the source video data. 4DVT enables video-realistic rendering of novel motions by combining multiple 4D video textures to synthesise plausible dynamic surface appearance. View-dependant optical flow is used for online alignment of parametric appearance from multiple views.

The research presented in this thesis demonstrated for the first time video-realistic interactive character animation from 4D actor performance capture. This represents a new approach to animated character production which achieves video-quality rendering and does not require highly skilled manual authoring.

Key words: Character Animation, 4D Video, 4D Performance Capture, Motion Graphs, Video-Realistic Animation, Free-Viewpoint Video

Email: d.casasguix@surrey.ac.uk — dan.casas@gmail.com

WWW: http://personal.ee.surrey.ac.uk/Personal/D.Casasguix/

Acknowledgements

Contents

Notation

Abbreviations

4DPMG	4D Parametric Motion Graphs
4DVT	4D Video Textures
fps	Frames per second
HD	High Definition
IK	Inverse Kinematics
MoCap	Skeletal Motion Capture
PCA	Principal Component Analysis
RMS	Root Mean Square
slerp	Spherical linear interpolation

Nomenclature

Chapter 3: Mesh Sequence Parametrisation

N	Number of meshes
N_x	Number of vertices
N_k	Number of triangles
x_i	i^{th} vertex
k_i	i^{th} triangle
$M_i = \{x_i, k_i\}$	Mesh
$\mathbf{M} = \{M_0, ..., M_N\}$	Vector of N meshes
$\mathbf{x} = \{x_0, ..., x_{N_x}\}$	Vector of N_x vertices
q_{ij}^k	Rotation of triangle k between meshes M_i and M_j

S_{ij}^k	Scale/shear transformation of triangle k between meshes M_i and M_j
T_{ij}^k	Affine transformation of k^{th} triangle between meshes M_i and M_j
$\mathbf{T} = \{k_0, ..., k_{N_k}\}$	Vector of N_k triangle transformations
$s_i(t) = x_i(t) - x_i(t-1)$	Vertex velocity
w_i	Blending weight
$\mathbf{w} = \{w_0, ..., w_N\}$	Vector of weights
$\mathbf{r} = \{r_0, ..., r_R\}$	Vector of precomputed weights used to generate D_{NL}
$M_L(\mathbf{w})$	Linear interpolated mesh
$M_{NL}(\mathbf{w})$	Non-linear interpolated mesh
$D_{NL}(\mathbf{r})$	Vertex displacement field
R	Number of precomputed D_{NL}
$b(\mathbf{M}, \mathbf{w})$	Mesh blending function
$g()$	Time warping function
$\mathbf{p}$	Vector of high-level motion parameters
$f(\mathbf{w})$	Function mapping between weights $\mathbf{w}$ and parameters $\mathbf{p}$
ϵ	Surface error correction threshold

Chapter 4: 4D Parametric Motion Graphs

P	Path inside trellis
E_S	Similarity cost
E_L	Latency cost
l_{max}	Trellis depth
Ω	Set of candidate paths
$d(M_i, M_j)$	Distance between meshes M_i and M_j
$s(M_i, M_j)$	Similarity between meshes M_i and M_j
$\mathbf{S}$	Similarity matrix

Chapter 5: 4D Video Textures

C	Number of cameras
v_i	i^{th} viewpoint
$I_c(t)$	Video sequence from camera c over time t
$I(t, v)$	Free-viewpoint video sequence for view v

$I(t, \mathbf{w}, v)$	Novel parametrised free-viewpoint video sequence for view v and motion parameters $\mathbf{w}$
$V(t) = \{I_c(t)_{c=1}^{C}\}$	Captured multiple-view video sequence
$M(t)$	Mesh at time t
$F(t) = \{V(t), M(t)\}$	4D video sequence
$h(F_1(t), ..., F_N(t), \mathbf{w}, v)$,	Function that combines N 4D video sequences
$a_{ij}(t, \mathbf{w}, v)$	Optical flow between images $I_i(t, \mathbf{w}, v)$ and $I_j(t, \mathbf{w}, v)$
$o(I_i(t, \mathbf{w}, v), I_j(t, \mathbf{w}, v))$	Function to find optical flow $a_{ij}(t, \mathbf{w}, v)$
R	Motion class
$E_A(P, v)$	Appearance cost
$z(a_{ij}, I_i, I_j, \mathbf{w})$	Function to warp and blend images I_i and I_j according to flow a_{ij} and motion parameters $\mathbf{w}$

Chapter 1

Introduction

The impressive technological advances occurring in today's world have enabled the investigation of systems for real-world scene capture and reconstruction. Media production industries, including television, film and video-gaming, have experienced a growing demand for producing photo-realistic 3D content. In particular, there is an increasing interest in 3D reconstruction of humans, which would allow the creation of *digital doubles*, enabling the seamless combination of video footage and computer generated humans. A number of applications could benefit from video-realistic digital characters: from interactive virtual avatars for video-games to actor replacement in dangerous film scenes.

Computer Vision and Computer Graphics research play a key role in the investigation of systems for 3D content creation. Three steps are required for highly realistic 3D human motion synthesis: accurate data capture; robust representation and performance reconstruction; and motion editing and control.

Initial approaches for skeletal human motion capture, referred to as MoCap, used optical or mechanical sensors to track a set of points on the surface which are related to body joints. Skeletal MoCap enabled the acquisition of real human motion with a level of detail difficult to replicate via manual animation [MG01]. Subsequent research focused on the reutilisation of captured MoCap data, allowing interactive control of a skeletal character. Real-time animation of 3D mesh characters has been achieved by the

combination of linear skinning methods and MoCap data. However, this approach suffers from two main limitations: first, synthesised animations do not reproduce non-rigid surface dynamics due to the simple rigging scheme; and second, the lack of appearance detail, requiring highly skilled artists and hours of manual work in order to incorporate texture information into the animated meshes.

In the last decade, approaches for 3D surface performance capture have been introduced, allowing the reconstruction of video-realistic 3D models that reproduce the motion captured in the source video. A synchronised multi-camera studio that simultaneously captures a scene from different viewpoints is generally used for performance capture [SMN*09]. Free-viewpoint rendering techniques enable video-realistic replay of the motion with interactive control of the rendered view. Model-based 3D reconstruction algorithms [CTMS03, dST*08, VBMP08, GSdA*09] derive a temporally consistent mesh structure that deforms in every frame in order to match the captured shape. However, their deformations are restricted by the model topology, hindering the reconstruction of complex surface dynamics. On the other hand, model-free methods [SH07b] use a combination of visual hull techniques, feature tracking and stereo matching, to generate a mesh sequence with a per-frame varying topology that reproduces the captured motion. However, a temporally coherent representation is required for integration into the standard animation pipeline.

Recent research on mesh sequence alignment [BHKH13] has enabled the representation of multiple 3D mesh sequences using a coherent mesh structure that deforms over time to match the captured shape, referred to as 4D video. Aligned sequences preserve the captured surface details and allow complex surface deformation. This has opened up the applications of 3D mesh sequences reconstructed from multi-camera performance capture, enabling the integration of video-realistic captured motions into the traditional character animation pipeline.

This new context for character animation has motivated the work presented in this thesis, which explores new methods for human motion synthesis exploiting datasets of captured 4D video sequences. In order to create believable video-realistic interactive characters from 4D video performance capture this research introduce a number of

techniques for real-time control, representation and rendering.

Previous research on skeletal MoCap data successfully synthesised novel motions by exploiting datasets of captured sequences [RCB98, HG07], however, these methods are not valid for 4D video data. While skeletal data consists of a reduced set of joint angles per frame which can be easily manipulated, 4D video data consists of thousands of 3D vertex positions that describe the surface shape at each frame. Therefore, novel approaches for both surface and appearance manipulation are required. Captured surface geometry as well as appearance detail need to be preserved in the synthesised motions in order to maintain the captured visual realism.

Our goal is to introduce methods to enable interactive video-realistic character animation by the combination of multiple captured 4D video sequences. In order to achieve this, a number of open problems related to 4D video are addressed in this thesis: development of methods for mesh sequence blending; study of shape and motion similarity for mesh sequence concatenation; development of methods for video-realistic dynamic appearance control for parametrised 4D models; and efficient implementation of mesh and appearance manipulation algorithms for interactive motion synthesis.

1.1 Thesis outline

The content of this thesis is structured as follows:

- **Chapter 2 - 4D Performance Capture.** This chapter presents an overview of the existing literature related to human motion capture. Starting from the early approaches consisting of manual drawings based on observations, to state-of-the-art methods for 3D performance capture in a multi-camera studio. Finally, the state-of-the-art framework for 4D video capture used in this thesis is described together with details of the reconstructed datasets used throughout this thesis.

- **Chapter 3 - Mesh Sequence Parametrisation.** This chapter investigates methods for high-level interactive animation of 4D video sequences by blending multiple captured clips. Previous research on both skeletal motion capture

(MoCap) and 3D mesh interpolation is initially reviewed. Three problems are addressed: time-warping to temporally align key events across sequences; high-level parametric control to provide the user with intuitive parameters of the motion to synthesis novel mesh sequences according to user controlled motion parameters; and finally, real-time mesh sequence blending. The latter is investigated in detail, evaluating linear and non-linear solutions. A novel hybrid non-linear approach is proposed, achieving both real-time performance and realistic deformations. Mesh sequence parametrisation is evaluated for multiple characters performing a variety of motions. The proposed approach enables parametric motion spaces to be constructed from a set of mesh sequences of example motions, allowing the synthesis of parametric 4D video motions.

- **Chapter 4 - 4D Parametric Motion Graphs.** This chapter introduces a novel representation technique for 4D video sequence concatenation, with the goal of seamlessly linking a set of parametric motions built using the approach proposed in Chapter 3. Related work on MoCap and 3D mesh sequence concatenation is reviewed. A novel data structure, referred to as a 4D Paramatric Motion Graph (4DPMG), is introduced to encapsulate sets of 4D video parametric motions and transition between them. Transitions between different parametric motion spaces are evaluated in real-time based on surface shape and motion similarity. The 4DPMG representation allows real-time interactive character animation while preserving the natural dynamics of the captured performance.

- **Chapter 5 - 4D Video Textures.** Previous approaches for free-viewpoint video rendering are limited to replay of the captured textures [ZKU*04, SH07b]. Therefore they cannot synthesise novel appearance to match the parametric poses generated using the 4DPMG introduced in Chapter 4. This chapter presents a novel approach to control the dynamic appearance of a parametrically controlled 4D video character to produce plausible video textures according to the motion, referred to as 4D Video Textures (4DVT). 4DVT enable real-time video-based character animation with interactive control of the motion and viewpoint whilst maintaining the video-realism. A user study is performed which confirms that

4DVT visual-quality is comparable to captured video.

- **Chapter 6 - Conclusions and Future Work.** This chapter presents an overall discussion of the results achieved in this thesis. Main conclusions and challenges that arise from our work are discussed. Research directions are proposed to further extend 4D video animation.

- **Supplementary Video.** A video of the results presented in this thesis can be downloaded from the following link:
  http://thesis:th3s1s@personal.ee.surrey.ac.uk/Personal/D.Casasguix/thesis

1.2 Contributions

The main contributions of the work presented in this thesis are:

- Parametrisation of 4D models for interactive control. High-level motion parametrisation is introduced by real-time blending of multiple captured mesh sequences. This leads to parametric control of mesh sequences analogous to approaches for skeletal motion parametrisation [RCB98].

- Hybrid non-linear 3D mesh blending. A novel approach is introduced which combines the realistic deformations achieved computationally expensive with non-linear 3D-blending schemes [Sor06, KG08, TH11] with the real-time performance of linear blending. This allows real-time interactive parametric controlled animation from multiple mesh sequences with the realistic deformation of non-linear methods.

- Parametric motion graph representation of a database of 4D videos for interactive animation. A database of mesh sequences is represented in a graph-structure, referred to as 4D Parametric Motion Graphs (4DPMG) with nodes representing parametrised motions, and edges transitions between motions which preserve surface shape and non-rigid motion. This representation allows real-time interactive control analogous to skeletal move-trees or parametric motion graphs [HG07].

- Online path optimisation for transition between parametric motion spaces with low-latency. This allows responsive interactive character control without delays in transitioning between motions.

- Real-time combination of multiple 4D videos to render novel motions and viewpoints with video-realistic quality with dynamic surface appearance. The proposed approach, referred to as 4D Video Textures (4DVT), enables the parametrisation of a set of 4D video examples of related motions, allowing to synthesise free-viewpoint video for novel motions with video-realistic quality. As the motion of the character changes, so does the dynamic appearance of the rendered video.

- View-dependant alignment of appearance between multiple 4D video sequences using optical flow estimation to allow interpolation of appearance whilst maintaining the dynamic detail.

1.3 List of publications

- D. Casas, M. Volino, J. Collomosse, A. Hilton. 4D Video Textures for Interactive Character Appearance. 2014. (Under submission).

- M. Tejera, D. Casas and A. Hilton. Animation Control of Surface Motion Capture. *IEEE Transactions on Cybernetics*, (In press), 2013.

- D. Casas, M. Tejera, J-Y. Guillemaut and A. Hilton. Interactive Animation of 4D performance capture. *IEEE Transactions on Visualization and Computer Graphics*, 19 (5), 2013.

- D. Casas, M. Tejera, J-Y. Guillemaut and A. Hilton. 4D Parametric Motion Graphs for Interactive Animation. *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, 2012, Costa Mesa (CA), USA.

- D. Casas, M. Tejera, J-Y. Guillemaut and A. Hilton. Parametric Animation of Performance Captured Mesh Sequences. *Computer Animation and Virtual Worlds*, 23 (2), 2012.

- D. Casas, M. Tejera J-Y. Guillemaut and A. Hilton. Interactive 3D Video Animation. *European Conference on Visual Media Production (CVMP), Short paper*, 2012, London, UK.

- D. Casas, M. Tejera, J-Y. Guillemaut and A. Hilton. Parametric Control of Captured Mesh Sequences for Real-time Animation. *Conference on Motion In Games (MIG)*, 2011, Edinburgh, UK.

- D. Casas, J-Y. Guillemaut, A. Hilton. Interactive Parametric Control of Mesh Sequences. *Eurographics / ACM SIGGRAPH Symposium on Computer Animation, Short paper*, 2010, Madrid, Spain.

Chapter 2

4D Performance Capture

Highly skilled artists have been drawing human-like motion for 2D cartoon characters for decades. While this approach has proved to be successful to animate characters in comics and 2D animation, it is not only very tedious but also does not achieve results as realistic as real human behaviour. Therefore, due to the high complexity of the motions that humans can perform, computer character animation has traditionally been based on captured data, allowing the reproduction of actual human motion.

This chapter reviews the literature around motion performance capture in the Computer Vision and Graphics research community. Initial approaches, based on rotoscoping to copy motion fron image sequences, quickly evolved into automatic marker-based skeletal capture when the first computer animation system were investigated, back in the 1980s. Since then, researchers have focused on both improving the quality and relaxing the requirements of the capturing schemes, leading to sophisticated markerless systems that allow detailed capture and reconstruction of fast motions, including surface and appearance information, using temporally coherent 3D mesh representation, known as a 4D video.

We then describe both the studio setup and state-of-the-art techniques used in this work to capture human performance in a synchronised multi-camera environment and generate a 4D dataset of motions. Data captured through this approach is used in the following chapters of this thesis as a input data to synthesise novel 4D character animation.

2.1 Literature review

2.1.1 Motion Capture

Artists have been interested in animating characters ever since the first form of animated cartoon was created in the 1850s. However, they soon realised that, due to the high complexity of the human body, drawing believable human motions was not an easy task.

In the late 1800s Muybridge and Marey pioneered in the use of photography to study human and animal locomotion. The later introduced the *chronophotographic gun*, and instrument capable of capturing 12 consecutive frame a second and recording them in the same frame, allowing detailed study of the motion of a variety of animals. Inspired by this work, a first attempt to mitigate the difficulties of drawing human-like motion consisted in creating animations observing real world motions for inspiration. Soon, in the 1910s, the idea of drawing motion from observation led to the first mechanism created to help artist draw realistic motions: the Rotoscope, a device that projects recorded film images onto a frosted glass, providing the animator with indications from real footage for their drawings.

A few decades later, in the 1950s, the first forms of computer animation appeared, enabling the generation of animated images by using computer graphics, however, motions were still handcrafted by animators. In order to provide realistic and less time-consuming motions, methods for automating the process of creating motion from observations were required.

Motion Capture, *MoCap*, techniques, initially introduced in the 1980s for biomechanics, and later in the 1990s adopted in film production, use optical, mechanical or magnetic sensors to capture the movements of the human body that can then be transferred to animated characters [Gle99, Men99, HFP*00]. *MoCap* overcomes traditional handcrafted animation pipeline by automatically inferring joint position/angles from the observation of markers located on surface of the character, providing a simplified human skeletal motion to the animator. This results in an animation pipeline that is less laborious and more realistic than any previous approach for human motion generation.

Extensive surveys published by Moeslund *et al.* stated that, after studying over 500 publications, the general structure of a *MoCap* system comprises the following steps: initialisation, tracking, pose estimation, and recognition [MG01, MHK06]. The *initialisation stage* ensures that the system has a correct interpretation of the current scene, including camera calibration, usually done offline, and model initialisation, which can be done manually or automatically, requiring a fixed set of sequence movements to identify joints of the skeleton and estimate limb lengths. The *tracking stage* aims to establish relations of the subject limbs between consecutive frames. This is achieve by initially segmenting foreground and background, then converting the segmented image into a simpler representation and finally defining how the subject is tracked from frame to frame. The next stage is *pose estimation*, which identifies how the human body limbs are configured in a given scene. Both *model-free*, with no prior model to fit into the pose, and *template-model* methods have been investigated in the literature. Finally, the *recognition state*, is usually part of the post-processing part and aims to identify and classify the captured motion.

2.1.2 Towards Markerless Motion Capture

Traditional marker-based *MoCap* captures a number of joint positions of the human body, enabling the replay of the skeletal captured motion. However, while these systems are precise, they are invasive and expensive. To overcome these limitations, markerless *MoCap* systems have also been investigated in the literature in the last decades [MG01, MHK06, Pop07]. Many applications, such as video surveillance and Human-Computer Interaction, would potentially benefit from markerless *MoCap* systems, which would allow unconstrained motion tracking from video footage.

One of the first approaches was published in the early 1980s by Hogg [Hog83], who used a model-based analysis-by-synthesis methodology to extract human motion from multiple views . A bounding box of the human shape was found by image subtraction, and the edges of the box were compared with the edges of a projected human model. Similarity, Rohr [Roh93] used the same approach improved by a Kalman filter and a motion model to obtain more robust results. Researches soon realised that the

problem with using articulated models is the high dimensionality of the configuration space, which exponentially increases computational costs. In fact, initial research was restricted to walking motions [Hog83, Roh93] to reduce the dimensionality. Gavrila and Davis [GD96] tried to solve the same problem but without assuming a known motion model, instead, they utilised four camera views to capture motions of characters wearing tight colour-specific clothes to obtain good results. Once a specific part of the body was identified, a hierarchical search was employed to constraint the rest of the model . Bregler and Malik [BM98] assumed fixed viewpoint of the character to simplify the problem and represent skeletal joint angles using products of exponential maps and twist motions that only required a simple linear system to be solved. They were able to fit a kinematic chain into the captured character and update it over time. Less restricted approaches, using stochastic tracking techniques [DBR00, MH03], have also been investigated. Deutshcer *et al.* introduced the *annealed particle filter* for searching high dimensional configuration spaces that did not rely on any of the assumptions for dimension simplification mentioned before.

At the same time, in the 1990s, methods for rendering volumetric representations of a scene from multiple camera views were also proposed [Lau94, KR97, SD99]. Inspired by earlier research for constructing discrete grids of voxels from a set of silhouettes, known as voxel carving [Pot87], Laurentini introduced a technique for shape reconstruction from silhouettes referred to as the *Visual Hull* –also know as *shape-from-silhouette*, *SFS*– that results in an upper bound volume representation of the real object's shape [Lau94]. Seitz and Dyer [SD99] extended the volume intersection problem, introducing a *voxel colouring* framework that enable the identification of colour-invariant points in a set of basis images referred to as the *photo-hull*. This allows a better volume reconstruction of the scene. More recently, approaches for real-time image-based visual hull reconstruction have been proposed [MBR*00], exploiting graphic-specific hardware and investigating not only 3D reconstruction of a captured volume but also slices of the visual hull, allowing the construction of interactive virtual environments [Lok01].

Taking advantage of these volume-from-silhouettes methods investigated in the 1990s [Lau94, SD99, MBR*00], further research in multi-camera markerless motion capture systems improved previous results, using reconstructed volume models as a input data

for motion estimation. Cheung *et al.* [CKBH00] used a 5-camera system that computed a real-time volumetric voxel-based reconstruction and fitted a 3D ellipsoid model into it. Unlike previous approaches, the tracking and fitting of the model was done in the 3D domain instead of projecting the predicted model into 2D . Theobalt *et al.* [TMSS02] introduced an approach for markerless full-body motion capture, combining colour-based optical tracking with the visual hull reconstruction from multiple camera views [Lau94]. Their system allowed real-time fitting of a skeleton to the video footage. Carranza *et al.* [CTMS03] proposed a model-based approach in which a humanoid 3D mesh was deformed to fit the captured character by maximising the overlap between projected model silhouettes and input camera silhouettes. De Aguiar *et al.* [dATM*04] use visual hulls built from a shape-from-silhouette method to identify the kinematic chain of motion. This approach, which identifies rigid body parts by subdividing the input volume to fit ellipsoidal shells in every time frame, does not require any previous knowledge of the skeletal model. Cheung *et al.* [CBK05] improved previous results on markerless motion capture by initially building a personalised kinematic skeleton for each character. Further approaches ([SH07b, dST*08, VBMP08, GSdA*09]) that aim to not only capture but also realistic rendering of the 3D character, are discussed in greater detail in the next section.

Recently, methods for markerless motion capture have moved away from synchronised camera setups, allowing the recovery of 3D human motion data from unsynchronised and uncontrolled capture environments [HRT*09, ESH*12] However, despite the efforts in developing motion capture systems and volumetric reconstruction from silhouettes, the reconstructed shape resulted only in a coarse approximation of the actual body, which didn't allow realistic re-rendering of the captured characters.

2.1.3 Free-Viewpoint Rendering

In the last three decades, Computer Vision and Computer Graphics researches have been interested not only in capturing both static scenes and dynamic motions –see Section 2.1.2– but also in reproducing captured video appearance from novel viewpoints. In order to provide full flexibility to the final replay, there has been a strong interest in

investigating methods for novel viewpoint rendering, enabling the generation of photo-realistic 3D digital video sequences that maintain the captured image quality. Methods to achieve this goal can be classified into two main groups: image-based methods, which only rely on the 2D image domain to generate 3D renders; and model-based methods, which use a 3D geometric proxy of the scene to compute the final render.

Chen and Williams [CW93] pioneered in image-based synthesis of novel viewpoints for 3D scenes from multiple captured images, introducing a real-time approach for image interpolation. Pixel-by-pixel correspondence between adjacent input images is automatically precomputed, based on the camera's position and orientation. An image morphing method was used to interpolate each image towards the other, blending pixel values. Neighbouring pixel similarity is exploited to group blocks of pixels that move in a similar manner, allowing efficient computation. To improve on previous approaches for 3D reconstruction from multiple images, Debevec *et al.* presented a photograph-based approach for modelling and rendering static architectural scenes, combining geometry-based and image-based methods [DTM96]. With minimal user interaction, consisting in labelling edge-correspondences across images, the computer determined the parameters of a hierarchical model of parametric polyhedral primitives to reconstruct the architectural scene. A novel view-dependent texture mapping approach was introduced to render the reconstructed 3D geometry. Pixel colours are assigned combining the original images, weighted according to the angular distance between the rendered virtual view and the original viewpoint. Texture *holes* present in the final render due to areas that are occluded in all input images were filled using texture interpolation [CW93].

Further research extended model-based methods for 3D reconstruction to dynamic scenes [MTG97, KR97]. Moezzi *et al.* presented an approach based on a shape-from-silhouette voxel-carving method using a 17 cameras setup that successfully reconstructed fast human motions such as a karate kick. Real-time view-dependant texture mapping was achieved by precomputing and assigning to each face its original captured color in each camera and retrieving it at run time. Similarly, Kanade *et al.* presented the *Virtualized Reality*^{TM}, a system with a $5m^2$ area dome and 51 cameras for free-viewpoint video replay of human performance [KR97]. Depth information was

derived from input images and converted to textured 3D mesh models. Although it improved previous research, their system still suffered from artefacts such as *ghosting* effect caused by differences in brightness between cameras, texture holes, motion parallax and low resolution details.

At the same time, in the late 1990s and early 2000s, model-based methods for 3D reconstruction and free-viewpoint rendering of dynamic scenes were investigated. A number of novel solutions for improved viewpoint interpolation were proposed. Buehler *et al.* [BBM*01] described an image-based approach that generalised previously proposed algorithms, including view-dependent texture mapping [DTM96] and light field/lumigraph approaches [GGSC96]. Buehler's approach, referred to as *Unstructured Lumigraph Rendering*, requires a set of source images, their camera locations and a geometry proxy of the scene. A *Camera Blending Field*, defined by the field-of-view and angular differences between the desired ray and the source images, was used to synthesise pixel values using a weighted sum of the input pixel data.

Zitnick *et al.* [ZKU*04] improved previous results by automatically inferring per-pixel image depth maps from synchronised multi-camera capture. A novel offline-computed two-layer depth map representation was introduced to overcome artefacts caused by unclear foreground / background edges, enabling high-resolution online free-viewpoint rendering. Vedula *et al.* [VBK05] proposed another image-based method for dynamic scene modelling from synchronised multi-camera capture based on the explicit recovery of scene properties using the so-called *scene flow* [VBR*05], a first order measure (velocity) of the instantaneous non-rigid motion of all objects in the scene. Reconstructed models overcome limitations of previous per-frame shape reconstruction since they allow computation of geometric information. For any requested position and time of the novel view, *scene flow* is used to interpolate the shape, which is textured by projecting it into the original images, taking advantage of the known geometrical correspondence. Recently, more general solutions for video-based rendering that allow seamless non-photorealistic interactive camera selection control from unsynchronised hand held cameras have been proposed [BBPP10].

In conclusion, although image-based and video-based methods generally achieve plau-

sible video-realistic results, they do not provide temporally coherent mesh geometry or 360 degree shape models, both properties required for the 4D-video manipulation techniques investigated in this thesis. Furthermore, image-based methods usually fail due to the inherent visual ambiguity in geometrically complex scene such as complex human motions.

Model-based approaches for free-viewpoint video replay of human performance have been proposed in order overcome these limitations. Carranza *et al.* proposed a framework [CTMS03] in which markerless *MoCap* is used to rig a generic humanoid mesh model and a free-viewpoint texturing approach was used to provide the final animated mesh with photorealistic appearance. Body pose parameters were found by maximising the overlap between projected model silhouettes and input camera silhouettes in every frame. This approach successfully captured human shape and reproduced temporally consistent mesh sequences with free-viewpoint appearance of human motion. However, no surface non-rigid motions –such as cloth movement, wrinkles or hair– are present in the final geometry proxy.

This limitation has been addressed by model-based approaches for 3D human capture from multi-camera studio capable of deforming the source 3D mesh model to enable the reproduction of both rigid and non-rigid motions [dST*08, VBMP08]. De Aguiar *et al.* [dST*08] used a static laser scan to obtained a high resolution 3D mesh template of the character to be captured. SIFT image features were used as a deformation constraints in a Laplacian deformation framework to deform a coarser tetrahedral mesh used as a first pose estimation to recover the global position. The high resolution mesh model is then mapped into the deformed tetrahedra, providing a deformed mesh with non-rigid surface details. Similarly, Vlasic *et al.* [VBMP08] also use a high resolution scanned mesh as a source model. For every captured frame, skeletal pose was obtained from the reconstructed visual hull and used to rig the template using a conventional linear blend skinning approach. A Laplacian deformation framework was used to deform the rigged template to fit the source silhouettes. This results in a temporally aligned mesh sequence that reproduces both rigid and non-rigid captured motions. More recently, Gall *et al.* proposed a similar approach [GSdA*09] that improved on the results obtained by Vlasic *et al.*, introducing a method for better skeletal fitting based on both

appearance information and the surface model rather than visual hull, which is more sensitive to silhouette errors.

On the other hand, Starck and Hilton [SH07b] introduce an approach for 3D mesh reconstruction from multi-camera capture that does not require any geometry prior. Their approach, which combines stereo and shape-from-silhouette reconstruction, enabled the the acquisition of shape, appearance and motion of fast human performances such as break-dancing moves, using 8 synchronised conventional HD cameras. Visual-hulls is initially derived from automatically extracted silhouettes. Wide-baseline feature matching between cameras allows refinement of the true surface inside the visual-hull. Finally, the volume is extracted by computing the surface within the visual hull that passes through the detected features, matches silhouettes images and maximises consistency in appearance across all views. Reconstructed 3D models, although less detailed than other approaches that used high resolution laser scans model [dST*08, VBMP08, GSdA*09], reproduce non-rigid surface dynamic details present in the original footage. Extracted surfaces provided a time-varying geometry along the sequence, consisting in a triangulated surface mesh that changed its geometry, topology and mesh connectivity in every frame.

2.1.4 Mesh Sequence Temporal Alignment for 4D Video Data

A critical step for editing and reuse reconstructed 3D data from multi-camera capture, known as 3D video data, is the temporal alignment of captured mesh sequences to obtain a consistent mesh structure with surface correspondence over time. Such temporally coherent mesh representation, referred to as 4D video data, would enable the reutilisation of the captured data using a conventional computer graphics pipeline. This assumes that deformable surfaces are represented with a consistent mesh structure with only the surface shape varying over time.

Due to the utilisation of a source geometric template that is iteratively deformed to match the captured silhouettes, model-based approaches for 3D reconstruction of human performance capture inherently provide a 4D mesh representation of the motion [CTMS03, dST*08, VBMP08, GSdA*09]. However, the use of a geometric template

also limits the range of surface deformation which can be represented. Model-free approaches [SH07a] output a 3D mesh per-frame with varying topology that needs to be post-processed to be converted into a 4D model representation. This allows the representation of more general shape deformation, but need to be post-processed. Therefore, in order to deform temporally unstructured mesh sequences into coherent 4D models, methods for mesh tracking and deforming are required.

An initial approach [SH05, SH07b] for converting a geometrically inconsistent mesh sequence into a temporally coherent representation of mesh sequences employed a bijective mapping for surfaces of genus-0 spherical topology to track the non-rigid deformations in a 2D domain, enforcing a one-to-one correspondence. However, this requires temporally consistent cuts to be applied to the arbitrary genus reconstructed surface to enforce a mapping to spherical topology for all frames.

Consequently, more general surface tracking methods have been investigated in order to provide 3D correspondences along free-form mesh sequences with changing topology. Vedula *et al.* introduce the 3D *scene-flow* [VBR*05], a dense three-dimensional vector field defined for every point in the scene that describes how each voxel moves across time. However, stable long term tracking is needed for wide-timeframe matches, and surface matching across different sequences was not guaranteed. Starck and Hilton [SH07a] present a framework to match arbitrary frames from captured sequences containing not only large-scale articulated motions but also large non-rigid surface deformations. Their approach, which introduce a set of feature descriptors invariant to isometric deformations, does not require any prior model and allowed wide-timeframe surface matching. Ahmed *et al.* [ATR*08] propose a robust system for surface correspondence across sequences. Their approach is intended to solve local and not global correspondences, hence matches are easier and faster to solve. Their approach looks initially for SIFT [Low04] features between consecutive frames, then harmonic functions are used to produce dense correspondences by propagating the sparse SIFT correspondence information. In the ideal case where the dense correspondence field was found through the whole mesh, trivial vertex displacement, supported by a basic Laplacian deformation scheme for smoothness, is used to deform one mesh into the consecutive one, generating a 4D representation of the sequence. Ahmed *et al.* assume isometric

deformations in consecutive frames of the captured data, and this is not always true, especially in the case of large cloth and hair movements.

More recently, Tung and Matsuyama overcame this and other limitations, proposing a method for surface matching that did not relay on appearance information. This approach allows matching between different actors [TM10]. Their approach is based on sparse geometric features and geodesic mapping, which ensures a one-to-one matching across surfaces, and allows matching under large non-rigid deformations.

Further research in surface tracking and mesh alignment resulted in methods that do not require any prior knowledge of the surface data. Instead, they focus on deforming a single frame through subsequent frames of the sequence. Cagniart *et al.* introduce an approach for full body tracking and alignment based on geometry matches [CBI09]. Correspondences between consecutive frames are found using an Iterative Closest Point, ICP, algorithm on randomly generated geodesic surface patches. A Laplacian [Sor06] mesh deformation framework is used to deform consecutive meshes using the nearest point correspondences as a constraints. More recent research on the patch based paradigm [CBI10a] found correspondences using overlapping patches to provide multiple targets for vertices. Accurate tracking was achieved by enforcing rigidity constraints between neighbouring surface patches. This approach was not limited to reconstructed geometry from multiple camera systems, it was also demonstrated to allow reconstruction of deforming surfaces from monocular camera sequences. Further advances in surface tracking of arbitrary shapes focused on the fact that 3D mesh models can sometimes carry reconstruction errors due to occlusion, noise and imperfect silhouette extraction from the input data. Cagniart *et al.* [CBI10b] propose a probabilistic model that can cope with such issues, deforming a reference model using a Bayesian framework which takes into account uncertainties in the acquisition process.

All approaches for surface tracking and alignment mentioned above suffer from the same source of error: their reliance on frame to frame correspondence or deformation. Hence, sequential approaches have three inherent limitations: accumulation of errors in frame-to-frame alignment resulting in drift in correspondence over time; gross-errors for large non-rigid deformations which occur with rapid movements requiring manual

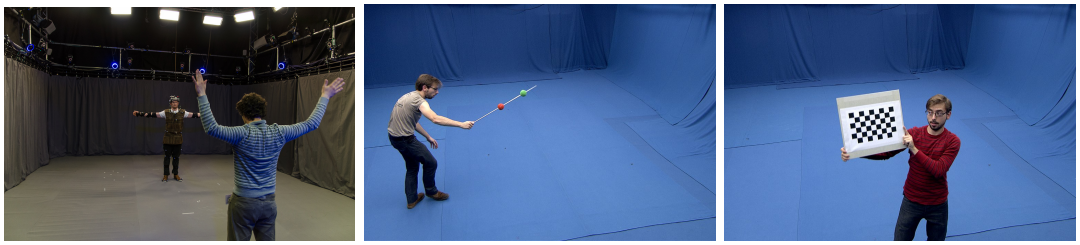
correction; and sequential approaches are limited to alignment across single sequences. Recently, non-sequential alignment approaches [HBH11, BHKH13] have been introduced to overcome these limitations, allowing the construction of temporally coherent 3D video sequences from multiple view performance capture database, as used in this work.

Huang *et al.* [HBH11] faced the problem of aligning multiple non-rigid mesh, 3D video, sequences into a single temporally consistent representation They propose a non-sequential approach based on a global alignment graph structure which used shape similarity [HHS10a] to identify frames for potential inter-sequence mesh deformation. Graph optimisation is performed to minimise the total non-rigid deformation required to deform the set of sequences into a coherent geometric structure. Results demonstrate that sequences deformed into a temporally aligned mesh representation accurately maintain the original captured shape and non-rigid motions. Further research in non-sequential approaches introduced *shape similarity trees* [BHH11], which propose a hierarchical scheme for non-sequential matching of frames across a sequence, generating a tree structure which represents the optimal path for alignment of all frames of a sequence, minimising the change in shape. Recently, Huang *et al.* [HHS10b] introduced a general formulation of the non-rigid alignment problem, not considering the inter-sequence alignment problem but the alignment across all frames of the database instead. The optimal *shape similarity tree* is defined as the minimum spanning tree containing all meshes in the input databases, edges costs are assigned according to the shape similarity [HHS10a]. Global alignment is achieved by pairwise non-rigid alignment based on the edges in the optimal shape similarity tree using any existing sequential non-rigid alignment techniques [dST*08, CBI10a, CBI10b].

2.2 A 4D Performance Capture Studio System

Actor performance is captured in a controlled studio environment using a multiple camera system for synchronised video acquisition described by Hilton and Starck [SH07b]. Nevertheless, the approach for 4D performance capture detailed in this chapter could be applied to any multicamera capturing setup. Likewise, any 4D motion dataset gener-

(a) *Studio #2 setup* **(b)** *Actor performance* **(c)** *Live camera feed*



(d) *Actor T-pose calibration* **(e)** *Extrinsic calibration* **(f)** *Intrinsic calibration*

Figure 2.1: *Shots taken during a camera setup and capture session.*

ated through other approaches available in the literature [CTMS03, dST*08, VBMP08, GSdA*09] could be also used as a input for the methods presented in the rest of the chapters of this thesis.

Throughout this work two different studio setups are used, referred as *Studio #1* and *Studio #2* in this thesis, consisting of 8 and 10 conventional HD cameras respectively. Following the evaluation and recommendations on multi-camera production studios published by Starck *et al.* [SMN*09], cameras were located equally spaced around a circle of 8 meters diameter about 2 meters above the studio floor. Cameras capture frames at 25 *fps* frequency rate, 1920 $\times$ 1080 pixel resolution, and use a shutter speed of 1/250 in *Studio #1* and 1/100 in *Studio #2*. In order to facilitate the postprocessing step that requires background and foreground segmentation [SH07b], *Studio #1* is surrounded by conventional blue curtains. Similarly, *Studio #2* uses grey light-reflective curtains that reflect the light emitted by blue LED rings located around each of the camera lenses, producing a blue background in the captured frames and reducing the undesirable blue projection over the captured surface cause by traditional blue curtains. Figure 2.1a show the grey light-reflective curtains and the blue LED rings. Figure 2.1b shows how a captured frame in *Studio #2* looks, notice the blue background appearance

from the reflected led illumination.

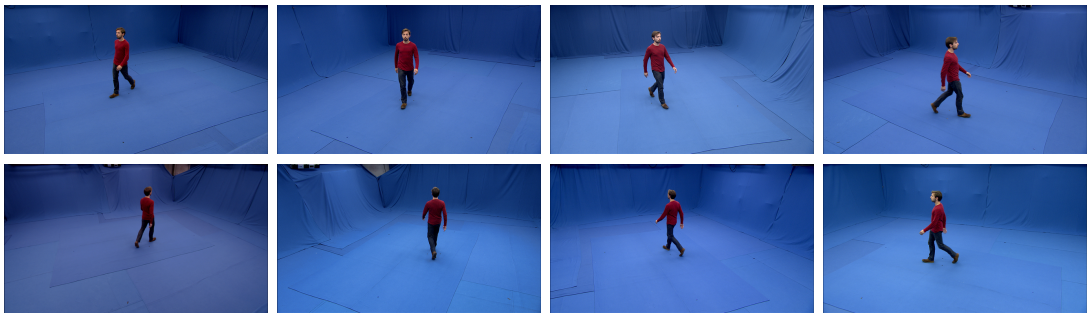
Extrinsic camera calibration is computed using the method also described by Starck and Hilton. A moving wand with two spherical markers at a known distance apart is captured from all cameras and a set of point correspondences between views is computed, see Figure 2.1e. Intrinsic calibration is performed using standard chessboard approach [opea], see Figure 2.1f. Actor performance, depicted in Figure 2.1b, starts by capturing a T-pose of the actor that will be used as a neutral-pose reference model, see Figure 2.1d. This pose is not strictly required in the reconstruction stage, but it is useful for future applications.

Figures 2.2a and 2.2c show frames recorded from the different view points captured in each of the studio setups. Foreground and background are then extracted from the input frames using standard chroma-key segmentation, generating silhouettes shown in 2.2b and 2.2d.

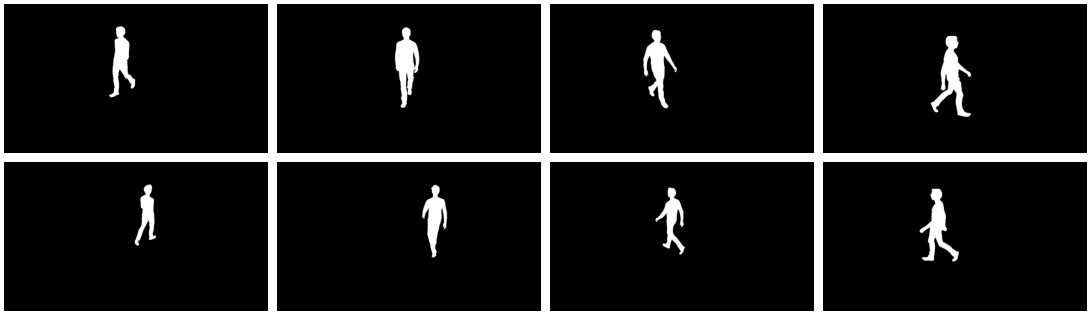
2.2.1 Shape reconstruction

Shape reconstruction is performed on a frame-by-frame basis using a multiple view silhouette and stereo approach building on state-of-the-art graph-cut optimisation techniques [SCD*06, SH07b]. This results in a reconstructed geometry that preserves non-rigid dynamic surface details such as cloth wrinkles and hair present in the original sequence, up to a 1.0 *cm* resolution detail. The resulting geometry consists of an unstructured mesh sequence with both the vertex connectivity and geometry changing from frame-to-frame. However, the character animation pipeline presented in this work requires a set of temporally aligned mesh sequences for multiple motions with the same mesh structure at each frame across all sequences.

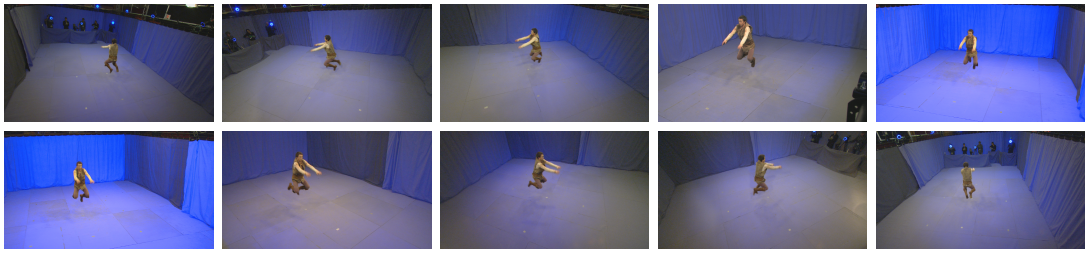
As discussed in Section 2.1.4, different sequential mesh alignment algorithms have been recently introduced, both model-based [dST*08] and model-free [BBK07, CBI10b, TM10]. Non sequential alignment approaches have also been recently investigated [HBH11, BHH11], providing robust techniques that are able to handle larger non-rigid deformations in consecutive meshes of the reconstructed captured sequences. In this



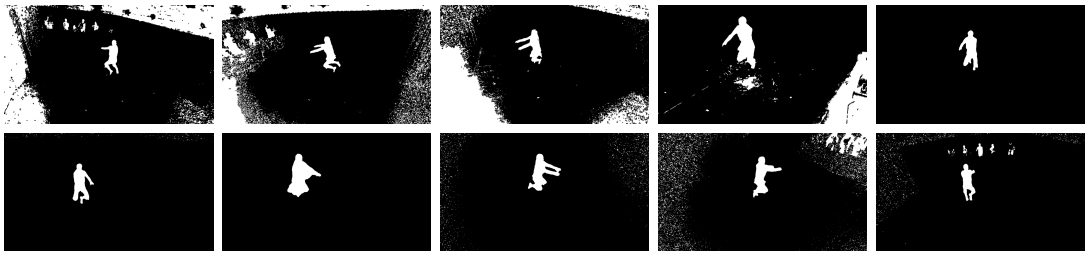
(a) *Studio setup #1, using 8 cameras, capturing a walk sequence.*



(b) *Silhouettes extracted from the input frames in Figure 2.2a.*



(c) *Studio setup #2, using 10 cameras, capturing a jump sequence.*



(d) *Silhouettes extracted from the input frames in Figure 2.2c.*

Figure 2.2: *Example of data acquired in each of the studios used throughout this work.*

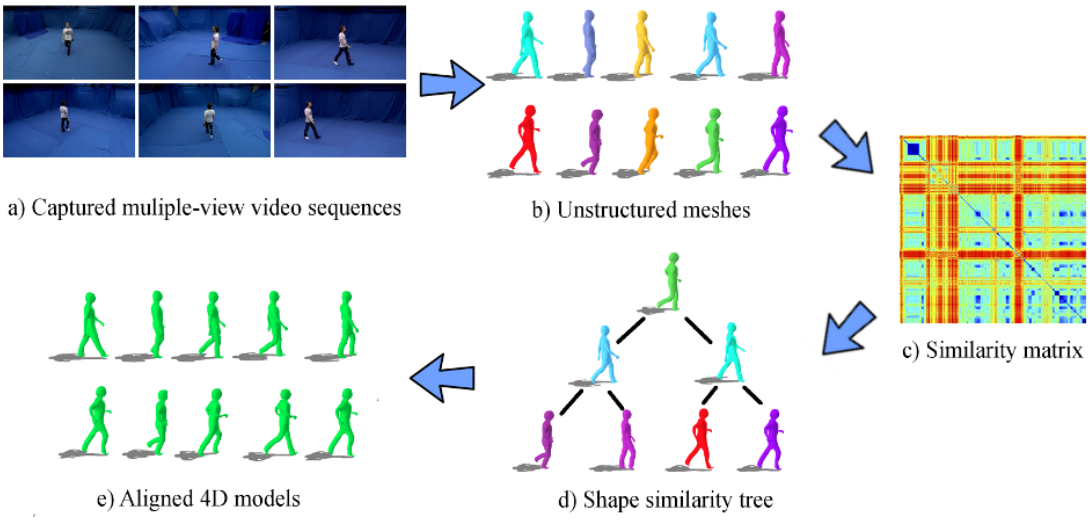


Figure 2.3: *Mesh sequence alignment pipline used throughout this work.*

work we use a non-sequential alignment approach to recover the non-rigid surface motion and represent all frames with a consistent structure based on a state-of-the-art approach presented by Budd *et al* [BHKH13]. Our pipeline follows the steps depicted in Figure 2.3. Although this approach is not a contribution of this thesis, it is highly relevant for this work and has been widely tested and used to generate the 4D models used in the following chapters. Sections 2.2.2 and 2.2.3 give a brief summary of the methods for completeness.

2.2.2 Shape similarity tree

Alignment across multiple unstructured mesh sequences is performed by constructing an intermediate *shape similarity tree*. This represents the shortest non-rigid surface motion path required to align each frame. The shape similarity tree allows frames from different mesh sequences to be aligned based on a measure of surface shape and motion similarity. The representation also ensures robust alignment of mesh sequences in the presence of large non-rigid deformations due to fast motion where sequential frame-to-frame surface tracking approaches may fail. The shape similarity tree is used to recover the non-rigid surface motion and obtain a consistent mesh structure for all sequences.

Given a set of input mesh sequences $\mathbf{M_i} = \{M_i(t_u)\}_{u=1}^{N_i}$, where N_i is the number of

meshes of the i^{th} sequence and $i = [1, N]$, where N is the number of sequences, to construct the shape similarity tree we require a measure of similarity $s(M_i(t_u), M_j(t_v))$ between a pairs of meshes which can be evaluated without prior knowledge of the mesh correspondence. A number of similarity measures for mesh sequences taking into account both shape and motion have been investigated [TM10, HHS10a]. In this work we utilise the temporally filtered volumetric shape histogram [HHS10a] as a measure of shape and non-rigid motion similarity which has been shown to give good performance on reconstructed mesh sequences of people. Evaluation of shape similarity between mesh reconstructions for all frames across all sequences results in a similarity matrix as illustrated in Figure 2.3(c), where blue indicates high similarity and red low similarity between the two input sequences.

Shape similarity is used to construct a tree representing the shortest non-rigid surface motion path required to align all meshes $\{M_i(t_u)\}_{u=1}^{N_i}$ from multiple captured mesh sequences. Initially a complete graph Ω is constructed with nodes for all meshes $M_i(t_u)$ in all sequences $i = [1, N]$ and edges $e_{iujv} = e(M_i(t_u), M_j(t_v))$ connecting all nodes. Edges e_{iujv} are weighted according to the similarity measure $s(M_i(t_u), M_j(t_v))$. The shape similarity tree T_{sst} minimising the total non-rigid surface motion required for alignment can then be evaluated as the minimum spanning tree (MST) of the complete graph Ω.

$$T_{sst} = \arg\min_{T \in \Omega} \left(\sum_{(i,j,u,v) \in T} s(M_i(t_u), M_j(t_v)) \right) \tag{2.1}$$

Parallel implementation of Prim's MST algorithm requires $O(n \log n)$ time where n is the number of graph nodes [CHL01]. This is prohibitively expensive for the graph Ω which typically has $10^3 - 10^5$ nodes. In practice as can be observed from the similarity matrix, Figure 2.3(c), many mesh pairs have a low similarity and, therefore not suitable for pairwise alignment. To reduce the computational cost in constructing the shape similarity tree we prune edges in the graph Ω according to a minimum similarity threshold.This similarity threshold can be calculated automatically from the similarity matrix as the minimum of the maximum similarity for each row in the matrix. Setting the threshold in this way ensures that all frames have at least one tree connection

within the threshold. Computation time for a 1500 node graph is $< 2s$[1].

2.2.3 Mesh sequence alignment

The shape similarity tree T_{sst} defines the shortest path of non-rigid surface motion required to align the mesh for every frame across all sequence. Starting from the root node M_{root} we align meshes along the branches of the tree using a pairwise non-rigid alignment.

Non-rigid pairwise mesh alignment uses a coarse-to-fine approach combining geometric and photometric matching in a Laplacian mesh deformation framework [Sor06]. This builds on recent work using Laplacian mesh deformation for sequential frame-to-frame alignment over mesh sequences [dST*08, CBI10a]. Here we use both photometric SIFT features [Low04] and geometric rigid patch matching [CBI10a] to establish correspondence between pairs of meshes [HBH11]. The combination of geometric and photometric features increases reliability of matching by ensuring that there is a distribution of correspondences across the surface. Alignment is performed starting from a coarse sampling (30 patches) which allows large deformations and recursively doubling the number of patches in successive iterations to obtain an accurate match to the surface. Since estimated feature correspondences are likely to be subject to matching errors we use an energy based formulation to introduce feature matches as soft constraints on the Laplacian deformation framework as proposed in [Sor06]:

$$\bar{x} = \arg\min_{x} \|Lx - \delta(x_0)\|^2 + \|W_c(x - x_c)\|^2 \tag{2.2}$$

L is the mesh Laplacian, $\delta(x_0)$ are the mesh differential coordinates for the source mesh with vertex positions x_0. x is a vector of mesh vertex positions used to solve for $Lx = \delta$. x_c are soft constraints on vertex locations given by the feature correspondence with a diagonal weight matrix W_c. A tetrahedral Laplacian system [dST*08] is used based on the discrete tetrahedron gradient operator G [BS08] with $L = G^T DG$, where D is a diagonal matrix of tetrahedral volumes. Equation 2.2 solves for the non-rigid

[1] timings are single threaded on an Intel Q6600 2.4GHz CPU

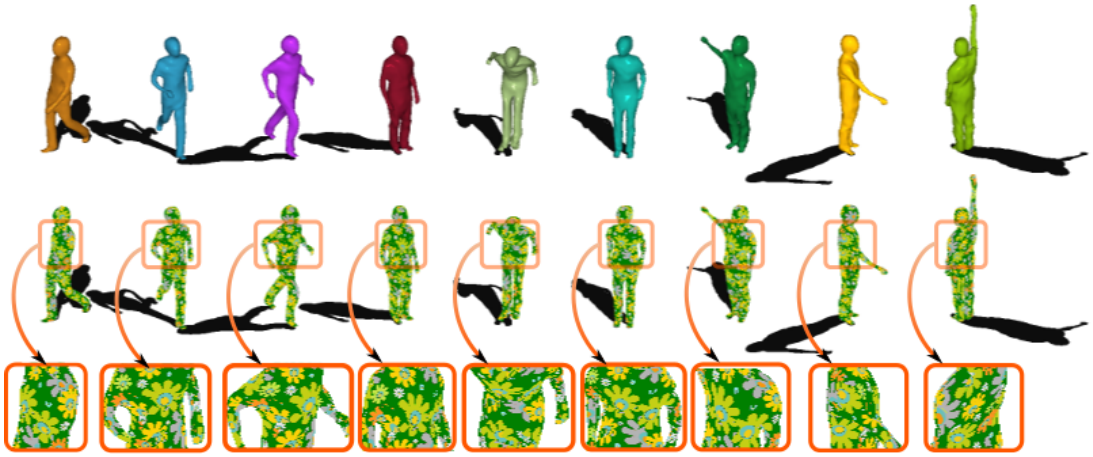


Figure 2.4: *Mesh alignment results. Top: reconstructed frames of a dataset containing walk, jog, jumps and reaching motions, each of them coloured differently to represent non-aligned geometry. Middle: aligned frames, coloured using a flower patter to show the performance of the non-sequential alignment algorithm. Bottom: close-up of the alignment, to highlight robustness of pattern location across all frames.*

deformation which minimises the change in shape whilst approximating the feature correspondence constraints.

Pairwise non-rigid alignment across the branches of the shape similarity tree T_{sst} results in known correspondence between the root mesh M_{root} and all other meshes $M_i(t_u)$. This correspondence allows every mesh to be resampled with the structure of the root mesh giving a consistent connectivity for all frames over all captured mesh sequences.

As a result of the 3D shape reconstruction process [SH07b] and the subsequent alignment step [BHKH13], a database multicamera capture is initially reconstructed into a set of unaligned mesh sequences, and then converted into a temporally coherent mesh representation in all frames, creating a 4D video dataset.

To test the proposed global non-sequential alignment approach, a dataset of 8 different unaligned mesh motions (such as walk, run, jumps, etc..), each of them between 20 and 40 frames long, was used. In Figure 2.4, the top row shows non-aligned meshes, randomly coloured to represent difference in geometry. The middle row shows aligned meshes using a flower pattern applied over the mesh, and the bottom row shows a close-up of the alignment to highlight the pattern consistent location across all frames

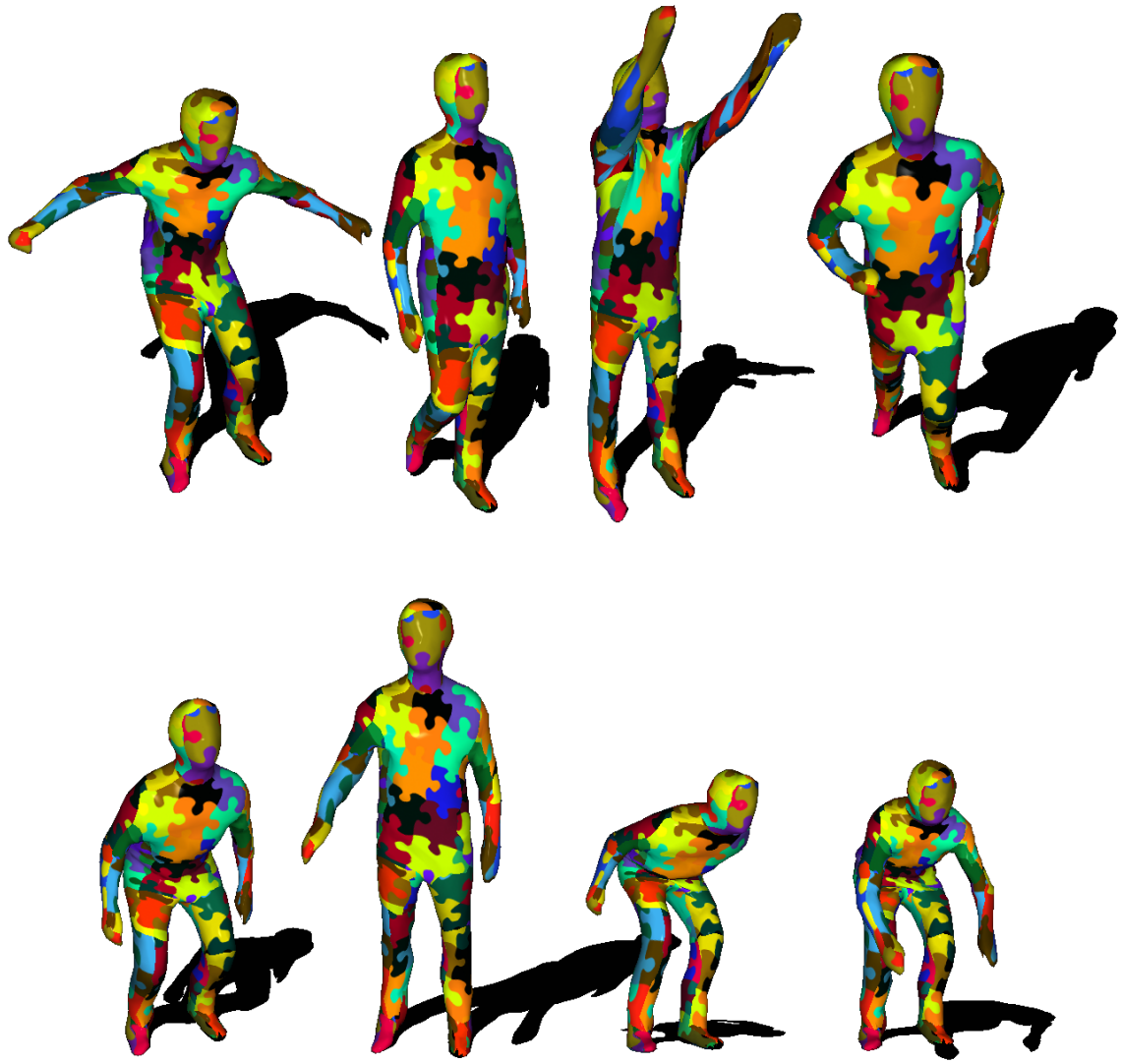


Figure 2.5: *Global non-sequential mesh alignment results. A puzzle pattern texture is used to highlight alignment robustness across different poses of the Dan dataset.*

to emphasise mesh alignment. Figure 2.5 presents higher resolution results for the proposed non-sequential alignment approach in the same dataset. Notice how the puzzle pattern remains fixed on the same body parts across all poses, showing the robustness of the proposed approach for global mesh sequence alignment.

2.3 Captured characters

In order to generate results for the methods introduced in the following chapters of this thesis, five different characters were captured using the studio setups mentioned in Section 2.2. Each character has distinct surface attributes, which allowed us to build an extensive dataset of motions containing a large range of both surface dynamics and appearance. Captured motions and character attributes are listed below, full sequences details are given in table 2.1.

Character Roxanne

Captured in *Studio #1*, a female character with long brown hair. She wears tight shorts, t-shirt, and a pony tail. The dynamics present in her hair are particularly difficult to reconstruct and align due to the rapid change in shape, especially in motion such as jog and jump. Figure 2.6 presents three arbitrarily selected frames from three different cameras to give an example of how the source data looks. Figure 2.7 illustrates reconstructed frames of her walk sequence.

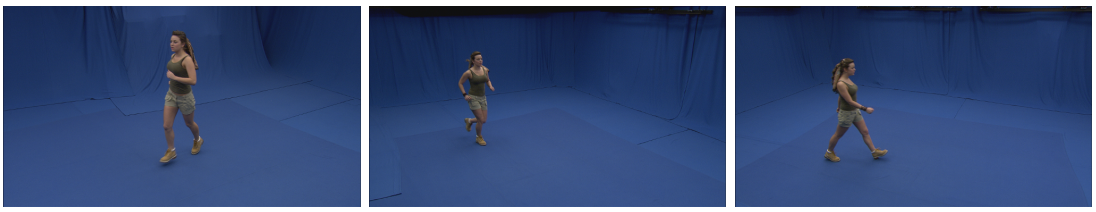


Figure 2.6: *Captured source frames used for reconstruction (arbitrarily selected for illustration purposes).*

Figure 2.7: *Reconstructed frames 3, 6, 9, 10 and 15 of the 'walk' motion of the 'Roxanne' character.*

Character J.P.

Captured in *Studio #1*, it is a male break-dancer character wearing loose clothing. He was captured performing a wide range of dancing motions, including back-flips and cartwheels. Figure 2.8 shows 3 arbitrarily selected frames to show an example of how the source data looks. Figure 2.9 shows reconstructed frames of his performance, notice the baggy trousers and loose t-shirt.

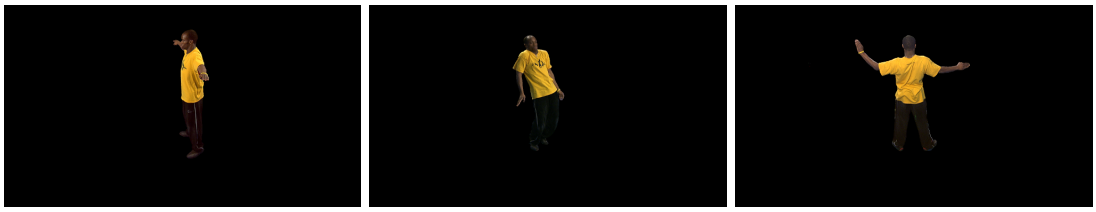


Figure 2.8: *Sample of source video frames (after background subtraction) used for reconstruction (arbitrarily selected for illustration purposes).*

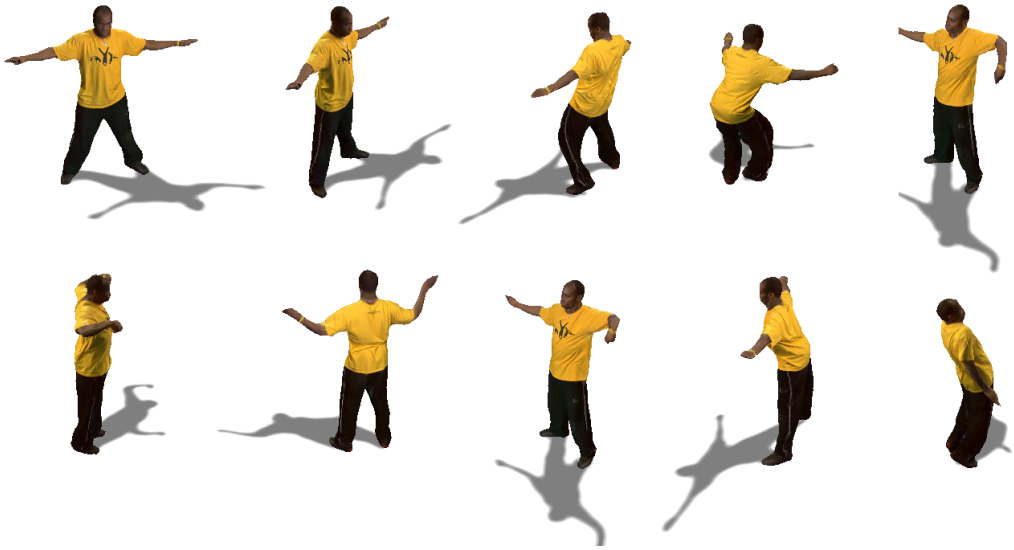


Figure 2.9: *Reconstructed frames 2, 7, 9, 13, 15, 25, 28, 34 and 39 of the 'pop' motion of the JP character.*

Character Dan

Captured in *Studio #1*, a male character wearing a red sweater and jeans. Overall his clothes are fairly tight, wrinkles are present in both jeans and sweater. Figure 2.10 shows three arbitrarily selected frames from sequences *walk*, *high jump* and *high reach* to show an example of how the captured source data looks. Figure 2.11 top row shows frames from a *short jump* sequence. Figure 2.11 bottom row shows frames of a high reach sequence.

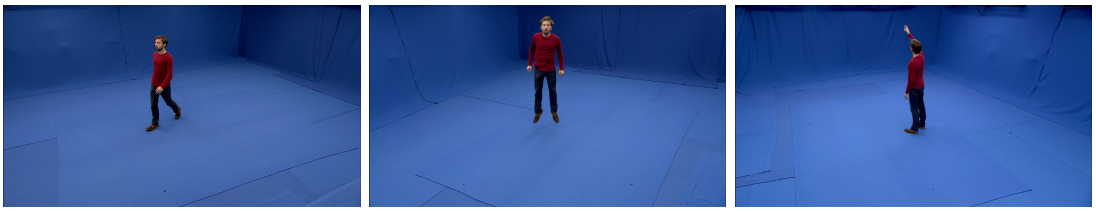


Figure 2.10: *Captured source frames used for reconstruction (arbitrarily selected for illustration purposes).*

Figure 2.11: *Top row: Reconstructed frames 1, 10, 12, 14 and 18 of the 'short jump' motion of the 'Dan' character. Bottom row: Reconstructed frames 1, 3, 7, 9 and 12 of the 'reach high' motion of the 'Dan' character.*

Character Infantry

Captured in *Studio #2*, it is a male character dress up with an infantry uniform from the Middle Ages. He wears brown tights and a dusty white shirt, with a leather vest on top and a thick brown leather belt. The vest is loose and long, the bottom part dynamics are similar to the motion of the cloth of a thick skirt. Figure 2.12 shows three arbitrarily selected frames from sequences *long jump*, *jog* and *standing* to show an example of how the captured source data looks. Figure 2.13 top row and bottom row show frames of short jump and long jump sequences, respectively.

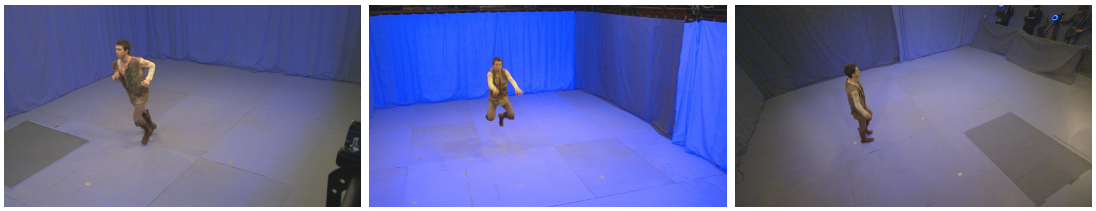


Figure 2.12: *Sample of captured source frames used for reconstruction (arbitrarily selected for illustration purposes).*

Figure 2.13: *Top row: reconstructed frames 1, 4, 6, 8 and 12 of the 'low jump' motion of the 'Infantry' character. Bottom row: reconstructed frames 3, 5, 8, 10 and 14 of the 'long jump' motion of the 'Infantry' character.*

Character Knight

Captured in *Studio #2*, it is a male character wearing a medieval knight dress. His costume includes white tights, a metallic upper-body chainmail armour and a leather cape on top. Both the chainmail and the cape contain highly non-rigid dynamic motions. These properties make the *Knight* the most challenging character investigated in this thesis. Figure 2.14 shows three arbitrarily selected frames to show an example of how the source data looks. Figure 2.15 top and bottom rows present reconstructed frames of the walk and the jog sequences, respectively.

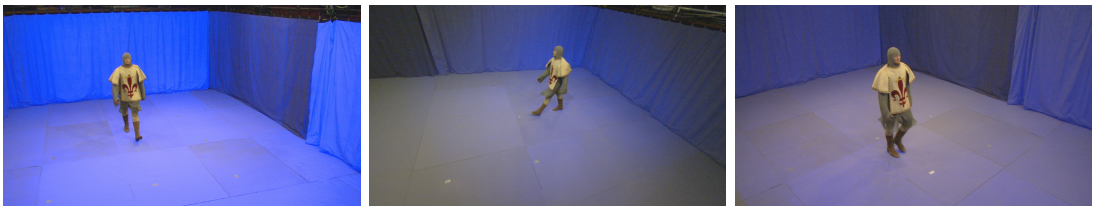


Figure 2.14: *Sample of captured source frames used for reconstruction (arbitrarily selected for illustration purposes).*

Figure 2.15: *Top row: Reconstructed frames 5, 7, 9, 10 and 12 of the 'walk' motion of the 'Knight' character. Bottom row: Reconstructed frames 2, 5, 6, 8 and 12 of the 'jog' motion of the 'Knight' character.*

Character	Studio	# vertices	Motions	Frames
Roxanne	Studio #1	2886	walk	23
			jog	21
JP	Studio #1	5580	flashkick	199
			free	499
			head	249
			kickup	219
			lock	249
			pop	249
Dan	Studio #1	2667	walk	29
			jog	19
			walk left	27
			walk right	29
			walk-to-stand	32
			stand-to-walk	31
			high jump	58
			low jump	21
			long jump	63
			short jump	36
			one-arm reach low	70
			one-arm reach high	100
			two-arms reach low	171
			two-arms reach high	161
			punch	20
			ducking	54
Infantry	Studio #2	4052	walk	32
			jog	19
			walk left	27
			walk right	29
			walk-to-stand	32
			stand-to-walk	31
			high jump	58
			low jump	21
			long jump	63
			short jump	36
Knight	Studio #2	4058	walk	32
			jog	21
			walk left	31
			walk right	30

Table 2.1: *Captured characters*

Chapter 3

Mesh Sequence Parametrisation

Inspired by previous research introducing methods to parametrically control skeletal motion capture (MoCap) sequence data [WH97, RCB98, PSS02, KGP02, KG04], our goal is to find analogous methods to interactively control a character created from 4D performance capture data. This chapter demonstrates that interactive animation from temporally aligned mesh sequences can be obtained by combining multiple captured clips, enabling continuous real-time control of movement. Using intuitive high-level parameters such as speed and direction for walking or height and distance for jumping, the user can control a virtual avatar built from blended 4D data that maintains the realism of the captured sequences.

A naive approach to combine mesh sequences is to linearly blend vertex positions. However, this may lead to unnatural geometric errors such as mesh shrinking and collapsing. In the last decade, non-linear mesh deformation approaches [ACOL00, SZGP05, LSLCO05, Sor06, XZY*07, KG08] have been proposed, offering more natural solutions to mesh deformation. However, these approaches are not suitable for interactive applications due to their relatively high computational cost. Typically non-linear mesh deformation requires iterative solution of an energy minimisation or matrix inversion problem which is not readily applicable to real-time applications even with GPU implementation.

In order to fulfil our requirement for real-time interpolation of 4D data, this chapter introduces a hybrid piece-wise linear approach for mesh blending [CTGH11, CTGH12b]

that combines the real-time performance of the linear methods with the natural deformation of non-linear approaches. The proposed approach is used to interactively combine captured mesh sequences, allowing real-time parametric control to synthesise novel 4D video data. Results of the proposed method are shown and evaluated.

3.1 Related Work

To date the primary focus for research in 4D performance capture has been free-viewpoint video replay [CTMS03, SH07b, dST*08, VBMP08] without modification or editing of the content. The lack of temporal coherence in the mesh sequence has prohibited the development of simple methods for manipulation. As discussed in Section 2.2.3, recent research has successfully achieved temporal alignment across mesh sequences [CBI10a, HBH11, BHH11, BHKH13], allowing the representation of a complete database of multiple 4D performance capture sequences with the same mesh topology. Such progress in mesh reconstruction and alignment has enabled the possibility of synthesising novel mesh sequences by combining reconstructed data.

On the other hand, in the last two decades many different approaches for skeletal motion capture (MoCap) data manipulation, ranging from methods for motion interpolation to pose editing, have been investigated. MoCap sequences typically represent the human motion as skeletal joint angles with 30-60 degree of freedom. To allow the flexible reuse of MoCap data in animation techniques have been introduced for editing and manipulation of MoCap sequences.

3.1.1 Parametrisation of Skeletal Motion Capture Data

Initial research on skeletal MoCap combined several clips to create novel motions. Wiley and Hahan [WH97] pioneered the mixing of motions to expand a dataset of MoCap data while retaining the quality of the original data. Their approach, based on linear interpolation of joint angles, does not guarantee the synchronisation in key-events on the motions, hence it is susceptible to artefacts such as foot sliding and unrealistic animation. Furthermore, the computational cost of the approach increased exponentially

with the number of motion samples. Linear blending of joint angles for parametric skeletal motion synthesis results in plausible motions because the joint-angle space represents the non-linear relationship between degrees of freedom and the Euclidean space in which the character moves. Rose *et al.* in their *Verbs and Adverbs* [RCB98] combined different styles of walk to interactively animate a character, allowing the user to control not only speed and direction but also the avatar's mood. A combination of radial basis functions and low order polynomials is used to create the interpolation space between example motions. Inverse kinematic constraints are used to avoid artefacts such us foot sliding. Following the same ideas, Park *et al.* also blend motions to create interactive parametric avatars [PSS02]. A new approach for pose interpolation based on quaternion algebra was introduced, ensuring correct pose and orientation. Speed and direction of the avatar were parametrically controlled by the user, while the actual path of the root was determined through a user-specified trajectory. Further improvements in parametric blends were presented also by Rose *et al.* [RISC01], introducing a new scheme for pose interpolation based on cardinal basis functions, greatly increasing the efficiency of the system. Additional samples, so-called *pseudo-examples*, derived directly from the source interpolation space designed by the artist were computed in order to improve the accuracy of scattered data interpolation.

Motion parametrisation methods require interpolated motions to be *semantically similar*, otherwise they fail to produce believable human poses. To fullfill this requirement, previous research [WH97, RCB98, PSS02] assumed that input clips were *similar* in motion, a task that was done by manually selecting the right clip segments. In order to overcome this laborious step, Kovar and Gleicher proposed an approach to automatically identify and parametrise motions in large data sets [KG04]. A pose search method based on numerically similar matches are used as intermediaries to find more distant matches. Furthermore, they introduced an automatic scheme for parametrising a space of blends according to user-specified motion features. This high-level parametric control of the motion allows the animator to control the desired motion by intuitive parameters rather than blending factors. Initial research to achieve intuitive parametric control was first investigated by Ahemed *et al.* [AMH01].

As captured motion data is a set of time-varying signals, statistical techniques have also

been investigated for motion synthesis and representation [BW95, MTH00, Bow00, AM00, PB02]. Molina-Tanco and Hilton presented a two-level approach to author novel motions from a database of motion capture example [MTH00]. A Markov chain of joint trajectories is built in the first level, enabling the generation of the overall motion path. The second level of the model matches the states of the Markov chain with actual segments of the captured motions, resulting in a novel synthesised realistic motion. Low-dimensional representation methods were also investigated by Bowden, who used K-means clustering and hierarchical Principal Component Analysis, PCA, to built a Markov chain that models a motion example [Bow00]. However, this approach is ill-suited for motion synthesis since its lack of flexibility: once the first state is chosen, the resulting action is either fixed or random. Alexa and Müller [AM00] also used PCA for 3D animation data reduction. Pullen and Bregler used signal processing to analyse human motion, dividing the data into frequency bands [PB02]. Frequency analysis was perform to match keyframed animation with captured motion in order to enhance the final synthesised motion. Mukai and Kuriyama [MK05] proposed a method that treats motion interpolations as statistical predictions of missing data in an arbitrarily definable parametric space. This approach relaxes the problem of spatial inconsistencies, such as foot-sliding, that occurred with previous methods.

Further research in synthesising novel motion by the reutilisation of captured clips focused not only on motion interpolation but also motion concatenation, leading to a graph representation that encapsulates both motion and transitions. This representation, usually referred as *Motion Graphs* [KGP02], was first used by Molina-Tanco and Hilton [MTH00]. A number of techniques [AF02, LCR*02, AFO03, IF04, HG07] introduced similar approaches for automatically identify logically similar motions in a data set, enabling the transition between a set of captured motions allowing to synthesise novel clips. Such approaches, mostly based on motion clip concatenation, are discussed in more detail in Section 4.1.

3.1.2 Parametrisation of 4D Captured Data

Due to the relatively recent introduction of techniques for temporally coherent surface motion capture [SH07b, dST*08, VBMP08, BHKH13] there has been relatively little research on parametrisation and animation control of reconstructed mesh sequences. Nevertheless, in the last decade researchers have investigated ways of editing and parametrising synthetic mesh sequences to increase the versatility of each motion clip.

Naive linear interpolation of vertex positions produces visual artefacts such as mesh shrinking, shortening and collapsing due to the nature of the Cartesian coordinates, which only encodes global information of each point. In order to overcome this limitation, non-linear methods for mesh interpolation have been investigated to synthesise novel parametric shapes. Lewis *et al.* observed that human shape interpolation could be uniformly represented as mappings from a *pose space*, defined by an underlying skeleton, to displacements in the objects local coordinate frame [LCF00]. This consideration enabled Lewis *et al.* to introduce a system for shape interpolation that avoids problems associated with linear shape interpolation, where nonlinearities were expressed in terms of skeletal deformations. A similar approach also based on radial basis interpolation for articulated shapes was presented by Sloan *et al.* [SRC01]. Allen *et al.* presented a shape interpolation scheme for 3D human models obtained through a set of static whole-body [ACP02]. However, unlike Lewis *et al.* [LCF00], they do not require temporarily coherent mesh models. After fitting a skeleton into the scanned model, shapes are combined using a k-nearest neighbour interpolation in pose space.

Skeleton-based techniques to approximate mesh kinematics fail in providing the rich range of deformations that a surface can adopt. Alexa *et al.* [ACOL00] proposed a different approach for mesh interpolation that preserves local geometric information without the requirement of articulated shapes . In contrast, they presented a more general scheme that blends the interior of shapes with minimal local distortion. Shapes are initially decomposed into isometric representations and then the corresponding vertex are interpolated using a least-distorting triangle-to-triangle morphing approach, minimising the paths of the vertices. Inspired by previous research in skeleton-based

inverse kinematics, further research in skeletonless mesh deformation was presented in *Mesh-Based Inverse Kinematics*, MeshIK, where a space of meaningful deformations is created from a set of example meshes [SZGP05]. A vector of features that encodes important shape properties is extracted from each example mesh. Edited meshes are reconstructed using feature vectors by solving a least squares minimisation problem for the free vertices while enforcing constraints in the vertices that the user moves.

Sorkine and Lipman present advances in geometry processing related to the Laplacian processing framework and differential surface representations [Sor06, LSLCO05]. In contrast to the traditional global Cartesian coordinates, which only encodes information about spatial location of each point, differential representations encodes information about the local shape of the surface, including the size and orientation of local details. This representation increases the robustness and realism of operations such as mesh editing and interpolation with respect previous approaches. Xu *et al.* [XZY*07] present a method that generalises previously introduced gradient domain editing techniques [HSL*06] to deform mesh sequences. Given a set of sparse and irregularly distributed keyframe constraints, their approach adjusts the meshes at the keyframes so satisfy these constraints, and then smoothly propagate the constraints and deformations at keyframes to the whole sequence to generate a new deforming mesh sequence. Kircher and Garland [KG08] propose an approach for editing free-form deformation of surfaces like cloth and faces. Their method is based on a differential surface representation that is invariant under rotation and translation and which is well suited for surface space-time editing. This method opened up a broad range of possible motion alterations including motion blending and keyframe animation. Recently, Tejera and Hilton [TH11] presented a shape constrained Laplacian mesh deformation framework for key-frame interactive mesh sequence editing. The learnt deformation space of motion ensures both preservation of the captured motion characteristics and underlying anatomical structure of the actor.

Nevertheless, most of the proposed techniques for mesh editing and blending require solution of a least-squares minimisation which is prohibitively expensive for interactive animation. Methods for real-time blending of captured 4D mesh sequences are required to create interactive characters from performance capture. Such methods need to be

both fast enough to run at interactive rates >25*fps* and robust enough to provide mesh deformations that maintain the captured realism.

The methods described above in this section aim to synthesise 3D mesh poses by editing or interpolating 3D mesh models, techniques that can lead to generation of novel 3D mesh sequences. Other approaches [SMH05, XYA06, HHS09, XLS*11] for 3D mesh sequence synthesis rely on mesh sequence editing together with concatenation of clips. In such approaches, analogous to *Motion Graph* for skeletal MoCap data [AF02, LCR*02, AFO03, IF04, HG07], similarities between captured 3D mesh motions are evaluated, enabling linking of different actions through frames with high shape similarity. Research in methods for 3D mesh sequence concatenation is discussed in more detail in Section 4.1.

3.2 Parametric Motion Control of Mesh Sequences

4D video data reconstructed from multi-camera capture enables the replay of the captured motions from any viewpoint, reproducing the non-rigid surface dynamics present in the captured character [BHKH13]. With increasingly availability of 4D video datasets an interest in reusing captured motions has appeared in the Computer Graphics community. The reutilisation of reconstructed clips could potentially allow authoring of novel 4D motions maintaining the realism of the captured data. Our goal is to find methods to combine multiple captured 4D sequences, and also capable of performing this online. Achieving such a goal would enable interactive real-time control of a 4D virtual character created from captured data.

Previous approaches for mesh sequence manipulation are limited by the lack of methods to interactively blend and edit the reconstructed models. Thus, our motivation is to investigate methods for real-time blending of 4D video sequences. Three steps are required to achieve high-level parametric control from mesh sequences: time-warping to align the mesh sequences; non-linear mesh blending of the time-warped sequences; and mapping from low level blending weights to high-level parameters (speed, direction, etc.). This section presents how the proposed approach fulfils these requirements.

Given a set of N temporally aligned mesh sequences $\mathbf{M} = \{M_i(t)\}_{i=1}^{N}$ of the same or similar motions (e.g. high jump and low jump) we are interested in finding a blended mesh sequence

$$M_B(t, \mathbf{w}) = b(\mathbf{M}, \mathbf{w}) \tag{3.1}$$

where $\mathbf{w} = \{w_i\}_{i=1}^{N}, w_i \in [0..1]$ is a vector of weights for each input motion and $b()$ is a mesh sequence blending function. We require this function to perform at online rates, $\geq 25\ Hz$, and also the resulting mesh $M_B(t, \mathbf{w})$ to maintain the captured visual quality of the source $\{M_i(t)\}_{i=1}^{N}$ meshes.

3.2.1 Time-warping

Each 4D video sequence of related motions (e.g. walk and run) is likely to differ in length and location of corresponding events, for example foot-floor contact. Thus, the first step for mesh sequence blending is to establish the frame-to-frame correspondence between different sequences.

Previous work on skeletal motion parametrisation assumed that individual mesh sequences $M_i(t)$ are temporally aligned by a continuous time-warp function $t = f(t_u)$ [BW95, WP95] which aligns corresponding poses of related motions prior to blending such that $t \in [0, 1]$ for all sequences.

Similarity, in our work, given two similar motion sequences $M_i(t)$ and $M_j(t)$ (e.g. walk and run) a similarity matrix containing shape similarity [HHS10a] for all possible pair of meshes is built. We then build a graph connecting each mesh $M_i(t)$ with both $M_i(t+1)$ and $M_j(t+1)$, assigning their shape similarity as a cost in its corresponding edge, as depicted in Figure 3.1, where arbitrary weights were assigned for illustration purposes. Dijkstra's shortest path optimisation algorithm is used to find the optimal path starting from the top-left node of the graph until reaching a leaf node. The resulting path is the optimal frame to frame correspondence between sequences $M_i(t)$ and $M_j(t)$. Manual annotations representing selected keyframes can be imposed as a hard constraints in the path optimisation.

Temporal sequence alignment helps in preventing undesirable artefacts such as foot skating in the final blended sequence. Key-frames containing poses such as foot-floor

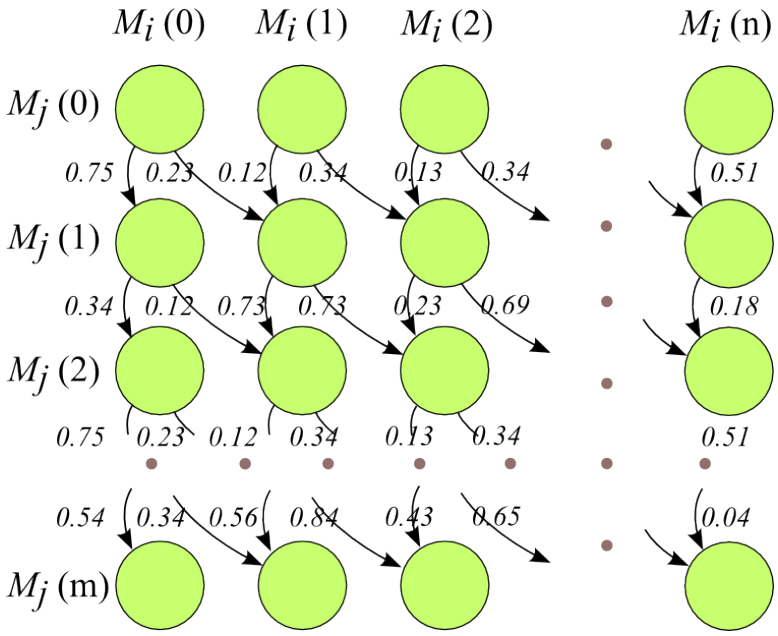


Figure 3.1: *Graph used to find corresponding frames between two different mesh sequences $M_i(t)$ and $M_j(t)$. Edges weights are assign using the similarity matrix built from computing the shape similarity [HHS10a] of all possible pair of frames (in this figure arbitrary given for illustration purposes). Dijkstra's path optimisation algorithm is used to find the optimal path.*

or hand-object contact are selected and provided to the time-warp function $t = g_{ij}(s)$ which defines the non-linear time warp between $M_j(s)$ and $M_i(t)$.

3.2.2 Real-time Mesh Blending

Previous research in 3D mesh deformation concluded that [LCF00] linear-methods for mesh blending, despite being computationally efficient, may result in unrealistic results. Non-linear methods [BS08] have appeared to overcome this limitations achieving plausible surface deformation, however the price paid is a significant increase in processing requirements. This results in non-linear method being unsuitable for online applications.

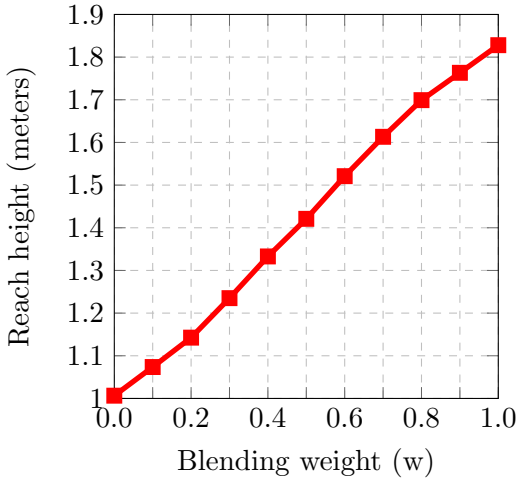
In this work we present a piece-wise linear approach for 3D mesh blending that combines the realistic deformation of the non-linear approaches with the real-time performance of the linear methods. Section 3.3 discusses in more detail the advantages and disadvantages of the current approaches and presents the blending approach that will be used

in the rest of this work. Section 3.4 presents and evaluation of the proposed method and shows examples of the results achieved using 4D video mesh data.

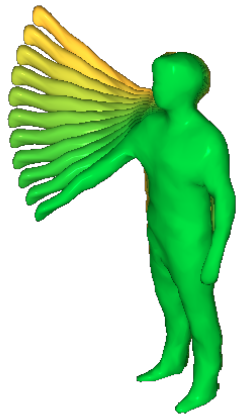
3.2.3 High-level Parametric Control

High-level parametric control is achieved by learning a mapping function $f(\mathbf{w})$ between the blend weights and user specified motion parameters $\mathbf{p}$. As in skeletal motion blending the blend weights do not provide an intuitive parametrisation of the motion. We therefore learn a mapping $\mathbf{w} = f^{-1}(\mathbf{p})$ from the user-specified parameter to the corresponding blend weights required to generate the desired motion. Motion parameters $\mathbf{p}$ are high-level user specified controls for a particular class of motions such as speed and direction for walk or run, and height and distance for a jump. The inverse mapping function $f^{-1}()$ from parameters to weights can be constructed by a discrete sampling of the weight space $\mathbf{w}$ and evaluation of the corresponding motion parameters $\mathbf{p}$ [AMH01].

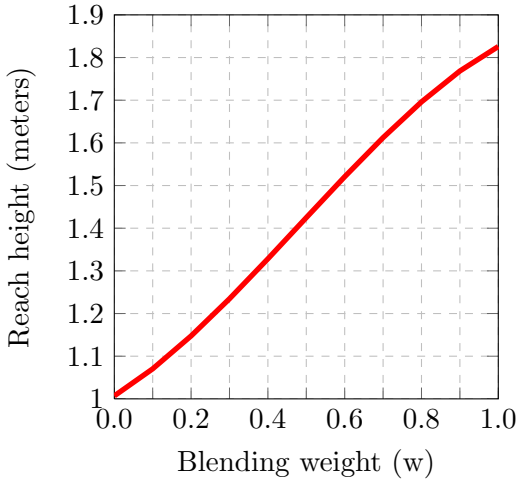
Figure 3.2 depicts an example of high-level mapping for a blended motion created combining a reach high and a reach low motion. Figure 3.2b shows both the captured high-reach motion (top yellow arm) and the low-reach (bottom green arm), as well as the in-between reach motions generated by non-linear blending. Figure 3.2a plots the actual hand position of the blended shapes depending on the blending weights w_i. We can observe that the relationship between the hand position and the blending weight is non-linear. In Figure 3.2c, using curve fitting, we find a 3^{rd} order polynomial that represents the relation $f(w) = h$, where w is the blending weight and h the actual height of the hand. In this particular example, least-square fitting gives the polynomial $f(w) = -0.0006w^3 + 0.0086 + w^2 + 0.0554w + 1.0070$. Finally, shown in Figure 3.2d, we find $f^{-1}(h) = w = 12.78h^3 - 53.84h^2 + 85.58h - 44.62$, enabling high-level reach control for the user: given a requested height h, the corresponding blending weight w can be automatically found.



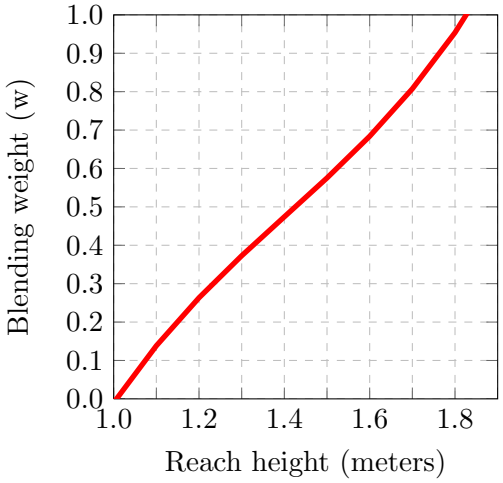
(a) *Hand position, in height, depending on the blending weight w of the blended shapes in Figure 3.2b.*



(b) *Yellow top and bottom green arms are the captured meshes. In-between poses are generated using a non-linear mesh blending scheme. Colour represents change in parameter w.*



(c) *Curve fitting to match points displayed in Figure 3.2a.* $f(w) = -0.0006w^3 + 0.0086 + w^2 + 0.0554w + 1.0070$.



(d) *Inverse mapping to find* $f^{-1}(h) = w$. *In this example,* $f^{-1}(h) = w = 12.7875h^3 - 53.8466h^2 + 85.5823h - 44.6235$.

Figure 3.2: *In order to achieve high-level parametric control, a mapping function $f^{-1}(h)$ is learned using the relation between blend weights and the actual blended mesh.*

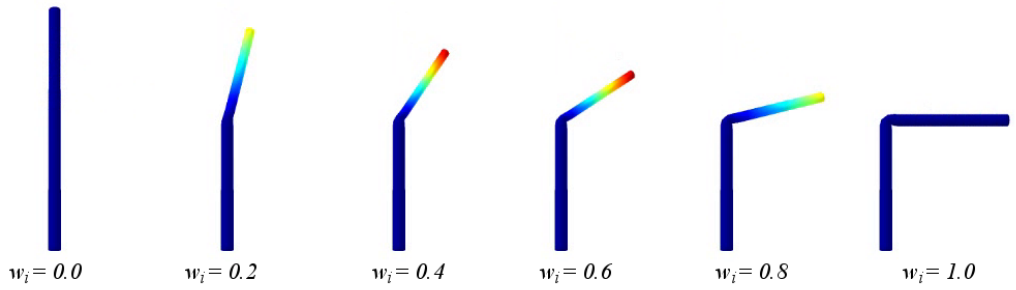


Figure 3.3: *Linear mesh interpolation between of 3D meshes according to Equation 3.2. Left and right: input meshes, including a straight tube and a 90° bended tube. Centre: in-between steps of the result of linearly blending the straight tube of the left and the bended tube of the right. Blended meshes fail in maintaining the original length and thickness of the object. Coloured using a heatmap to highlight errors in geometry caused by the linear approach, no error in dark blue to high error in red. Error is computed using vertices distance between linear and non-linear interpolation.*

3.3 3D Mesh Blending

As discussed in Section 3.1.2, methods for 3D mesh interpolation can be traditionally categorised into two different groups: linear and non-linear. This section discusses both approaches and introduces a novel approach that combines the robustness of the non-linear methods with the fast performance of the linear. These properties make the proposed approach suitable for real-time 3D mesh sequence parametrisation.

3.3.1 Linear Blending

Methods for parameterisation of skeletal motion capture have previously been introduced [RCB98, KG04, MK05, HG07] allowing continuous high-level movement control by linear interpolation of joint angles. An analogous approach for 3D mesh models consists in linearly blend its global vertex positions as follows

$$M_L(\mathbf{w}) = \frac{1}{\sum w_i} \sum_{i=1}^{N} w_i \mathbf{x_i}, \tag{3.2}$$

where $M_L()$ is the linearly blended mesh, $\mathbf{w} = \{w_i\}_{i=1}^{N}$ a vector of weights for N meshes, $\mathbf{x_i}$ a vector of vertex positions of the i^{th} mesh, and $w_i \mathbf{x_i}$ denotes the product of

the mesh vertex positions $\mathbf{x_i}$ by weight w_i. This technique is computationally efficient but may result in unrealistic deformations or mesh collapsing if there are significant differences in shape. This is depicted in Figure 3.3, which shows how a straight tube is progressively blended into a 90° bended tube using linear interpolation from Equation 3.2. The upper part of the shape of the tube is severely shortened when intermediate blending weights are used.

Such artefacts appear due to the nature of the linear approach: it only uses the global spatial location of each vertex, and ignores shape local information such as neighbour' distance, face orientation or size. In order to overcome these limitations, methods and operators that encapsulate the local information of each face of the original mesh are required.

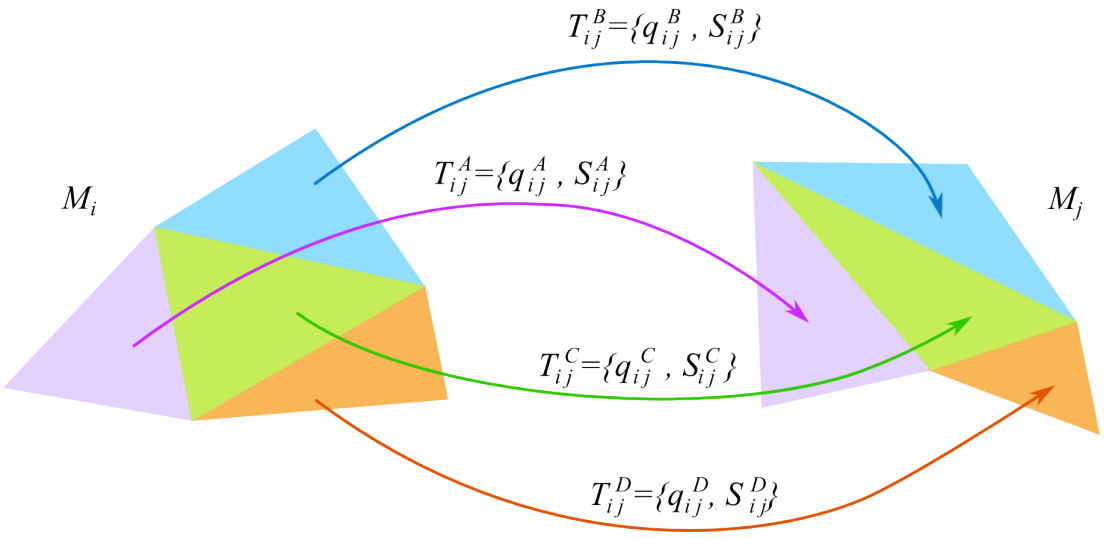
3.3.2 Non-Linear Blending

In the last decade, a range of non-linear mesh blending approaches have been introduced [SP04, Sor06, XZY*07, KG08]. These methods are based on differential surface representation [BS08] which encodes local information about shape surface details such as face orientation relative to neighbouring faces. This representation allows mesh blending and editing methods to produce plausible deformations, but commonly requires least-squares solution of a system of equations which is prohibitive for real-time interaction.

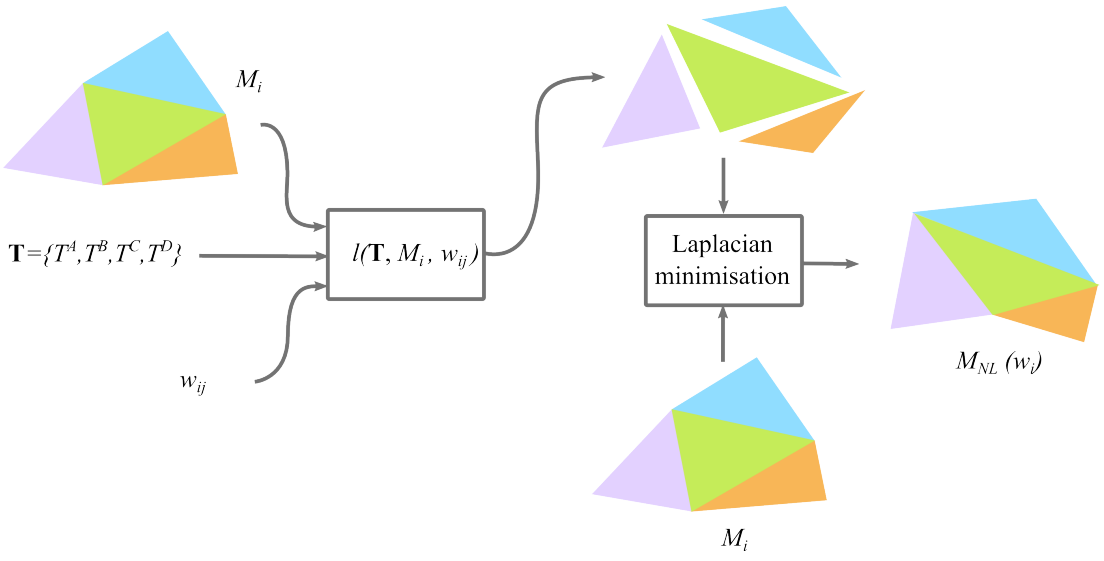
In this work we have used the following non-linear approach. Given a set of N temporally aligned meshes $\mathbf{M} = \{M_i\}_{i=1}^{N}$ of similar poses we want to compute a blended mesh deformation according to a set of weights $\mathbf{w} = \{w_i\}_{i=1}^{N}$, where $\sum_{i=1}^{N} w_i = 1$

$$M_{NL}(\mathbf{w}) = b(\mathbf{M}, \mathbf{w}) \tag{3.3}$$

where $b()$ is a non-linear blend function which interpolates the rotation and change in shape independently for each element on the mesh according to the weights $\mathbf{w}$ and performs a least-squares solution to obtain the resulting mesh $M_{NL}(t, \mathbf{w})$. Computation of triangle rotations q^k and scale/shear transformations S^k is performed using *slerp* and



(a) *A vector of affine transformations* $\mathbf{T} = \{T_{ij}^{A}, T_{ij}^{B}, T_{ij}^{C}, T_{ij}^{D}\}$ *is found for each pair of meshes* M_i *and* M_j



(b)

Figure 3.4: *Pipeline of the proposed non-linear mesh blending approach. From an input mesh* M_i*, a vector of transformations* $\mathbf{T}$*, and a blending weight* w_{ij}*, the function* $l()$ *computes a per-triangle interpolation that finally is used to create the resulting mesh after applying a Laplacian minimisation.*

linear interpolation, respectively, as shown in Equations 3.4 and 3.5:

$$q_i^k = slerp(q_i^k, q_j^k, w_i) \tag{3.4}$$

$$S_i^k = S_i^k + w_i(S_j^k - S_i^k) \tag{3.5}$$

where k denotes the index of each triangle, i and j refer to the pair of meshes (M_i, M_j) being blended and w_i is the relevant weight. Applying the transformations q^k and S^k results in a set of new triangles than can be linked back together using existing approaches [KG08, Sor06, SP04, XZY*07] to non-linear mesh editing to obtain natural deformation between the keyframes. In particular, we employ a Laplacian deformation framework [Sor06, TH11]. For $N > 2$, function $b()$ performs the non-linear interpolation iteratively for each pair of frames.

Figure 3.4 depicts the proposed approach. Figure 3.4a presents two example meshes M_i and M_j, both containing a set of triangles {A, B, C, D}, and the computed vector of transformations $\mathbf{T} = \{T_{ij}^A, T_{ij}^B, T_{ij}^C, T_{ij}^D\}$, where T_{ij}^k is the affine transformation between M_i and M_j of the k^{th} triangle. Figure 3.4b shows the proposed pipeline for non-linear mesh interpolation. From an input mesh M_i, a vector of transformations $\mathbf{T}$ and a blending weight w_{ij}, the function $l()$ applies a per-triangle weighted transformation resulting in an unconnected mesh created by the interpolated triangles. Using the local surface information from M_i, encoded through differential coordinates [Sor06, TH11], a minimisation of the Laplacian deformation reconnects the interpolated triangles, creating the resulting $M_{NL}(w_i)$ blended mesh.

This approach enables the computation of an interpolated mesh $M_{NL}(w_i)$ from two or more input meshes, while minimising the change of their original local properties. This helps in avoiding undesirable deformation artefacts such as mesh thinning, shrinking and collapsing. For $N > 2$ we apply the approach iteratively for each pair of frames.

An example of the results achieved with the proposed scheme is shown in Figure 3.5, where a 3D mesh of a straight tube is blended with a 90° bended tube. In contrast to the mesh shortening observed with linear interpolation, the Laplacian deformation results in intermediate meshes than maintain the original volume, length and thickness of the source models. However, the approach requires a relatively computationally expensive iterative non-linear optimisation which is not suitable for real-time animation.

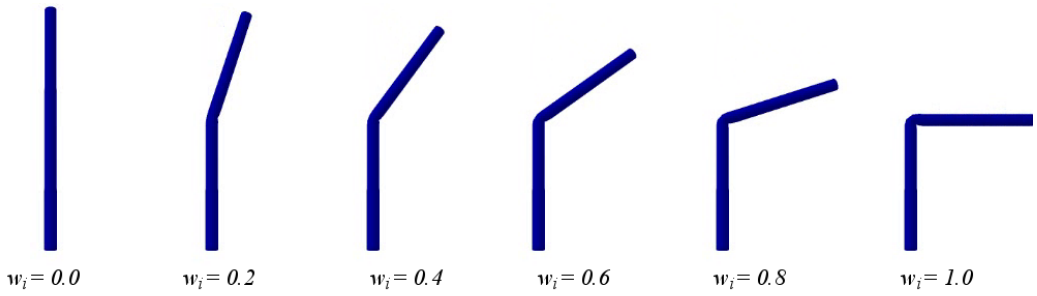


Figure 3.5: *Left and right: input pair of 3D meshes. Centre: blended meshes computed using the non-linear approach presented in Section 3.3.2 and blending weight 0.2, 0.4, 0.6 and 0.8 respectively. Notice how the volume, length and width of the input model is we kept, greatly improving the results achieve with a linar approach depicted in Figure 3.3.*

3.3.3 Hybrid Piecewise Linear Blending

In this work we introduce a hybrid piece-wise linear solution which approximates the non-linear deformation whilst maintaining real-time performance. Our approach for mesh blending exploits offline pre-computation of non-linear deformation for a small set of intermediate parameter values. Differences between the linear and non-linear mesh deformation are pre-computed and used to correct errors in linear deformation at run-time. This approach approximates the non-linear deformation to within a user-specified tolerance whilst allowing real-time computation with a similar cost to linear blending. The price paid is a modest increase in memory required to store intermediate non-linear mesh displacements for blending.

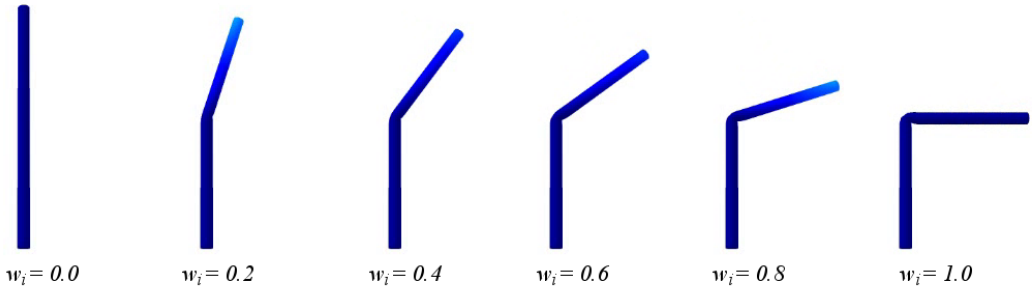
Given the non-linear mesh deformation $M_{NL}(\mathbf{w})$ (Equation 3.3) and linear approximation $M_L(\mathbf{w})$ (Equation 3.2) we can evaluate a displacement field

$$D_{NL}(\mathbf{w}) = M_{NL}(\mathbf{w}) - M_L(\mathbf{w}) \tag{3.6}$$

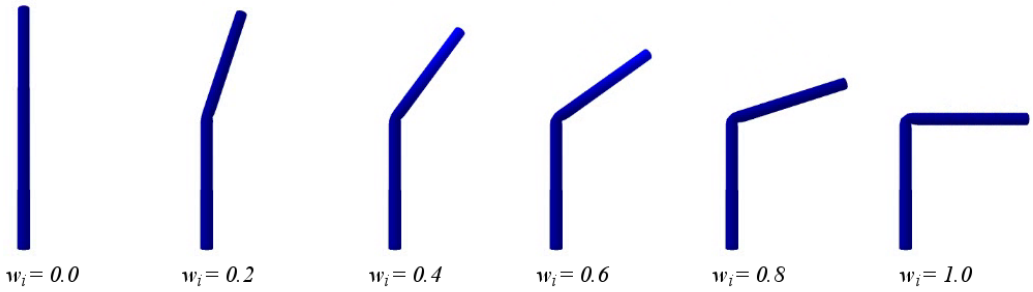
that computes the difference in shape between the linear and the non-linear approach.

The exact non-linear deformation for blend weights $\mathbf{w}$ can then be recovered by linear interpolation together with a non-linear correction:

$$M_{NL}(\mathbf{w}) = M_L(\mathbf{w}) + D_{NL}(\mathbf{w}). \tag{3.7}$$



(a) *Results achieved using the proposed hybrid piece-wise linear approach with a vector of pre-computed weights* $\mathbf{r} = \{0.5\}$. *Notice how the upper part of the tube is coloured in light blue, showing small geometric dissimilarities with respect to non-linear results.*



(b) *Results achieved using the proposed hybrid piece-wise linear approach with a reference vector of precomputed weights* $\mathbf{r} = \{0.25, 0.5, 0.75\}$. *Blended meshes are almost equivalent to the results achieved with a non-linear approach.*

Figure 3.6: *Left and right: input pair of meshes. Centre: blended meshes computed using the proposed hybrid piecewise linear approach for blending weights 0.2, 0.4, 0.6 and 0.8.*

An advantage of storing the displacement field D_{NL} is that for blending between mesh sequences of similar motions linear blending gives a reasonable approximation for large parts of the surface $D_{NL} \approx 0$ allowing efficient compression whilst accurately reproducing regions of significant non-linear deformation. An evaluation of the compression achieved with respect the area corrected with the non-linear approach is presented in Section 3.4.

To accurately approximate the non-linear deformation for blending a set of N source meshes $\mathbf{M}$ with arbitrary weights $\mathbf{w}$ we pre-compute the non-linear displacement field $D_{NL}(\mathbf{r})$ at a discrete set of intermediate weight values $\{r_i\}_{i=1}^{R}$ to give an additional set of M_{NL} reference meshes for interpolation. Real-time online interpolation is then

performed using a linear vertex blending with the non-linear correction:

$$M(\mathbf{w}) = \sum_{j=1}^{N+N_{NL}} g(\mathbf{w}, \mathbf{r}_j)(M_L(\mathbf{r}_j) + D_{NL}(\mathbf{r}_j)) \tag{3.8}$$

where $g(\mathbf{w}, \mathbf{r}_j)$ is a weight function giving a linear blend of the nearest reference meshes and zero for all other meshes. Equation 3.8 gives an exact solution at the original and non-linear interpolated reference meshes, and an approximate interpolation of the nearest reference meshes elsewhere. A recursive bisection of the weight space $\mathbf{w}$ is performed to evaluate a set of non-linearly interpolated source meshes such that for all $\mathbf{w}$ the approximation error $(M_{NL}(\mathbf{w}) - M(\mathbf{w})) < \epsilon$. Typically for interpolation of mesh sequences representing related motions only a single subdivision is required.

Figure 3.6 depicts an example of the results for a range of weights w_i achieved using the propose hybrid blending approach, using the same input meshes used in Figures 3.3 and 3.5 to illustrate the linear and non-linear methods introduced earlier in this chapter. Meshes are coloured using a heatmap to highlight differences in the geometry of the blended result with respect the non-linear approach. Figure 3.6a uses a reference weight vector $\mathbf{r} = \{0.5\}$, thus precomputing just a single subdivision of displacement fields. Notice how the error significantly decreases with respect the linear approach illustrate in Figure 3.3. Results in Figure 3.6b are generated using a reference vector of weights $\mathbf{r} = \{0.25, 0.5, 0.75\}$, thus 3 intermediate displacement fields were precomputed. The error with respect to the non-linear approach hardly noticeable.

3.4 Evaluation of Hybrid Non-Linear Blending

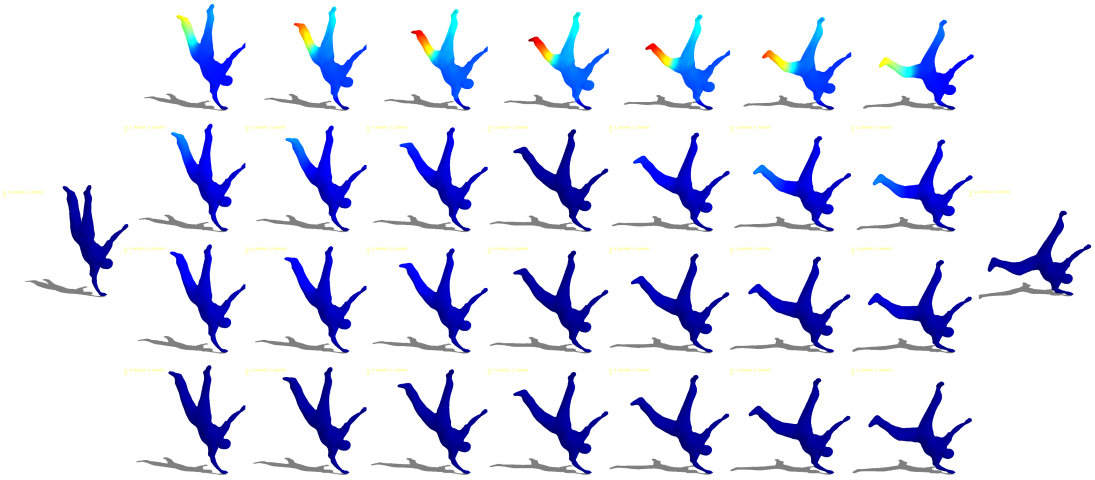
Figures 3.7a and 3.7b present a comparison of errors for linear blending with the proposed hybrid non-linear approach with one and three interpolated reference meshes. Meshes are coloured using a heatmap to highlight errors in geometry with respect the non-linear result. A more challenging blend is shown in Figure 3.8, where the opaque poses shown in 3.9d are interpolated. The linear method, shown in the centre-left column, clearly fails in maintaining the original volume of the mesh, resulting in sever unnatural deformations, specially for the limbs and head. On the other hand, the non-linear approach shown in the left-column, does not suffer from unnatural deformations.

Both the original volume and length of the limbs are maintained, enabling synthesis of novel poses that maintain the realism of the captured data. Centre-right and right columns are the results obtained by the proposed hybrid approach, coloured using a heat-map to highlight differences with respect to the non-linear approach. Notice how arms and head maintain realistic pose and proportions.

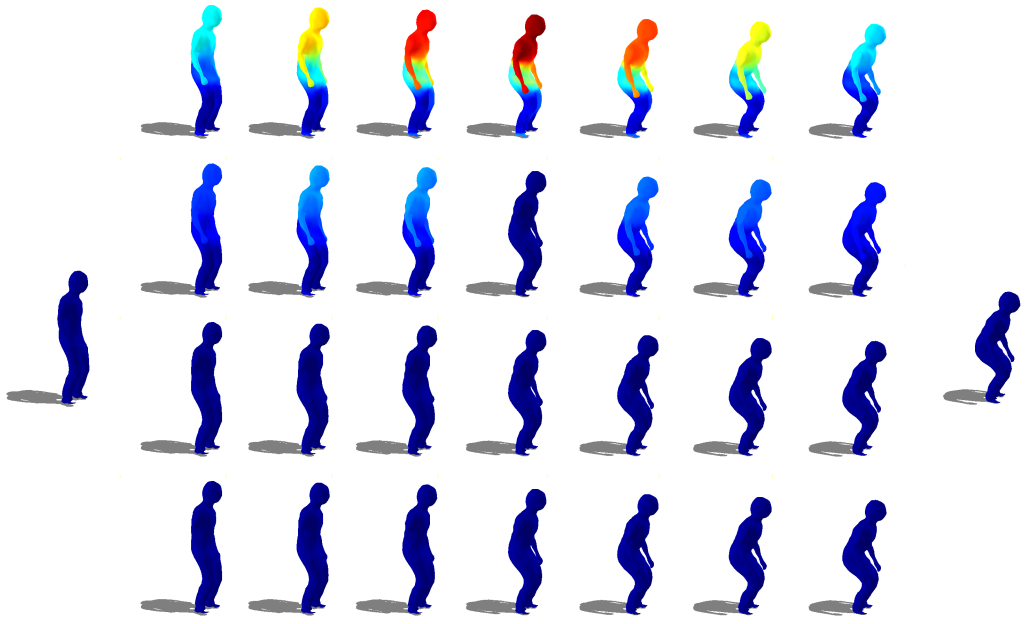
Figure 3.9 illustrates the geometric errors of the proposed hybrid blending approaches used in Figure 3.8, evaluating 3 different error metrics: average vertex displacement error, maximum vertex displacement error and RMS. Linear interpolation suffers from undesired vertex displacement, resulting in large difference to the non-rigid approach, consequently generating unnatural deformations. The proposed approaches for hybrid mesh blending significantly reduces such error using a single vector of precomputed offsets ($\mathbf{r} = \{0.5\}$, in blue). Using three precomputed displacement vectors ($\mathbf{r} = \{0.25, 0.5, 0.75\}$, in red) our approach achieves results with hardly any visual difference with respect the non-linear approach.

Table 3.1 presents quantitative results for error and CPU-time of figures 3.7a, 3.7b and 3.8. A number error measures have been compared, including RMS, absolute maximum vertex displacement, and displacement error as a percentage of model size. This evaluation demonstrate that the proposed real-time hybrid non-linear mesh blending approach achieves accurate approximation even with a single intermediate non-linear displacement map whereas linear blending results in large errors.

The influence of the threshold ϵ in the final result is shown in Figures 3.10a and 3.11a. In the former, the walk and jog poses located in the top row, in green, are linearly blended resulting in a mesh $M_L(\mathbf{w}), \mathbf{w} = \{0.5\}$, shown in the centre of the top row. Bottom row shows the result of the propose hybrid blending approach for different values of ϵ. For threshold values larger than 6 mm, only the areas with large errors (i.e. the limbs) are corrected, while the remaining of the mesh remains identical to the results of the linear approach. If the threshold ϵ is set to lower values, more regions of the linearly blended mesh will be corrected, although this might be unnecessary because they do not produce visual artefacts. This in an important observation that allows us to decrease the size of D_{NL} without compromising on perceived visual quality, as shown



(a) *Result of blending two poses of a street dancer using linear (top row), hybrid with one and three reference meshes (2nd/3rd row) and non-linear (bottom row). Top row shows that linear blending results in large errors (red) for the left leg which are corrected with the hybrid approach.*



(b)

Figure 3.7: *Result of blending two equivalent poses of the low jump (left) and high jump (right) sequences. 2nd and 3rd rows show that our proposed hybrid approach, with one and three references meshes respectively, gives an approximation to the non-linear blending whilst the top row shows large errors with linear blending.*

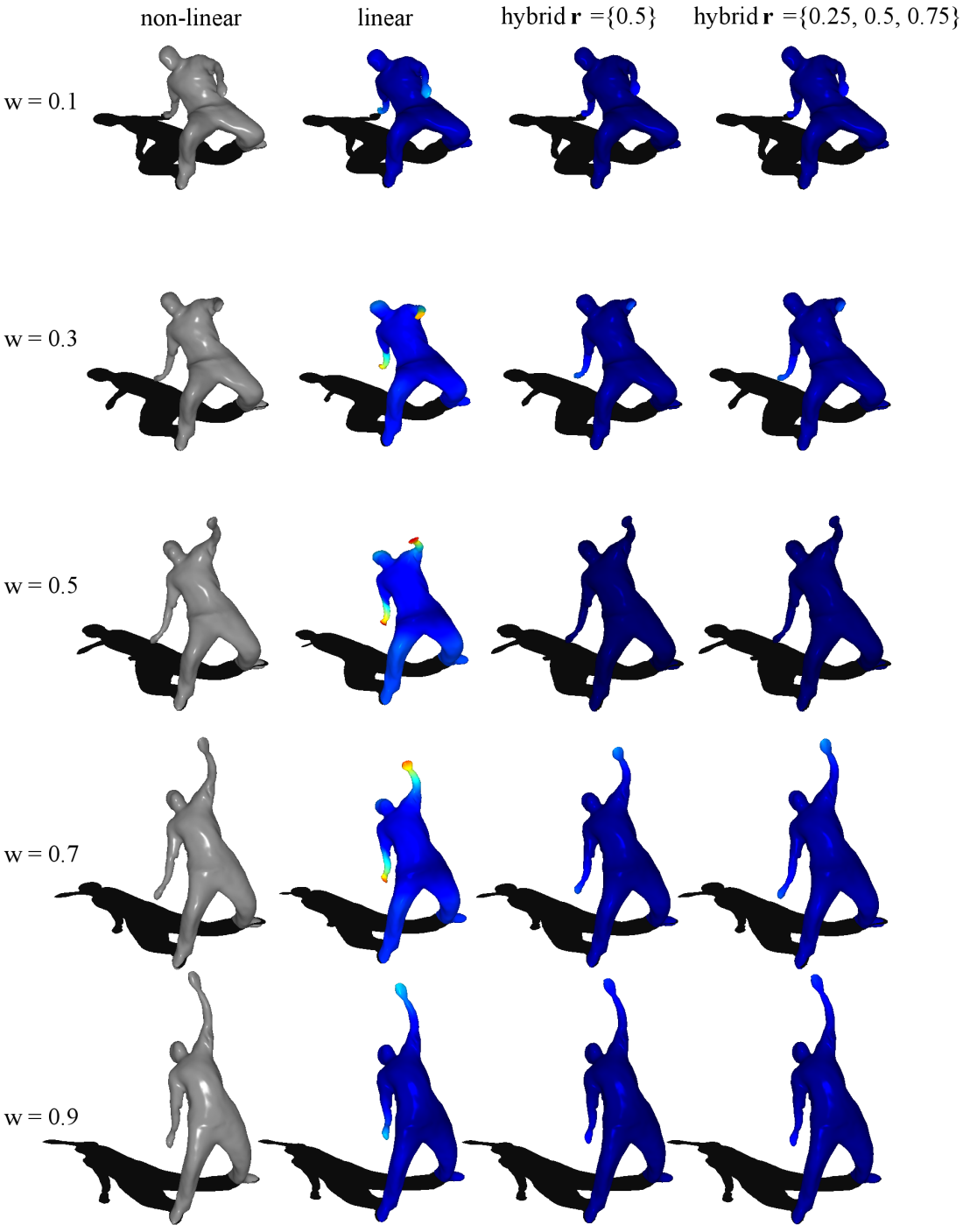
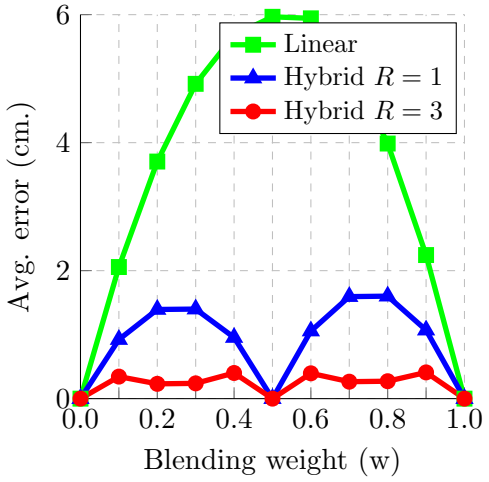
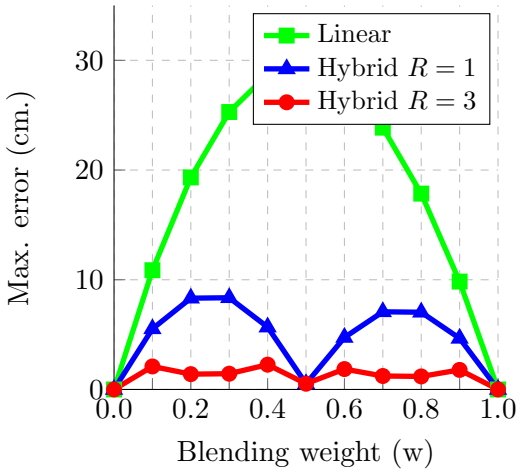


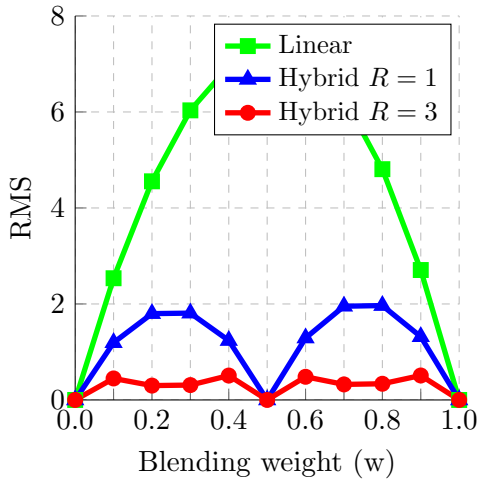
Figure 3.8: *Results of blending the two opaque grey poses shown in Figure 3.9d. From left to right columns: non-linear approach, linear, hybrid with* $\mathbf{r} = \{0.5\}$ *and hybrid with* $\mathbf{r} = \{0.25, 0.5, 0.75\}$*. Quantitative evaluation plotted Figure in 3.9 and shown in Table 3.1.*



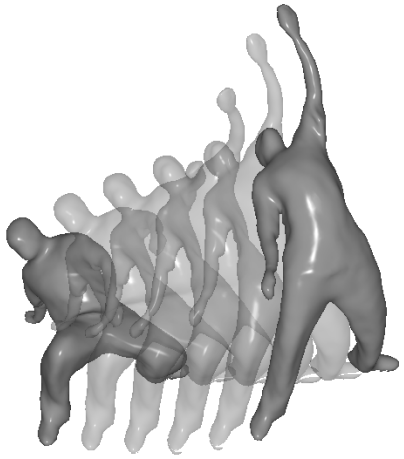
(a) *Average vertex displacement error with respect the non-linear solution for blending weights* $w_i \in [0..1]$



(b) *Maximum vertex displacement error with respect the non-linear solution for blending weights* $w_i \in [0..1]$.



(c) *RMS error with respect the non-linear solution for blending weights* $w_i \in [0..1]$.



(d) *In opaque grey: two input shapes to be interpolated. In-between semitransparent shapes are the non-linear results. For detailed results, see Figure 3.8.*

Figure 3.9: *Visualisation of the errors associated with each of the blending approaches discussed in this section for interpolation of the two opaque shapes shown in Figure 3.9d. For comparison purposes, three different error metrics (average vertex error, maximum vertex error and RMS) are shown. Actual resulting blended meshes are shown in Figure 3.8. Sample rate* $\Delta w = 0.1$.

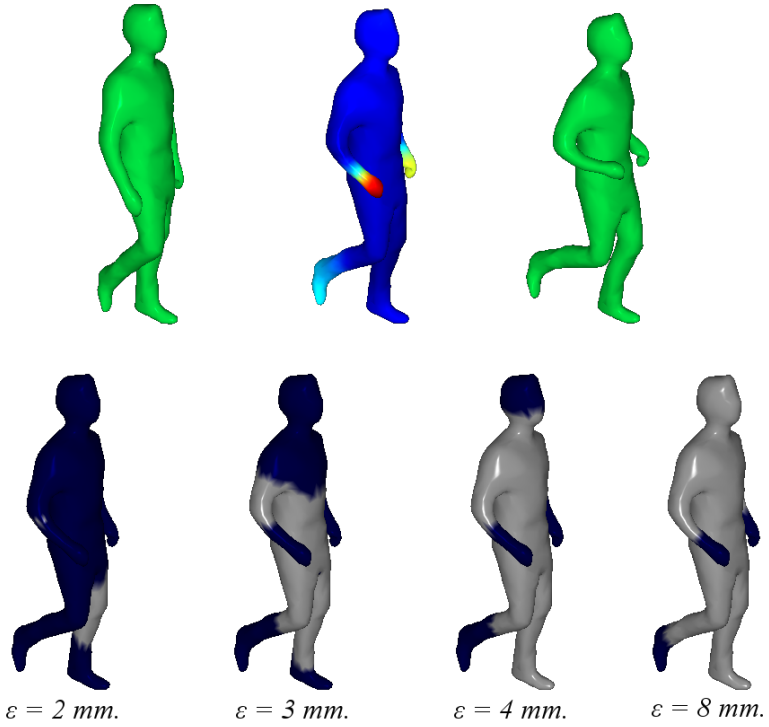
Sequence	#vertices	Method	Max. Error %	Max. Error cm.	Max. RMS	Time sec / frame
Fig. 3.7a	5,580	Linear	14.38 %	23	7.01	0.008
		Hybrid $R = 1$	3.67 %	6	1.63	0.015
		Hybrid $R = 3$	1.60 %	2	0.4	0.017
		Non-linear	0.00 %	0	0.00	0.749
Fig. 3.7b	3,000	Linear	9.14 %	15	5.32	0.004
		Hybrid $R = 1$	1.34 %	2	0.67	0.014
		Hybrid $R = 3$	0.93 %	1	0.42	0.016
		Non-linear	0.00 %	0	0.00	0.789
Fig. 3.8	5,580	Linear	17.34 %	29	7.29	0.004
		Hybrid $R = 1$	5.06 %	8	1.97	0.014
		Hybrid $R = 3$	1.26 %	1	0.51	0.016
		Non-linear	0.00 %	0	0.00	0.789

Table 3.1: *Maximum vertex displacement error with respect to non-linear blending as a percentage of model size for in meshes in Figure 3.7.*

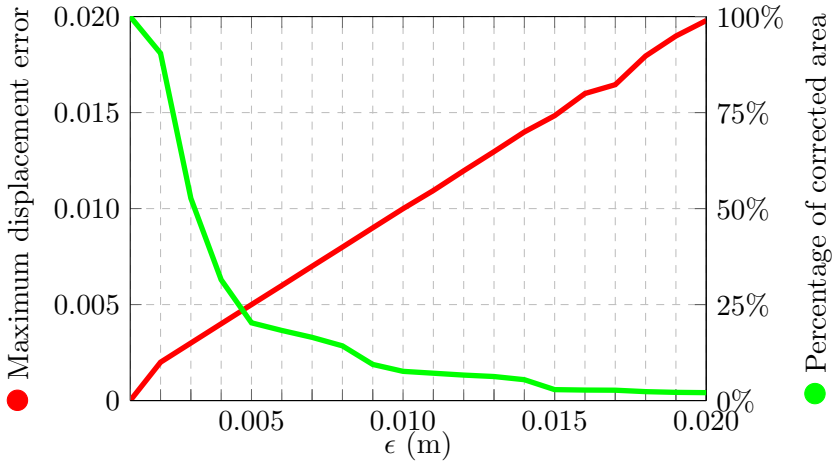
in Figure 3.12. In order to better evaluate the influence of the threshold ϵ, a more challenging example is shown in Figure 3.11a, where two significantly different poses in shape, taken from the J.P., are blended. Again, linear blend fails dramatically, resulting in up to 20 $cm.$ vertex displacement errors. Furthermore, a quantitative evaluation is presented in Figures 3.10b and 3.11b, showing how the percentage of the area of the mesh which is corrected depends on ϵ, and its influence on the final displacement error.

Figure 3.12 characterises the representation error and storage cost against the number of subdivisions for different error thresholds ϵ of the blend in Figure 3.10a. A relatively small error reduction for thresholds below $5mm$ is present, while the memory usage increases significantly. This is caused by the $5mm$ resolution of the original database, since details below this level are not reconstructed.

This evaluation demonstrates that the proposed hybrid non-linear mesh blending allows accurate approximation of non-linear mesh deformation whilst maintaining the computational performance of linear blending to allow real-time interactive animation.

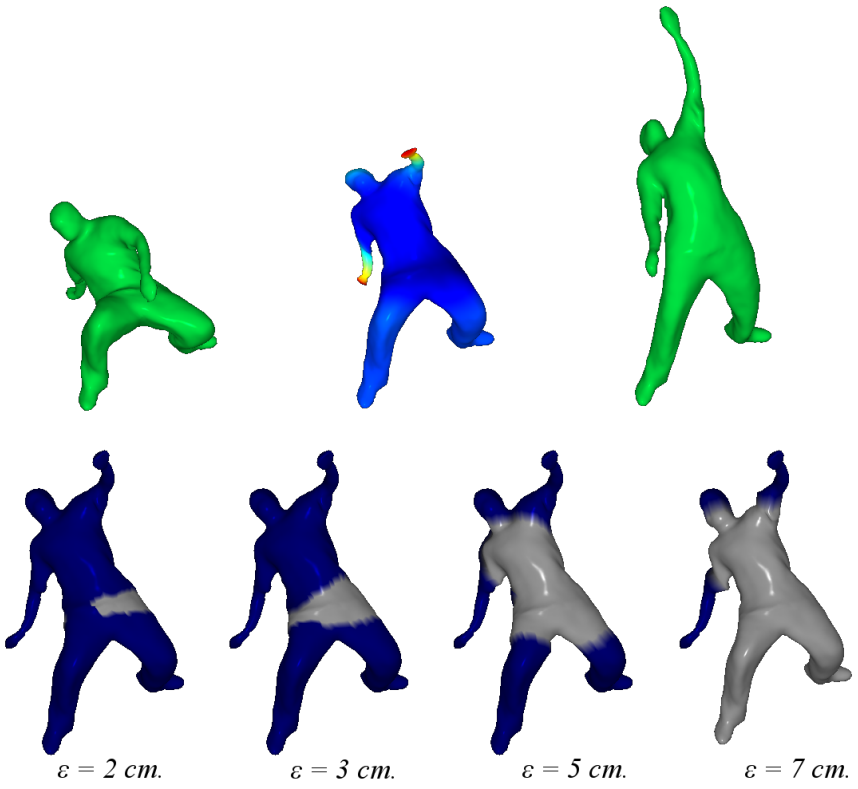


(a) *Top row: left and right, two equivalent poses of walk and run sequences. In the middle the resulting linear blended mesh, coloured using a heat-map (red largest error) to display the errors with respect to the non-linear result. Bottom row: results of our hybrid approach using different threshold ϵ (2, 3, 4 and 8 mm.). Grey represent areas with errors below the threshold (non corrected), and blue represents areas above the threshold that have been corrected.*

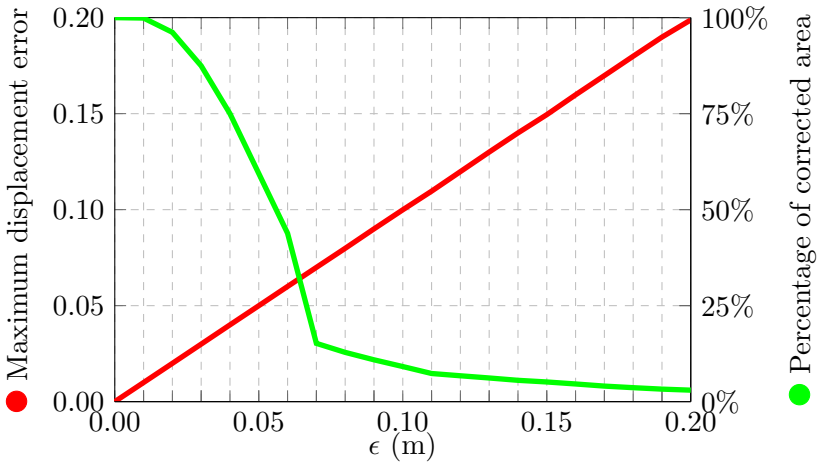


(b) *Maximum displacement error (red) and percentage of the mesh area corrected (green) for the pose interpolation shown in Figure 3.10a for a number of values of ϵ.*

Figure 3.10: *Qualitative and quantitative evaluation of the influence of parameter ϵ in the hybrid blending approach presented in this section.*

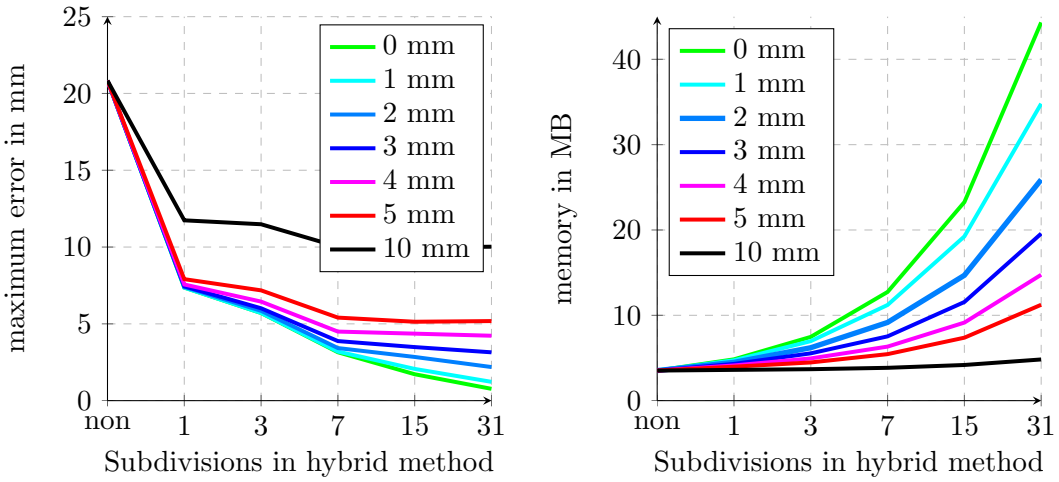


(a) *Top row: left and right, two poses from J.P. dataset. In the middle the resulting linear blended mesh, coloured using a heat-map (red largest error) to display the errors with respect to the non-linear result. Bottom row: results of our hybrid approach using different threshold ϵ (2, 3, 5 and 7 cm.). Grey represent areas with errors below the threshold (non corrected), and blue represents areas above the threshold that have been corrected.*



(b) *Maximum displacement error (red) and percentage of the mesh area corrected (green) for the pose interpolation shown in Figure 3.11a for a number of values of ϵ.*

Figure 3.11: *Qualitative and quantitative evaluation of the influence of parameter ϵ in the hybrid blending approach presented in this section.*



(a) *Maximum displacement error, depending on the number of subdivisions and threshold ϵ*

(b) *Memory in MB of the matrix D_{NL}, depending on number of subdivisions and threshold ϵ*

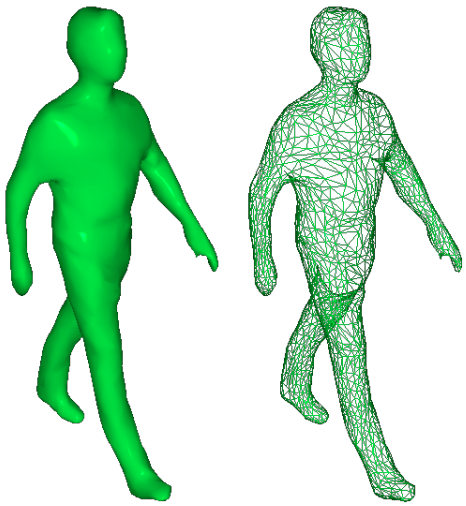
Figure 3.12: *Evaluation of the maximum error and memory usage of the hybrid blending method, for the walk-run parameterised motion.*

3.5 Results

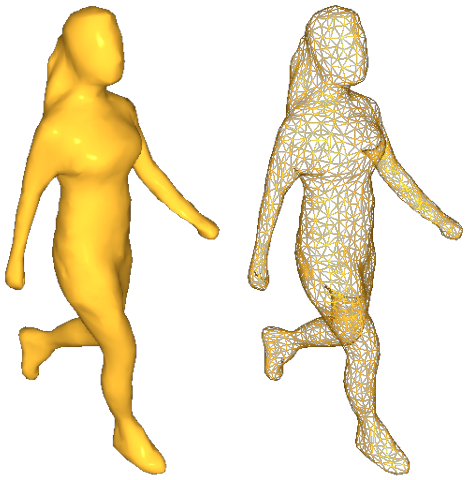
3.5.1 Data

Datasets used in this work are reconstructed from multiple view video capture of actor performance in two different studio setups, as described in Section 2.2. In *Studio #1*, 8 HD cameras equally spaced in a circle were used, forming a capture volume $5m^2 \times 2m$. This studio was used to capture characters *Dan* and *Roxanne*, presented in figures 3.13a and 3.13b. A second studio, *Studio #2*, formed by 10 HD cameras, was used to capture characters *Infantry* and *Knight*, presented in figures 3.13c and 3.13d.

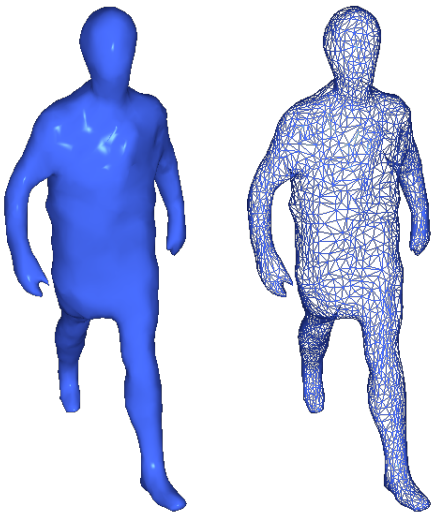
Reconstruction is performed using multi-view stereo [SH07b] followed by per-character temporal alignment [HBH11, BHKH13] of all frames to have a consistent mesh structure, see Section 2.2.1 for more details. Throughout this work we use a single intermediate mesh for hybrid non-linear interpolation which gives a an accurate approximation, as shown in Figure 3.7.



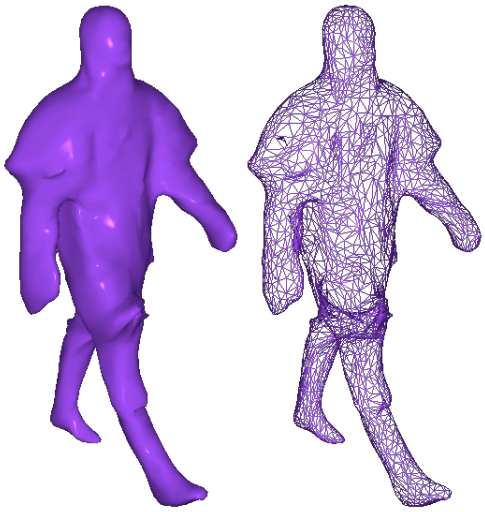
(a) *'Dan' character. 2667 vertices and 5330 triangles.*



(b) *'Roxanne' character. 2886 vertices and 5772 triangles.*



(c) *'Infantry' character. 4052 vertices and 8100 triangles.*



(d) *'Knight' character. 4058 vertices and 8120 triangles.*

Figure 3.13: *Samples of the 4 different character datasets used in this work. Each subfigure shows a an opaque render of an arbitrary pose of the dataset and its wireframe.*

3.5.2 Parametric Animations

Figure 3.14 shows parametrised motion spaces for walking, jumping and reaching motions constructed from pairs of mesh sequences for *Dan* character. Rendered meshes are coloured to show the parameter change. Figure 3.15 presents results for the *Infantry* character. Figure 3.16 presents equivalent results for character *Knight*.

Figures 3.17 and 3.18 present a multi-parameter character animation constructed from four mesh sequences from dataset *Dan*, with walking speed and direction control, in which the presented hybrid blending method with 1 reference runs at 0.020 $secs/frame$ using 4 input sequences and 3 blending weights, with a maximum displacement error of 0.73% with respect to the non-linear approach. Figures 3.19 and 3.20 present similar results using datasets *Infantry* and *Knight* respectively.

These results show that the proposed mesh sequence blending approach using the hybrid non-linear deformation achieves a natural transition between the captured motions. Four datasets of different actors performance, captured in two different studio setups, have been tested and evaluated. Tested datasets contain different levels of surface motion complexity, ranging from characters with tight clothes such as jeans and t-shirt as *Dan* to long hair and loose clothing as in *Knight* and *Roxanne*.

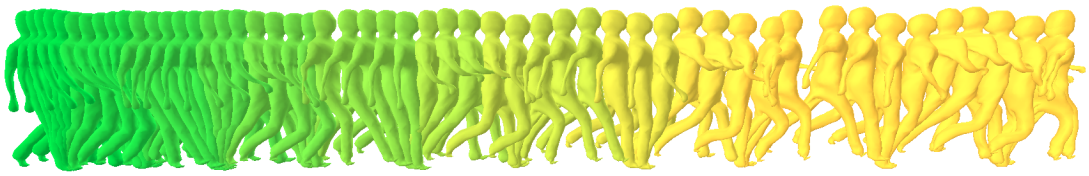
3.5.3 Discussion

Qualitative results demonstrate that the proposed approach successfully synthesises 3D mesh sequences combining similar captured motions, resulting in novel motions that maintain the realism of the reconstructed data. However, our scheme presents a number of limitations. The approach may fail to produce natural intermediate motions if the input motions are not related, for example a blend between a walk and a jump. Note that this limitation was also present in previous research on skeletal motion parametrisation. Nevertheless, to fully exploit the captured datasets of motions, methods of combining dissimilar motions are required.

Other limitations are related to the accuracy of the 4D data alignment, a requirement in the proposed approach. Errors in the mesh surface alignment may cause undesired

drift in the blended vertices position, resulting in visual surface artefacts. Highly non-rigid surface areas such as hair typically suffer from alignment errors, thus such areas may not produce plausible intermediate meshes.

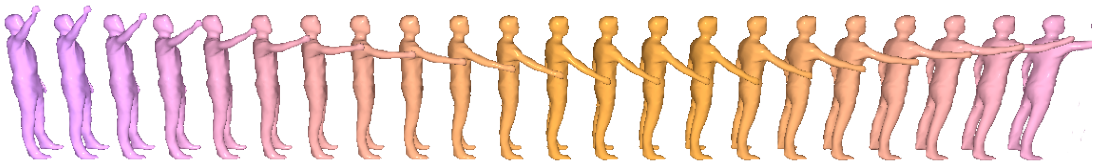
Limitations related to anatomical and physical facets of the character are also present. No anatomical constraints are imposed to the synthesised blends, thus these may result in unnatural poses. Furthermore, foot-floor and hand-object constraints are not enforced, exposing intermediate meshes to external contact artefacts. Those could be fixed imposing both space-time and mesh-IK [TH11] constraints.



(a) *Walk-to-run speed parametrisation, from walk (green) to run (yellow).*

(b) *Length of jump parametrised from short (red) to long (orange).*

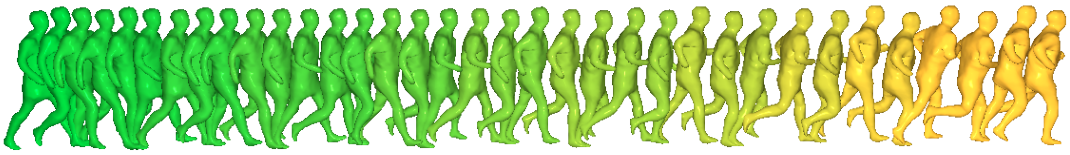
(c) *Height of jump parametrised from low (grey) to high (purple).*



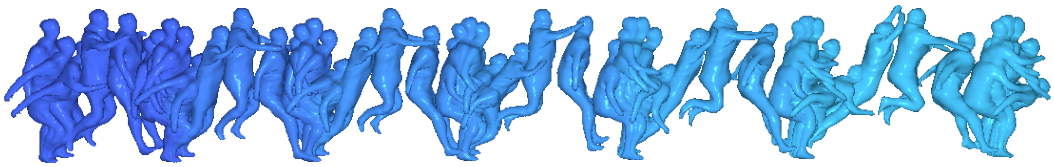
(d) *Parametrised reach action from high (light purple) to low (orange).*

(e) *Parametrised reach action from high (light purple) to low (orange).*

Figure 3.14: *Examples of parameterised motions between two motion sequences with continuous parameter variation (every 5^{th} frame). Results generated using dataset 'Dan'.*



(a) *Walk-to-run speed parametrisation, from walk (green) to run (yellow).*



(b) *Length of jump parametrised from short (dark blue) to long (light blue).*

(c) *Height of jump parametrised from low (purple) to high (pink).*

Figure 3.15: *Examples of parameterised motions between two motion sequences with continuous parameter variation (every 5^{th} frame). Results generated using dataset 'Infantry'.*

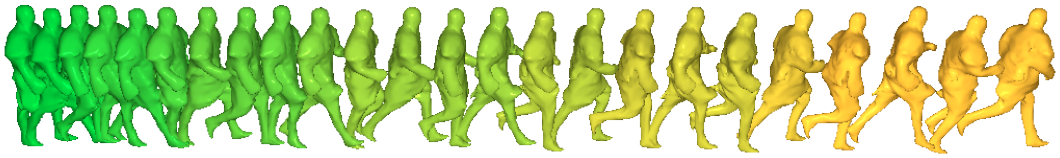


Figure 3.16: *Walk-to-run speed parametrisation, from walk (green) to run (yellow). Results generated using dataset 'Knight'*

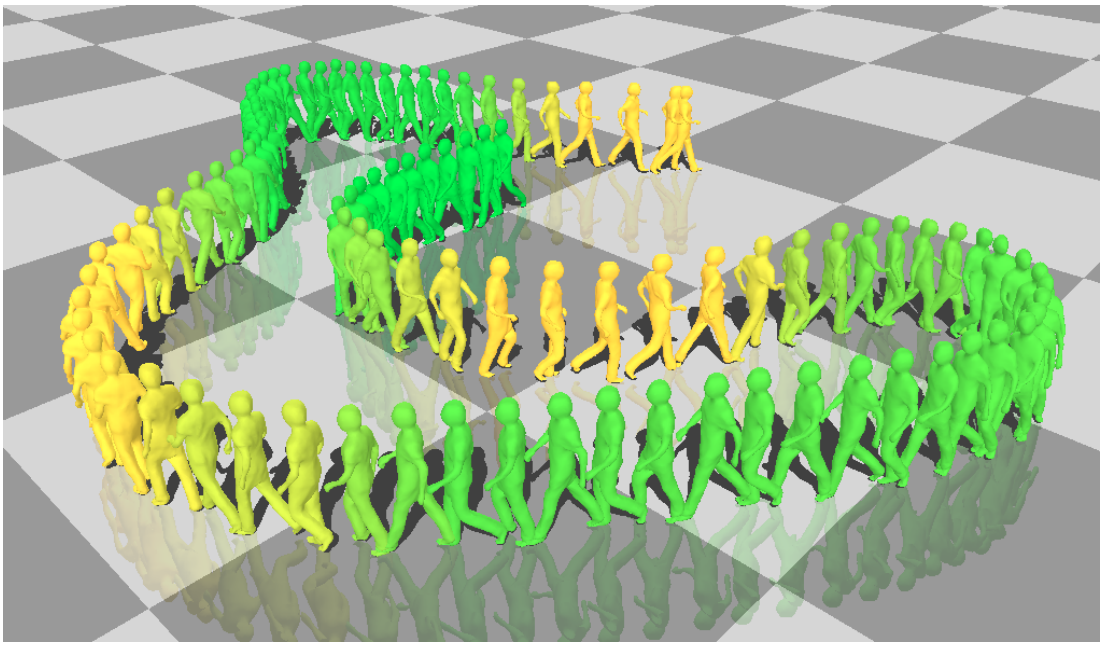


Figure 3.17: *Screenshot of the interactive scenario created to control a character in real time using the proposed approach. The user can interactivelly control a character built using dataset 'Dan', manipulating its speed and direction.*

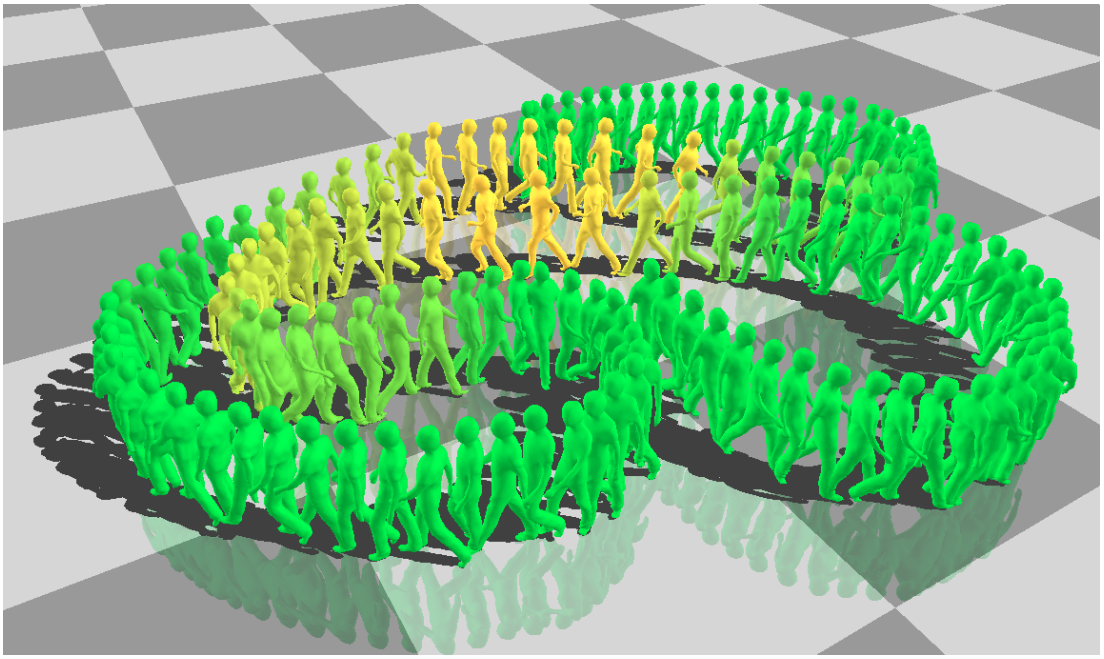


Figure 3.18: *Path interactively travelled by the user, using character 'Dan', controling speed and direction.*

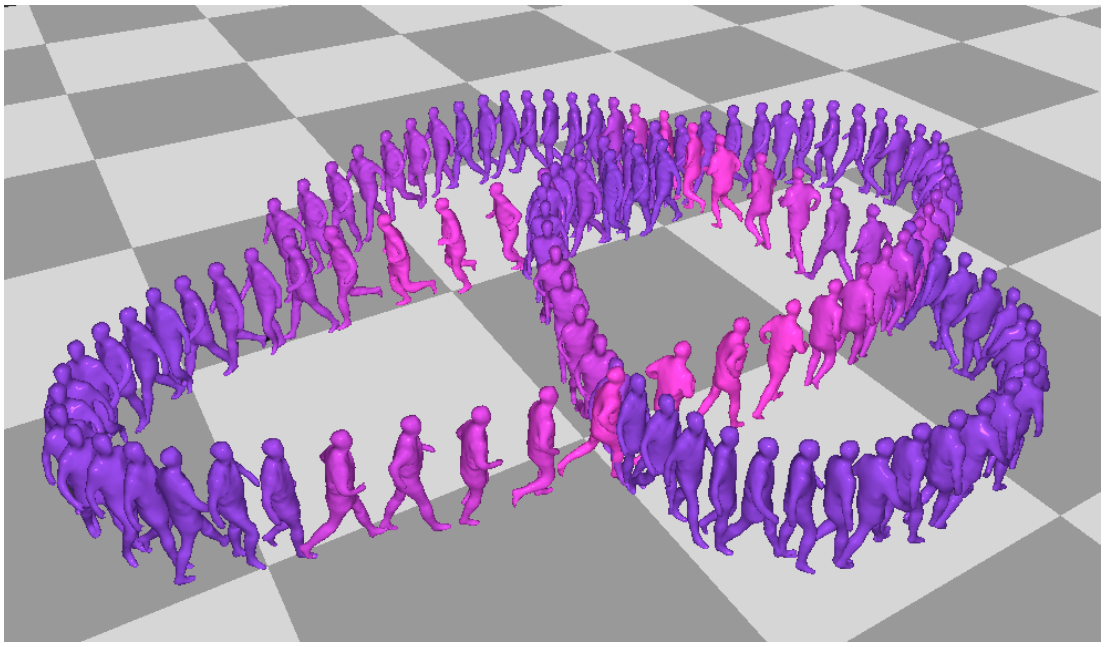


Figure 3.19: *Interactive character created using dataset 'Infantry'. User controls speed and direction.*

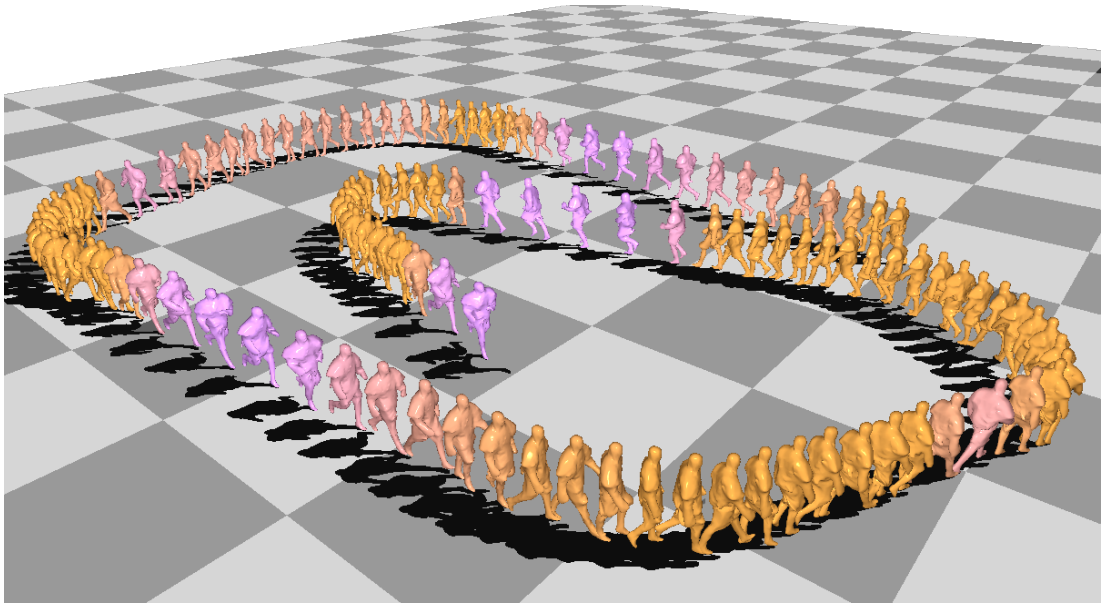


Figure 3.20: *Interactive character created using dataset 'Knight'. User controls speed and direction.*

Chapter 4

4D Parametric Motion Graphs

Chapter 3 showed how novel 3D-mesh sequences can be generated by the combination of captured 4D video sequences. However, the requirement for these sequences to be similar in motion restricts parametrisation to single motion classes. Artefacts such as mesh collapsing and shrinking, together with unrealistic deformations, may appear if the interpolated sequences are not from the same class of semantically related motions. Methods to overcome such limitation are required to fully exploit the datasets of 4D video sequences to synthesise novel motions with a greater range of actions.

Over the last decade, research with skeletal motion capture, MoCap, data has approached this problem by proposing methods based on finding similarities between the captured clips to concatenate different motions [MTH00, LCR*02, AF02, KGP02, SO06, HG07]. This allows the synthesis of a novel motion sequence as a result of seamlessly concatenating segments of captured data. A new data structure referred as *Motion Graph* was proposed to encapsulate links between clips of MoCap data [KGP02]. Our goal is to find analogous methods for 4D video data, to allow transitions between different classes of parametrised motion (for example walk to jump).

Introduction of motion graphs for 4D video is a challenging problem do to the relative complexity of the data compared to skeletal MoCap. In MoCap, similarity between motions can be computed from direct comparison of joint angle or positions [KGP02], in 4D video measurement of similarity based on surface shape is required. Furthermore,

there is a significant difference in data size, while MoCap is traditionally stored using a hierarchical skeletal model of < 50 joints, 4D video meshes consist of thousands of vertices.

This chapter introduces the *4D Parametric Motion Graphs*, 4DPMG, [CTGH12a, CTGH13], a new data structure that encapsulates different 4D video parametric spaces, described Chapter 3, and the links between them. The 4DPMG allows real-time interactive control of a 3D-mesh character created by the combination of multiple 4D video sequences of different motions.

4.1 Related Work

Two approaches have been traditionally used to animate virtual characters: key-frame animation, in which a highly skilled animator draws key poses of a character motion and then in-between poses are filled; and animation from captured skeletal motion data (MoCap). Key-frame animation is time consuming, requiring a highly skilled animator and may fail in produce motion detail present in real motion. Skeletal MoCap, achieved by mechanical sensors or marker-based technologies, provides a highly detailed rigid skeletal motions, and it have been widely used in both visual-effects and gaming industries [Gle99]. However, the motion is baked in and requires additional tools to adjust the character motion for artistic or scene constraints.

In order to exploit the captured MoCap data, in animation production over the last two decades there has been an increasing interest by the Computers Graphics community on finding methods for reusing and editing skeletal motion.

4.1.1 Reuse of Skeletal Motion Capture

The main goal is to exploit captured sequences to enable the synthesis of novel motions that maintain the realism of the source clips. Approaches can be divided into three groups: interpolation methods, previously discussed in Section 3.1.1, which aim to blend a set of captured sequences to generate novel motions; skeletal editing methods, which aim to modify a captured clip by manually editing a captured pose; and concatenation

methods, which aim to synthesise motions by linking a set of captured sequences. The later is discuss in detail in the remainder of this section.

Early work in piecing together motion clips was presented by Perlin and Goldberg [Per95, PG96], who used simple blends to link procedurally generated motions to synthesise coherent sequences. Lamouret and van de Pane [LvdP96] propose a technique for creating new animations from physically simulated motions stored in a database. However, their system was applied to a simple agent, the Luxo lamp, with only five degree of freedom. Molina-Tanco and Hilton [MTH00] present a two-level approach to author novel motions from a database of MoCap example. A Markov chain of joint trajectories is built in the first level, enabling the generation of the overall motion path. The second level of the model matches the states of the Markov chain with actual segments of the captured motions, resulting in a novel synthesised realistic motion.

In the early 1990s, online motion generation was also investigated in the game industry. *Move trees* [MBC01] are graph structures that represent connections in a database of motions. However, graph edges, representing links between motion clips, are manually found. Further research on graphs representing links between captured sequence focused on relaxing these manual requirements, leading the algorithms for automatic computation of seamless motion links.

Lee *et al.* [LCR*02] achieve real-time control of three-dimensional avatars by preprocessing plausible transitions between motion segments, and clustering clips for efficient motion search at run-time. Kovar *et al.* [KGP02] introduce the so-called *Motion Graph*, consisting in a graph-like structure that encapsulates not only original motion clips, but also connections between them. Novel motions can be created by simply building walks on the graph. Arikan and Forsyth [AF02] presented a framework that generates human motions by cutting and pasting motion capture data, based on randomised search of a hierarchy of graphs. This approach can generate motion sequences that automatically satisfy a set of user constraints, including multiple character interaction. Pullen and Bregler [PB02] keyframe a selected set of the character's degrees of freedom and find matches between these and lower frequency bands of motion data. As a result, synthetic sequences are generated by short clip concatenation. Arikan *et al.* [AFO03]

presented an intuitive framework for motion synthesis based on user annotations (i.e: walk, run and then jump). The MoCap database needs to be annotated offline, and a Support Vector Machine (SVM) classifiers is used to generalise the user annotations to the entire database. Final motion is constructed by cutting pieces of motions from the database and assembling them together. Ikemoto and Forsyth [IF04] propose a method to significantly increase a collection of MoCap sequences by cutting limbs from one motion sequence and attaching them to another, referred as *transplantation.* They describe a methodology to evaluate the realism of the results, which can also be applied for general motion synthesis assessment.

Further research [ZS08] improves on traditional motion graph connectivity by initially creating a new set of interpolated motions from the original captured sequences, expanding the dataset. This allows the construction of a *well-connected motion graph*, with smoother transitions. Heck and Gleicher [HG07] introduce a new data structure so-called *Parametric Motion Graphs* to describe valid ways of generating linear blend transitions between motion clips dynamically generated through parametric synthesis in real-time. Their main contribution is to find connectivities not between individual motions but to groups of similar motions.

Due to the growing complexity of motion graphs [SO06, HG07, ZS08], recent research in MoCap animation also focuses on developing highly structured representation for large unstructured motion capture datasets. The *Motion-Motif Graphs* [BCvdPP08] use motion segmentation and clustering techniques in unstructured motion data to automatically build compressed graphs from large datasets of motions with more than 100,000 frames. More recently, *Motion Graphs++* [MC12] introduced a new highly structured generative statistical model for both motion analysis and synthesis. The approach is capable of generating infinite variations within the same action, and it is demonstrated to work in a database containing two hours of MoCap data and more that 15 different actions.

4.1.2 2D Video-Based Animation

Approaches for more general computer generated animation (i.e: not restricted to character animation) based on traditional 2D video footage have also been recently explored. *Video Rewrite* [BCS97] uses existing clips of a person talking to synthesise novel videos of a person mouthing words never pronounced before. Phonemes are automatically detected in the original input and the corresponding video segments are stitched back together to generate the desired output. *Video Textures* [SSE00] introduce a method for video clip evaluation to extract its structure, and for synthesising a new, similar looking video of arbitrary length. Among other applications, *Video Textures* allows 2D interactive control of video-based animation. *Human Video Textures* [FNZ*09] improve previous motion concatenation techniques by simultaneously capturing maker-based data and video data. Markers provide 3D spatial information that combined with 2D information from video footage allow the computation of seamless transition between original video clips. More recently, Hilsmann *et al.* [HFE13] introduce a *pose space image based rendering* demonstrating photorealistic animation of clothing from a set of 2D images augmented with 3D shape information to support natural transitions.

4.1.3 Reuse of 4D Performance Capture

Inspired by previous research in concatenation of MoCap sequences [KGP02, AF02, LCR*02], and taking advantage of recent improvements in 4D performance capture discussed in Section 2.2.1, methods to create interactive 3D-mesh characters from 4D performance capture have recently appeared. Analogous to Motion Graphs [KGP02] for MoCap data, the goal is to generate novel 4D video sequences by the concatenation of segments of captured sequences.

Starck *et al.* [SMH05] introduce a video-based representation for free-viewpoint visualisation and concatenate animation from captured 3D video sequences. Their approach uses a spherical matching algorithm to derive surface correspondence from global surface information, allowing the computation of seamless transitions between original 3D performance capture clips. Animation control is achieved using a motion graph structure of geometry, with transitions manually identified. Huang *et al.* [HHS09] in-

troduce the so-called *Surface Motion Graphs*, enabling the synthesis of novel 3D-video sequences concatenating segments of 3D performance capture according to user constraints on movement, position and timing. The resulting motion is created by finding the optimal path in the *Surface Motion Graph* that satisfies user constraints. Shape similarity [HHS10a] are used to identify transitions between input sequences, which does not require a consistent mesh topology over time.

More recently, example-based approaches through resampling multi-view video sequences have been extended to body motion [XLS*11] allowing offline animation via key-frame or skeletal motion. These approaches preserve the realism of the captured sequences in the final render, but are not suitable for interactive real-time character control.

4.2 Introducing 4D Parametric Motion Graphs

This chapter introduces the *4D Parametric Motion Graph*, 4DPMG, a graph representation for interactive animation from a database of 4D video sequences. Given a 4D performance database of mesh sequences for different movements with a consistent mesh structure at every frame, referred to as 4D video, we first achieve parametric control by combining multiple sequences of related motions, as discussed in Chapter 3. This gives a parametrised motion space controlled by high-level parameters (for example walk speed/direction or jump height/length). However, parametric motions can only be synthesised by combining semantically similar sequences (i.e: walk and run), whereas the interpolation of non-similar sequences, such as jump and walk, would fail in generating a human-realistic motion. Therefore, in order to fully exploit a dataset of 4D video sequences, methods for linking motions performing different actions are required, enabling the generation of a richer range of actions.

For both motion parametrisation and motion transition of 4D video sequences it is necessary to introduce techniques which preserve the surface motion whilst allowing real-time interaction. The 4DPMG representation allows interactive real-time control of character animation from a database of captured mesh sequences.

The 4DPMG represents the set of character motions and transitions which can be controlled interactively at run-time from a 4D performance capture database. Parametrised sets of motions form the graph nodes each representing a distinct set of motions which the character can perform with associated user-controlled animation parameters. The problem is then to define the graph edges which allow smooth transition between motions at run-time whilst maintaining real-time interactive control of the character motion. The parameter space for each node is continuous and we cannot therefore pre-compute all possible optimal transitions between parametrised motions. 4DPMG introduces a real-time approach to evaluate the optimal motion transitions with low-latency. Figure 4.1 shows a simple 4DPMG with nodes for four motions: walk with parameters for speed and direction; long-jump with parameters for length; jump-up with parameters for height; and reach with parameters for hand position. The arrows between nodes indicate possible transitions and the arrows to the same node indicate loops for cyclic motion.

4.3 4DPMG Nodes: Parametric Motion Spaces

As illustrated in Figure 4.1, nodes of a 4DPMG are created by the combination of similar motions. Each node can be considered as an independent parametric motion space created as described in Chapter 3. This particular example depicts a 4-node parametric motion graph with speed and direction control in node 1; reach hand control in node 2; jump height control in node 3; and length of jump control in node 4.

The hybrid piece-wise linear approach for 4D video sequence interpolation introduced in Section 3.3 is used provide real-time control of movement within each node, with intuitive high-level parameters such as speed and direction for walking or height and distance for jumping.

4.4 4DPMG Edges: Parametric Motion Transitions

Transitions between parametrised spaces for different motions in the 4DPMG are required to allow interactive character animation with multiple motions. Natural tran-

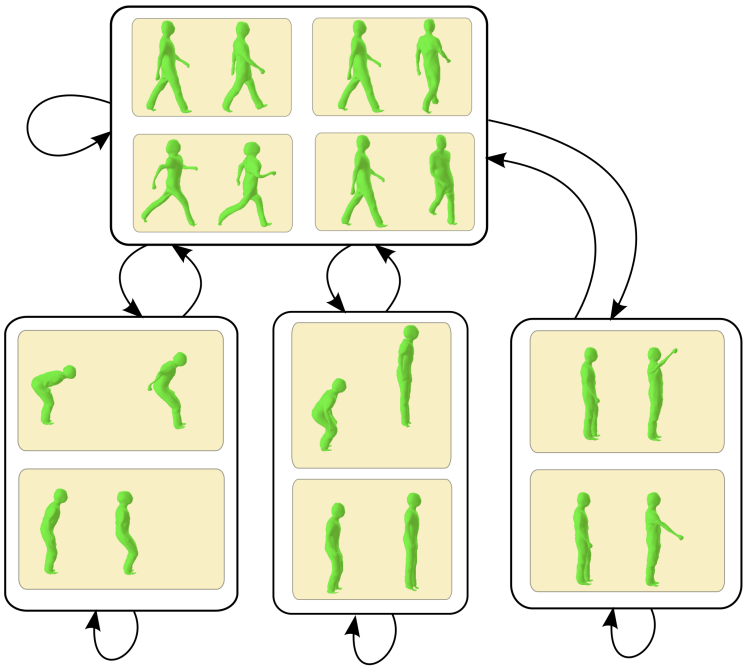


Figure 4.1: *Illustration of a 4D Parametric Motion Graphs showing four nodes with parametrised motion spaces: walk (top) parametrised for speed and direction; long-jump (bottom-left) parametrised for length; jump-up (bottom-middle) parametrised for hight; and reach (botom-right) parametrised for hand location.*

sitions require a similar shape and non-rigid motion in both spaces. In addition, for responsive interaction in real-time character control it is important to ensure low latency between the user input and motion transition.

Parametric transitions have been previously investigated by Shin and Oh [SO06], proposing the *Fat Graphs* for skeletal MoCap data. Groups of transition edges are used to create a *fat edge*, which parametrises similar motion segments into a blendable form. Similarly, Heck and Gleicher presented the *parametric motion graphs* [HG07], a novel representation for parametric skeletal motion control. A discrete set of good transitions is precomputed by evaluating the similarity in pose and motion between pairs of source and target points in the parametric space. To limit the memory required a fixed number of good transitions are stored and interpolated at run-time. However, precomputation of a fixed set of transitions can result in a relatively high latency due to the delay between the current state and next pre-computed good transition.

In this work we introduce an alternative approach which does not require the pre-computation of a discrete set of transitions. Optimal transitions are computed at run-time based on the current state in the source space and desired state in the target motion space. Our approach presents two main advantages over previous techniques for parametric motion control: transitions are not limited to a pre-computed fixed set allowing the best transition for a given starting point to be evaluated at run-time; and transition points can be evaluated on the fly to minimise latency whilst ensuring a smooth transition.

4.4.1 Finding Transition Candidates

A parametrised motion is defined as a mesh sequence

$$M^r(t^r, \mathbf{w}^r), \tag{4.1}$$

where $\mathbf{w}^r \in \Re^{N^r}$ is the set of user controlled parameters for the r^{th} motion class and $t^r \in [0..1]$ is the normalised frame time. Each parametrised motion $M^r(t^r, \mathbf{w}^r)$ is computed from a set of 4D mesh sequences $\{M_i^r(t)\}_{i=1}^{N^r}$ according to Equation 3.8. Given a current source state $M^s(t^s, \mathbf{w}^s)$ and a target state $M^t(t^d, \mathbf{w}^d)$ the problem is then to find the optimal transition path at run-time.

To evaluate the best transition path P_{opt} online we optimise a cost function representing the trade-off between similarity in mesh shape and motion at transition, $E_S(P)$, and the latency, $E_L(P)$, or delay in transition for a path P between the source state $M^s(t^s, \mathbf{w}^s)$ and target state $M^d(t^d, \mathbf{w}^d)$:

$$P_{opt} = \arg\min_{P \in \Omega} (E_S(P) + \lambda E_L(P)) \tag{4.2}$$

where λ defines the trade-off between transition similarity and latency ($\lambda = 5$ throughout this work). The transition path P is optimised over a trellis of frames starting at the current frame $M^s(t^s, \mathbf{w}^s)$ in the source motion space and a trellis ending at the target frame $M^d(t^d, \mathbf{w}^d)$ in the target motion space as illustrated in Figure 4.2. The trellis is sampled forward in time at discrete intervals in time Δt and parameters $\Delta \mathbf{w}$ up to a maximum depth l_{max} in the source space. Similarly from the target frame a

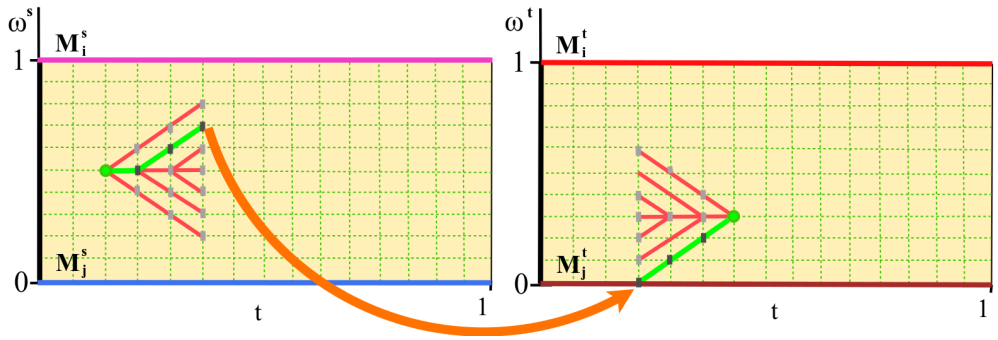


Figure 4.2: *Illustration of the transition between a parametrised motion space source created by the sequences M_i^s (pink) and M_j^s (blue), and a parametrised motion space target created by sequences M_i^d (red) and M_j^d (brown). A trellis linking the current location on the source parametric space and the target location is built, shown in red, with the candidate transition frames, shown in grey. The green path represents the optimal path P_{opt}. Orange arrow shows the actual transition. Motions used to create a parametric space are depicted in Figure 4.1.*

trellis is constructed going backward in time. This defines a set of candidate paths $P \in \Omega$ with transition points between each pair of frames in the source and target trellis. For a path P the latency cost $E_L(P)$ is measured as the number of frames in the path P between the source and target frames. Transition similarity cost $E_S(P)$ is measured as the similarity in mesh shape and motion at the transition point between the source and target motion space for the path P.

4.4.2 Transition Similarity Cost

The mesh and motion similarity between any two source and target frames $s(M^s(t^s, \mathbf{w}^s), M^d(t^d, \mathbf{w}^d))$ is defined as follows. As the vertex correspondence between the source and target meshes are known we can compute the shape and motion similarity between any pair of meshes. The Euclidean distances between their vertex positions and velocities gives a distance

$$d(M_i, M_j) = \frac{1}{N_x}(\|x_i - x_j\|^2 + \lambda\|s_i - s_j\|^2), \tag{4.3}$$

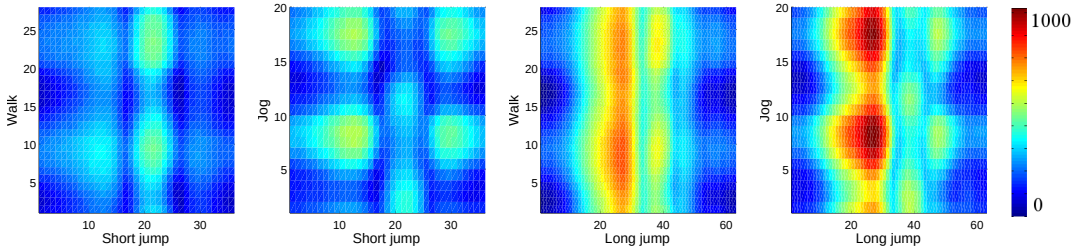


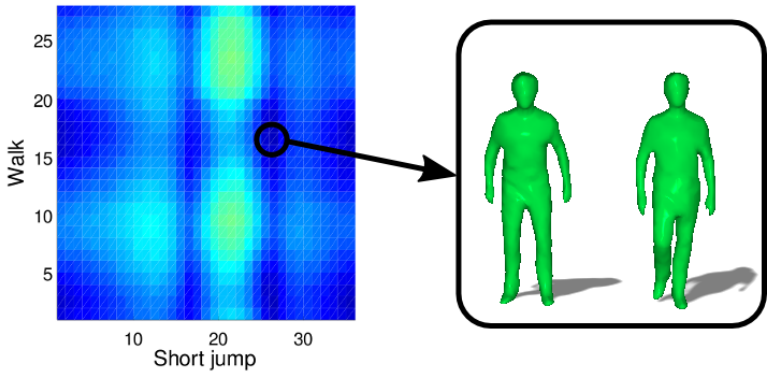
Figure 4.3: *Similarity matrices computed between two parametric spaces each created from two input motions. (a) walk and short jump similarity; (b) jog and short jump similarity; (c) walk and long jump similarity; (d) jog and long jump similarity.*

where vertex velocity $s_i(t) = x_i(t) - x_i(t-1)$ and N_x number of vertices. Similarity is then computed by normalising by the maximum distance:

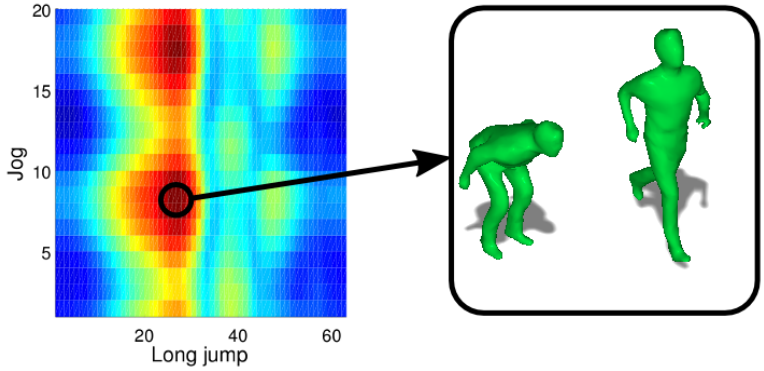
$$s(M_i, M_j) = 1 - \frac{d(M_i, M_j)}{max(d(M_i, M_j))}. \tag{4.4}$$

Evaluation of the similarity $s(M^s(t^s, \mathbf{w}^s), M^d(t^d, \mathbf{w}^d))$ between pairs of frames in a continuous source and target parametric motion space is prohibitively expensive for online computation. Real-time computation can be achieved by approximating the similarity using an interpolation of the corresponding pre-computed similarity between the mesh sequences used to build the source and target motion spaces. Section 4.4.3 gives further details on how the matrix of similarities $\mathbf{S}$ is pre-computed offline and how arbitrary similarities are approximated at run time.

Figure 4.3 depicts an example of the precomputed similarities between two parametric motion spaces, each of them built from two motions: *node A*, containing a *walk* and a *run* motion, and *node B* containing a *short jump* and a *long jump* motion. Subfigures 4.3a, 4.3b 4.3c 4.3d illustrate the four similarity matrices. Dark blue areas indicate good possible transitions, with high shape similarity between the two linked poses. A more detailed illustration of the similarity matrices is presented in Figure 4.4. Figure 4.4a highlights the point where the highest similarity is located, showing the two poses that correspond to this point. Similarly, Figure 4.4b highlights the least similar location, as well as the meshes evaluated there. A simple visual qualitative inspection shows that, while the poses in Figure 4.4a are similar in shape (both consist in an



(a) *Highest similarity is found at the highlighted location, where the two illustrated poses are compared.*



(b) *Lowest similarity is found at the circled location, where the two rendered poses are compared.*

Figure 4.4: *Highest and lowest similarities found between an example 2-node 4DPMG created from a walk and jog motion (node A); and a short and jump motion (node B).*

upright standing position), the poses in Figure 4.4b are highly dissimilar, validating the proposed approach for mesh similarity evaluation.

To achieve real-time evaluation of the optimal transition with accurate approximation of the true similarity we introduce a non-linear bisection approach analogous to the hybrid mesh blending presented in Section 3.2. Source and target motion parameter spaces are subdivided with offline non-linear interpolation of intermediate meshes. Real-time online computation of the similarity between motion spaces is then performed by linear interpolation of the similarity. Bisection subdivision allows accurate approximation of the true non-linear similarity. In practice, a single bisection gives an

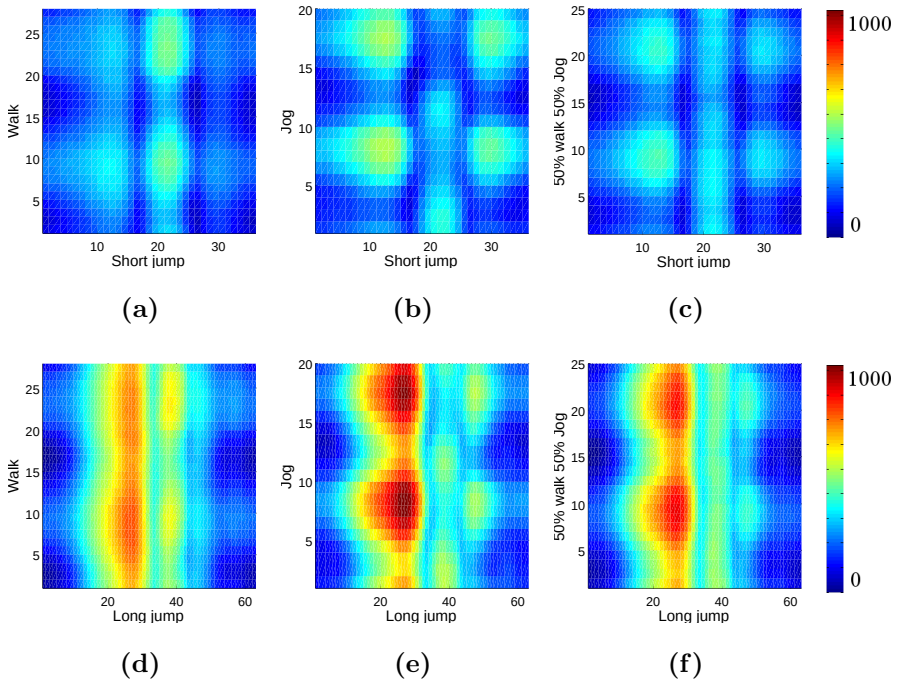


Figure 4.5: *(a,b) Similarity matrices for input sequences walk/short jump, and jog/short jump; (c) Similarity matrix for non-linear blended sequence* $M_{NL}(\mathbf{r}), \mathbf{r} = \{0.5\}$ *and short jump. (d,e) Similarity matrices for input sequences walk/long jump and jog/long jump. (f) Similarity matrix for non-linear blended sequence* $M_{NL}(\mathbf{r}), \mathbf{r} = \{0.5\}$ *and long jump.*

accurate approximation for path optimisation as discussed in Section 4.4.5. As in the hybrid mesh blending approach, the price paid is a small pre-computation step and a modest increase of memory usage. The proposed approach enables the computation of mesh similarities with significantly lower errors than using the linear method, while maintaining the online performance.

Figure 4.5 depicts similarity matrices generated using the proposed approach for similarity interpolation in a simple 4DPMG consisting of two nodes: node A, containing walk and jog motion; and node B, containing short and long jump. Subfigures 4.5a and 4.5b illustrate the similarity matrix between walk/short jump and jog/short jump respectively. Subfigure 4.5c presents the interpolated similarity matrix for $\mathbf{M}_{NL}(\mathbf{w}), \mathbf{w} = \{0.5\}$. Subfigures 4.5d,4.5e and 4.5f show another example for walk/jog node and long jump motion.

4.4.3 Optimal Transition Path Evaluation

As explained in Section 3.2, from a set of captured mesh sequences $\{\mathbf{M}_p(t)\}_{p=1}^{N}$ defining a parametric motion space we can pre-compute interpolated mesh sequences $M_{NL}(t, \mathbf{r}) = b(\mathbf{M}, \mathbf{r})$, where $b()$ is a non-linear blend function and $\mathbf{r}$ the blending weights. Bisection subdivision of the parametric motion space gives a set of reference mesh sequences $\{M_R(t, r_p)\}_{p=1}^{N_R}$ composed of the set of captured mesh sequences $\{M_p(t)\}_{p=1}^{N}$ and a non-linear set $\{M_{NL}(t, r_p)\}_{p=1}^{R}$ using a blending weight set $\{\mathbf{r}_p\}_{p=1}^{R}$ where $N_R = N + R$.

To evaluate the transition cost between a source and target parametric motion space we pre-compute the shape and motion similarity between each pair of source and target reference mesh sequences. For each pair of source $M_R^s(t^s, \mathbf{r^s})$ and target $M_R^d(t^d, \mathbf{r^d})$ mesh sequences we evaluate the shape and motion similarity $s(M_R^s(t^s, \mathbf{r^s}), M_R^d(t^d, \mathbf{r^d}))$ for all frames $t^s \in [0, T^s], t^d \in [0, T^d]$ giving a matrix $\mathbf{S}_{sd}$. Pre-computing the similarity between all pairs of source and target mesh sequences gives a similarity matrix $\mathbf{S}$ of size $N_R^s T^s \times N_R^d T^d$, where N_R^s and N_R^d are the number of reference mesh sequences in the source and target motion space respectively. Figure 4.3 illustrates an example of the similarity matrices between two input parametric spaces created from a walk/jog and short/long jump respectively. Figure 4.5 illustrates an example of the in-between non-linear similarity matrices precomputed for $\mathbf{r} = \{0.5\}$ in the walk/jog space and both the short and long jump sequences.

Online real-time computation of the shape similarity between the mesh in the source $M^s(t^s, \mathbf{w}^s)$ and target $M^t(t^d, \mathbf{w}^d)$ spaces for any source $\mathbf{w}^s$ and target $\mathbf{w}^t$ parameter value can then be evaluated as a weighted sum of the similarity of the corresponding pairs of reference meshes in source $M_R^s(t^s, \mathbf{r}_p^s)$ and target $M_R^t(t^d, \mathbf{r}_q^d)$ where $\mathbf{r} = h^{-1}(\mathbf{w}) \in [0, 1]$. For convex weights it can be shown that the following inequality holds (see Appendix A for proof):

$$
\begin{aligned}
& s(M^s(t^s, \mathbf{w}^s), M^d(t^d, \mathbf{w}^d)) \\
& \geq \sum_{p=1}^{N^s} \sum_{q=1}^{N^d} \mathbf{r}_p^s \mathbf{r}_q^d s(M_R^s(t^s, \mathbf{r}_p), M_R^t(t^d, \mathbf{r}_q)) \\
& \geq \mathbf{r}^s \mathbf{S}^\top (\mathbf{r}^d)^\top
\end{aligned} \tag{4.5}
$$

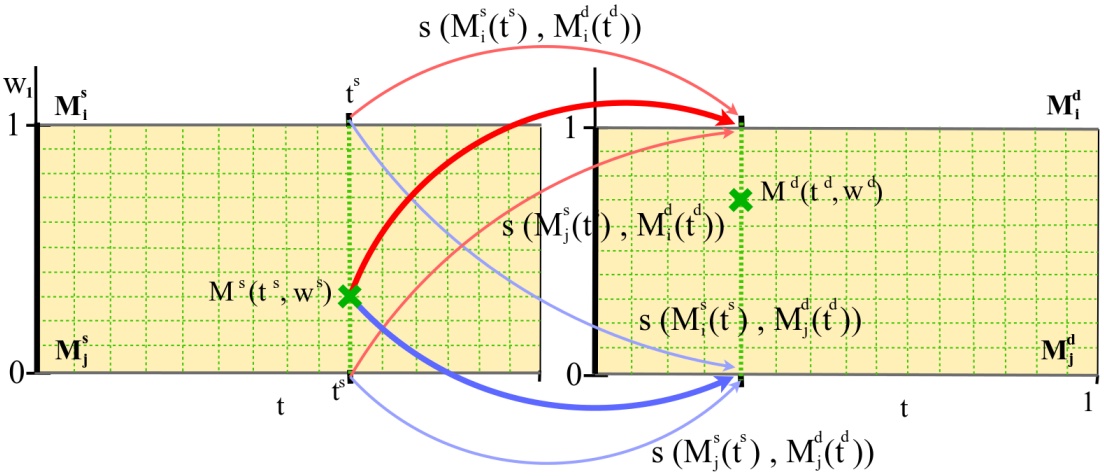
Thus a lower-bound on the similarity between any two frames in the parameterised source and target motion space can be evaluated by a sum of the weighted similarity between the source and target mesh sequences. As the pairwise sequence similarities can be pre-computed, evaluation of the maximum lower-bound on similarity between any point in the two parameterised spaces can be efficiently computed at run-time. If $\mathbf{w}^s$ and $\mathbf{w}^d$ are the source and target weight vectors then we can evaluate the maximum similarity according to Equation 4.5. Bisection subdivision of the source and target parametric motions spaces to give a set of non-linearly interpolated reference meshes results in a piecewise linear approximation of the non-linear similarity which can be efficiently evaluated at run-time.

Figure 4.6 illustrates an example of how the approximate similarity between two arbitrary meshes $M^s(t^s, w^s)$ and $M^d(t^d, w^d)$ is computed in a scenario with $\mathbf{r} = \{\}$. First, the true similarities $s(M_i^s(t^s), M_i^d(t^d))$, $s(M_j^s(t^s), M_i^d(t^d))$ and $s(M_i^s(t^s), M_j^d(t^d))$, $s(M_j^s(t^s), M_j^d(t^d))$ are retrieved from the matrix of precomputed similarities $\mathbf{S}$ and each pairs is interpolated according to the blending weight w^s, giving the approximate similarities $s(M^s(t^s, w^s), M_j^d(t^d))$ and $s(M^s(t^s, w^s), M_i^d(t^d))$ as shown in Subfigure 4.6a. Finally, the resulting pair of interpolated similarities is again interpolated, according to the destination node weight w^d, giving the requested approximated similarity $s(M^s(t^s, w^s), M^d(t^d, w^d))$, as illustrated in Figure 4.6b.

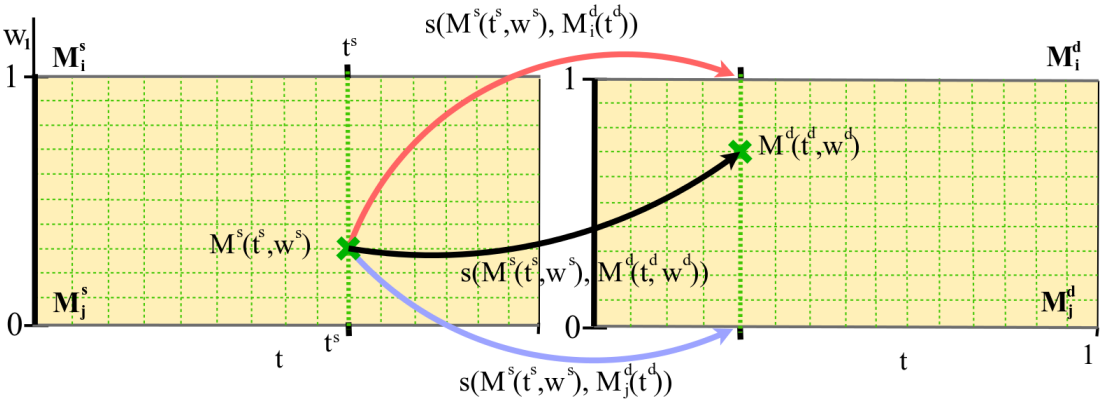
The optimal transition path P_{opt} for a given source frame (current state) and target frame (end state) is evaluated according to Equation 4.2 at run-time by evaluating the cost for all paths in the trellis defining possible transitions between the source and target motion spaces. The similarity cost $E_S(P)$ for path P is defined as the similarity at the transition frames between the source and target motion spaces evaluated according to Equation 4.5:

$$E_S(P) = s(M^s(t^s, \mathbf{w}^s), M^d(t^d, \mathbf{w}^d)). \tag{4.6}$$

The optimal path P_{opt} can be evaluated with low-latency at run-time because computation of the similarity cost by interpolation, Equation 4.5, is computationally efficient. Furthermore, the path is optimised across all possible transitions between the trellis in the source and target motion spaces as discussed in Section 4.4.1.



(a) *Step one for online similarity approximation for any arbitrary pair of meshes* $s(M^s(t^s, w^s), M^d(t^d, w^d))$. *Similarities* $s(M_i^s(t^s), M_i^d(t^d))$ *and* $s(M_i^s(t^s), M_j^d(t^d))$; $s(M_j^s(t^s), M_i^d(t^d))$ *and* $s(M_j^s(t^s), M_j^d(t^d))$ *are retrieved from the matrix* **S** *and interpolated according to the weight* w^s.



(b) *Final step, similarities* $s(M^s(t^s, w^s), M_i^d(t^d)$ *and* $s(M^s(t^s, w^s), M_j^d(t^d)$ *are interpolated according to weight* w^d

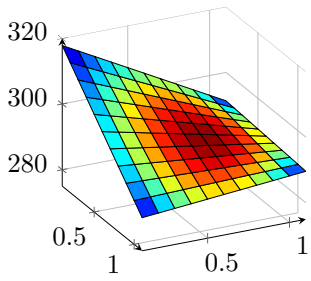
Figure 4.6: *Online approximation of the similarity between two parametric meshes* $s(M^s(t^s, w^s), M^d(t^d, w^d))$.

4.4.4 Mesh Similarity Evaluation

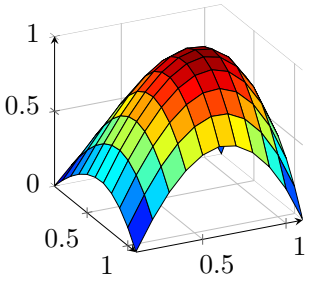
As stated in Section 4.4.2, a matrix $\mathbf{S}$ containing the similarities $s(M_R^s(t^s, \mathbf{r_p}), M_R^d(t^d, \mathbf{r_q}))$ is precomputed offline and used at runtime to approximate any similarity $s(M^s(t^s, \mathbf{w}^s), M^d(t^d, \mathbf{w}^d))$. The approximation error depends on the weights in $\mathbf{r_p}$ and $\mathbf{r_q}$, which are based on a recursive bisection subdivision of the weight space.

Figure 4.7 shows the influence of the number of subdivisions in the weight space $\mathbf{r}$ on the final performance of the proposed approach. Figure 4.7c characterises the ground-truth non-linear similarities at time $t^s = 0.08$ and $t^d = 0.54$ between a parametric space built from walk and run motions and a parametric space built from low jump and high jump motions of the dataset *Dan*. Figure 4.7a is the result of approximating these similarities using a linear approximation, coloured as a heat map to highlight the differences with respect to the non-linear approach. Figures 4.7d and 4.7f characterise the similarities obtained using one subdivision and two subdivisions of the weight space respectively. Figures 4.7b, 4.7e and 4.7g show the error (difference between hybrid linear approximation and ground-truth non-linear similarity) normalised to the range [0,1]. Note that even with a single bisection, Figure 4.7e, the maximum error in similarity computation is $< 33\%$ and with two bisections, Figure 4.7g, the maximum error in similarity is $< 10\%$.

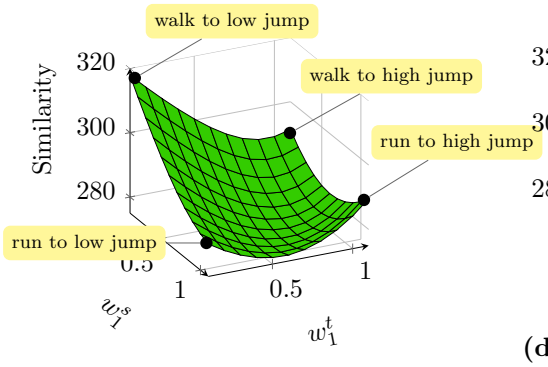
Table 4.1 shows the computational time and quantitative errors for figures of Figure 4.7. Notice how the error with respect to the non-linear case decreases significantly using the proposed approach with one subdivision ($\mathbf{r} = \{0.5\}$). Using two recursive subdivisions ($\mathbf{r} = \{0.25, 0.5, 0.75\}$) the error is not significant in the computation of the best transition. Real-time online performance is maintained with the evaluation of all similarities for a trellis of depth 10 taking less than 9ms for all subdivision levels. These quantitative results demonstrate that the proposed approach gives accurate approximation of the non-linear similarity between parametric motion spaces whilst allowing computationally efficient evaluation for real-time motion control. The price paid is pre-computation of the reference meshes and evaluation of pairwise similarities that need to be stored in memory. Throughout this work two subdivisions have been used to accurately approximate the similarities between the different nodes of a 4DPMG.



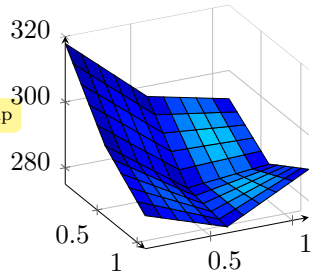
(a) *Approximation of the similarities using a linear approach.*



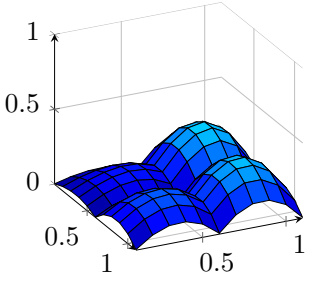
(b) *Normalised geometric representation of the linear errors.*



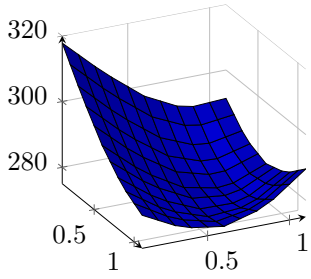
(c) *Non-linear similarities.*



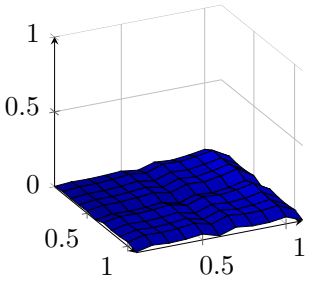
(d) *Approximation using the proposed approach with* $\mathbf{r} = \{0.5\}$.



(e) *Normalised geometric representation of the errors of Figure 4.7d.*



(f) *Approximation using the proposed approach with* $\mathbf{r} = \{0.25, 0.5, 0.75\}$.



(g) *Normalised geometric representation of the errors of Figure 4.7f.*

Figure 4.7: *Influence of the number of subdivisions in the weight space* $\mathbf{r}$ *on the error in similarity computation. The two mesh parametric spaces being compared have been built from a walk and a run motions, and from a high jump and low jump motions.* $t^s = 0.05$ *and* $t^d = 0.54$, $t \in [0, 1]$. *Errors and computational time are presented in Table 4.1.*

Method	offline time	online time	avg. error	max. error
Non-linear	0.000 s	160.58 s	0.000	0.000
Linear	39.698 s	0.009 s	10.05	17.61
1 subdivision	78.087 s	0.009 s	2.55	5.88
2 subdivisions	238.023 s	0.009 s	0.64	1.74

Table 4.1: *Computational time (offline and online) and errors of the proposed approach presented in Section 4.4.4 to approximate the computation of mesh similarities for all possible transition with a trellis depth = 10. Evaluation of the transition example shown in Figure 4.8d.*

Method	$\mathbf{E_S(P_{opt})}$	$\lambda\mathbf{E_L(P_{opt})}$	**Latency(s)**	**online CPU time (s)**	**offline CPU time (s)**
1 Precomputed	251.10	207.00	1.656	0.000005	93.32
5 Precomputed	236.13	191.59	1.533	0.000007	93.32
10 Precomputed	237.28	208.38	1.444	0.000010	93.32
Online depth = 12	254.41	166.99	1.336	0.000190	12.56
Online depth = 10	276.30	116.63	0.933	0.000085	12.56

Table 4.2: *Comparison of transition cost, latency and computation time for pre-computed fixed transitions with the proposed online computation(λ = 5, frame-rate= 0.04s). Cost and computation times are average values of 50 randomly generated transitions for each method between walk/run and long/short jump parametric motion spaces.*

4.4.5 Transition Performance Evaluation

The performance of the proposed online approach for transitions between parametric motion spaces has been evaluated against a naïve offline method in which a fixed set of the n best transitions between two parametric spaces was precomputed and stored. We present both quantitative and qualitative evaluations. The former is presented using mesh similarity and transition latency metrics, while the latter is shown using animation still renders of the resulting transitions.

Table 4.2 presents quantitative results for transition cost, latency and computation time averaged over 50 transitions with random start and end points in the source and target motion space. Results demonstrate that the proposed online path optimisation achieves lower latency (less time delay in transition) for a similar transition cost, whilst

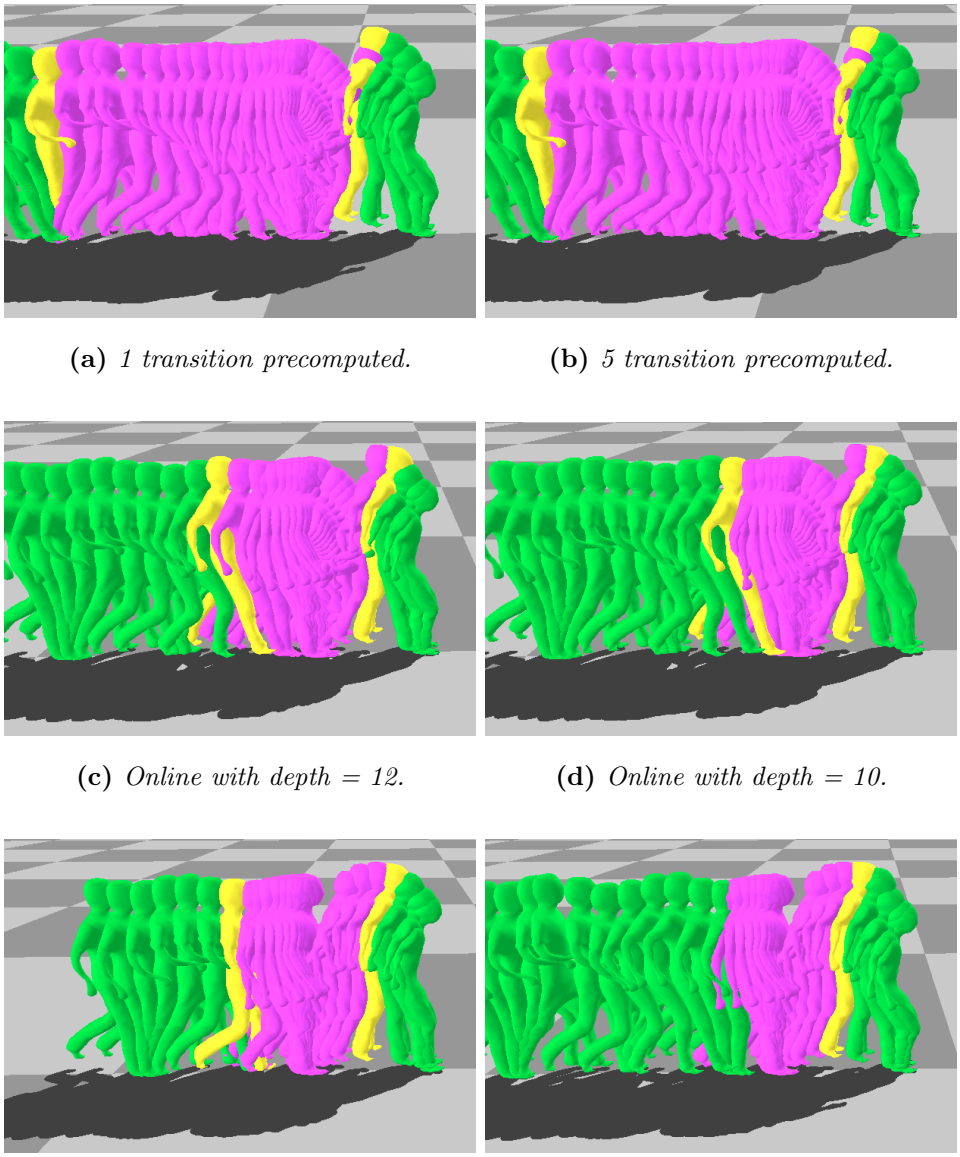
maintaining real-time performance for evaluation of transitions (<0.2ms/transition). Pre-computation of fixed transitions gives a marginally lower similarity as it selects the best transition points, but results in significantly higher latency.

Figure 4.8 shows a qualitative comparison of the same transitions using 6 different configurations with dataset Dan. Figure 4.8a shows the result using 1 precomputed transition, forcing the character to transit always in the same point in the parametric space. In that setup, from the moment that the transition is requested we have to wait until the character reaches that specific point, thus a relatively high-latency transition is expected. Similarly, Figure 4.8b shows the result of the same transition using 5 precomputed transitions. Since the amount of transition points to be evaluated is also small and constant, high-latency transitions are as well expected in this case, as it is depicted by the large amount of pink meshes in the figure. Figures 4.8c to 4.8f show the performance of the proposed online path optimisation approach using trellis depths (l_{max}) of 12, 10, 5 and 3 respectively. A smooth transition with reduced latency (smaller number of transition frames) is produced for online path optimisation with a trellis depth of 12 and 10 as shown in Figures 4.8c and 4.8d. Figures 4.8e and 4.8f show that as the trellis depth is reduced to 5 and 3 unnatural jumps in the motion appear at the transition. This is due to the severely restricted path propagation, causing the optimisation to fail in finding a good transition point between the motion spaces within the trellis. Further transition examples using dataset Dan are shown in Figure 4.9b and 4.9a.

In this work we use time step size $\Delta t = \alpha T$ and parameter step size $\Delta \mathbf{p} = \alpha(\mathbf{w}_{max} - \mathbf{w}_{min})$ with $\alpha = 0.1$, where the parameter range between the interpolated motion samples is $[\mathbf{w}_{min}, \mathbf{w}_{max}]$.

4.5 Results

In order to test the proposed approach, a variety of 4DPMG are built using the datasets mentioned in Section 3.5.1. Interactive characters are loaded into a OpenGL [opeb] application described in Appendix B that allows real-time motion control and online render of the resulting pose in a 3D virtual scenario.



(a) *1 transition precomputed.*

(b) *5 transition precomputed.*

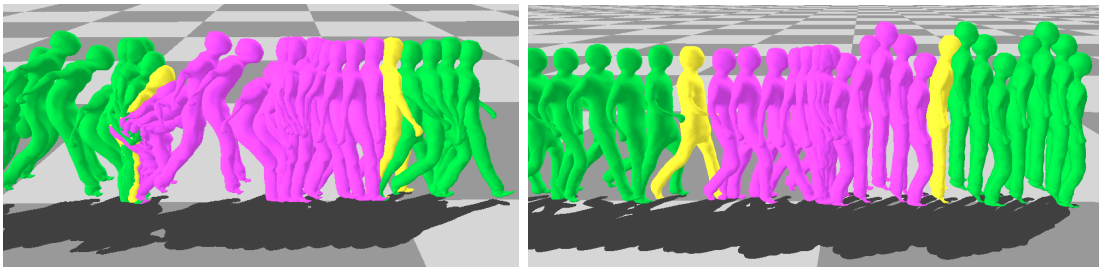
(c) *Online with depth = 12.*

(d) *Online with depth = 10.*

(e) *Online with depth = 5.*

(f) *Online with depth = 3.*

Figure 4.8: *Comparison of the same transition between a run motion* $w = 0.9$ *of the parametric space showed in Figure 3.14a to a jump motion* $w = 0.65$ *of the parametric space showed in Figure 3.14b. In yellow, source* $M^s(t^s, \mathbf{w}^s)$ *and target* $M^d(t^d, \mathbf{w}^d)$ *meshes. In pink the meshes computed to transition.*



(a) *Transition from long jump to walk.*

(b) *Transition from jog to low jump. Notice how the character automatically reduces the speed in the first part of the pink section, until reaching the walk pose where the optimal transition was found.*

Figure 4.9: *In yellow, the source and target poses requested by the user. In pink the meshes automatically generated to generate the transition. Using dataset Dan.*

4.5.1 Qualitative Results

Figures 4.10, 4.11, 4.12, 4.14, 4.13 4.15 4.16, and 4.18 present a variety of motion sequences interactively created using the dataset Dan with the proposed 4DPMG implementation. Animated versions of each of these figures as well as further results are included in the supplementary video attached to this thesis. Figures 4.16 and 4.17 show similar results for database Infantry, notice how the long jumping motion is significantly exaggerated in this dataset. Resulting clips combine walk, jog, turn, different styles of jump, and reaching motions. Different colours are used to highlight changes in motion parametrisation.

Visual qualitative evaluation of these results confirms that our approach for real-time character animation from 4D video capture achieves realistic results. Parametric transitions generated with the framework introduced in this chapter maintain the visual quality of the captured sequences. Meshes which have been online automatically synthesised can hardly be distinguished from the captured meshes, both in sill figures and animated video. Transition latency is always < 1.2 seconds, enabling interactive control of the motion with minimal lag between the requested pose and the actual transition.

Object interaction is also demonstrated with 4DPMG. In Figure 4.15, wooden boxes are rendered to act as an obstacle that the character needs to avoid. Our approach

also allows the user to draw a path that will be followed by the character, avoiding the obstacles. In Figure 4.13, wooden benches separated by different distances were loaded to increase the realism of the parametric jumps. In Figure 4.17 character Infantry interactively avoids two piles of wood of different lengths, followed by a sharp left turn. A realistic grass texture is used to add more realism into the final render.

Animations generated using the proposed method can be also combined with traditional 2D video footage, as Figure 4.18 demonstrates. Standard camera tracking methods applied to video footage allow the extraction of camera location parameters. Animation results can then be rendered from the requested viewpoint, allowing real-time control of a synthetic character virtually moving in 3D *inside* real video footage.

4.5.2 Limitations

Results presented in this chapter demonstrate the potential for interactive character animation from 4D performance capture. 4DPMG enable real-time interaction with high-level movement control. Nevertheless, the current implementation presents some limitations.

Online computation of transitions between motion spaces with up to 3 parameters can be performed in real-time ($< 0.2ms$) for trellis depth < 10. Online optimisation of transition paths is shown to give reduced latency compared to fixed precomputed transitions. This allows optimal transitions to be computed with respect to the current state. However, higher-dimensional spaces require an exponentially increasing number of candidate transitions to be evaluated. GPU implementation and pre-computation of transitions costs could be employed at the expense of increased storage and fixed transitions locations leading to increased latency.

The current implementation relies on the assumption that a *good* transition point will be found in the dataset. However, this is an assumption that may not be true, specially if the dataset being used contains a sparse variety of motions with minimal similarity between them. Hence, the proposed approach may produce unrealistic transitions between parametric spaces if the input data does not contain a minimal similarity between

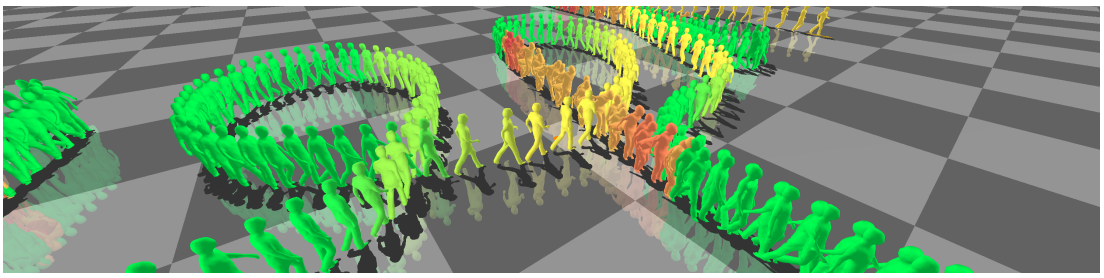


Figure 4.10: *Real-time interactive character animation with a 4DPMG. The presented approach allows us to exploit a small dataset of 4D motions and create realistic novel motions.*

some of the clips. In order to avoid unpleasant transitions, a threshold could be specified in Function 4.6, forbidding all transitions which the similarity is lower than a certain value. If no *good* transition can be found, the character must remain in the current node.

Nodes of the proposed 4DPMG framework consist in independent parametrised motion space. In order to produce meaningful interpolated blends, nodes have to be populated with semantically similar motions, which are currently manually selected. This limitation is overcome in previous motion graph research with skeletal motion capture, MoCap, by methods for automatic sequence labelling [KG04, MC12, KTT*12]. Analogous methods for 3D-mesh sequences can potentially be investigated to relax the current manual selection of similar motions, thus fully automating the definition of the graph structure.

Finally, it is important to highlight that the proposed approach for character animation from 4D performance capture applies only to the geometry of the motion. However, datasets used in this work have been created from studio capture footage, as shown in Chapter 2, hence the appearance of the captured models is available. Chapter 5 investigates methods for texturing parametric models to synthesise photorealistic character animation from 4D performance capture.

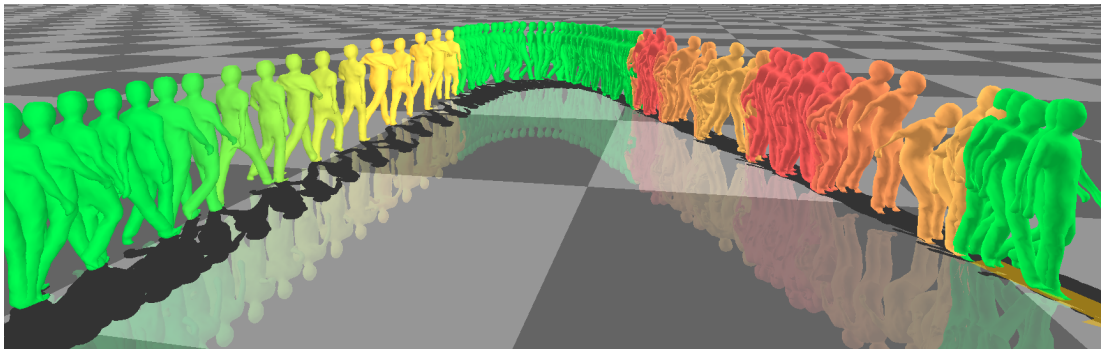


Figure 4.11: *Close-up of the character Dan going from walk to jog, then turning right and and transitioning into a parametric jump motion.*

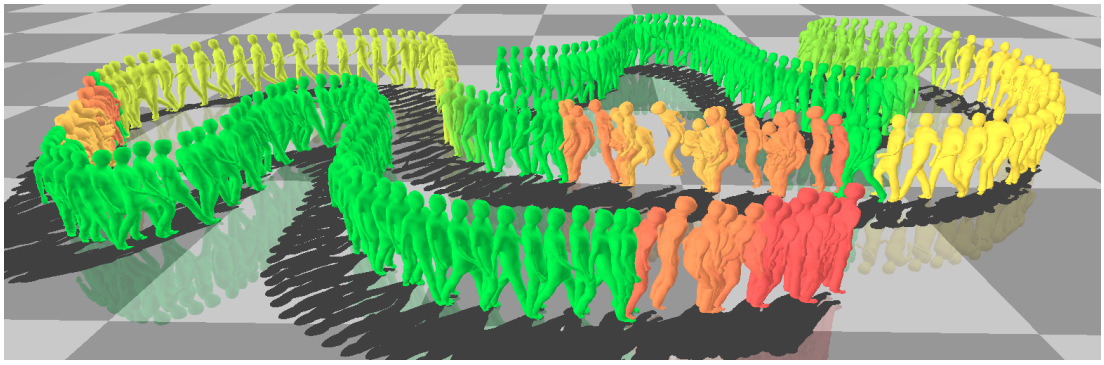


Figure 4.12: *All kind of turn can be synthesised combining walk and turn motions. Transitions to other parametric motions such as jump control can be performed at any time.*

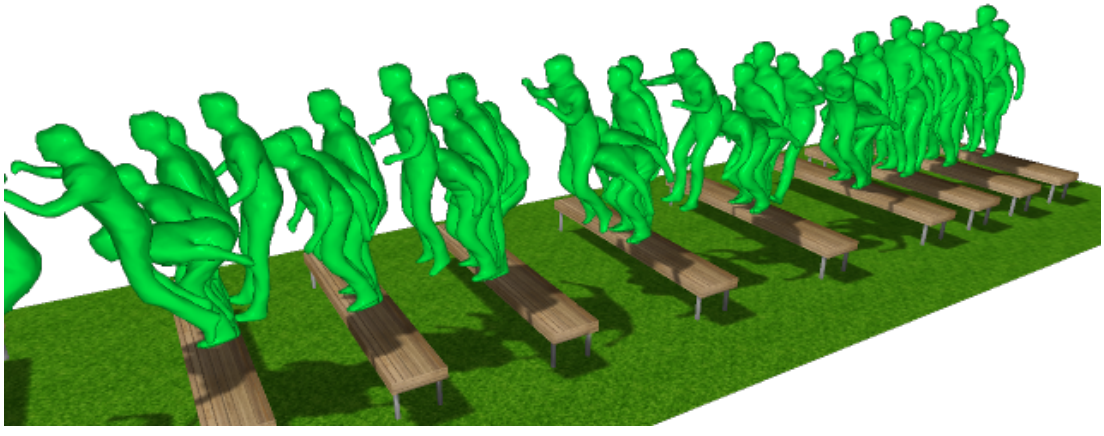


Figure 4.13: *CG objects can be loaded into our OpenGL application, increasing the realism of the final result. In this case, a parametric jump motion is used to jump between benches, non-equally spaced.*

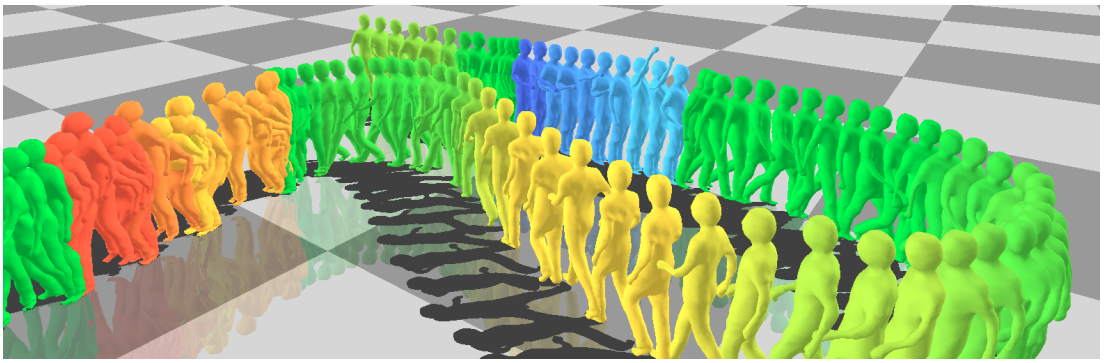


Figure 4.14: *A character interactively controlled combines walk, run, reach and jump. Using dataset Dan.*

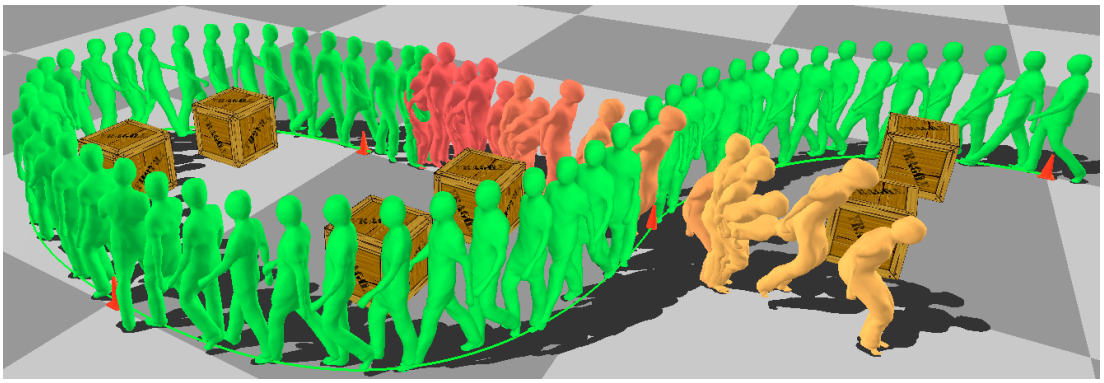


Figure 4.15: *A character automatically follows a path set by the user to reach the red cones, avoiding the obstacles. Using dataset Dan.*

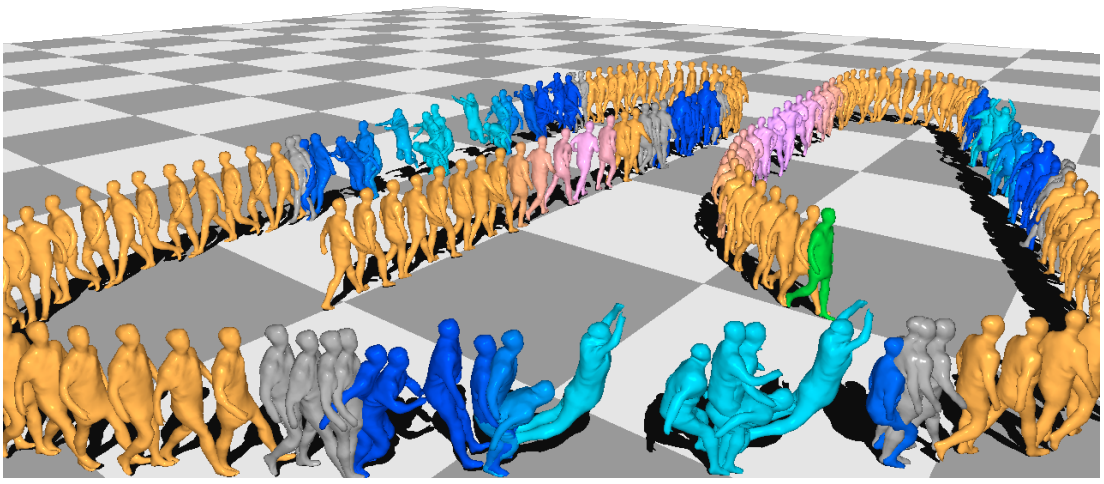


Figure 4.16: *Infantry character was interactively controlled, combining motions such as walk, jog, left turn, right turn, short jump and long jump. Grey meshes indicate transitions between parametric spaces.*

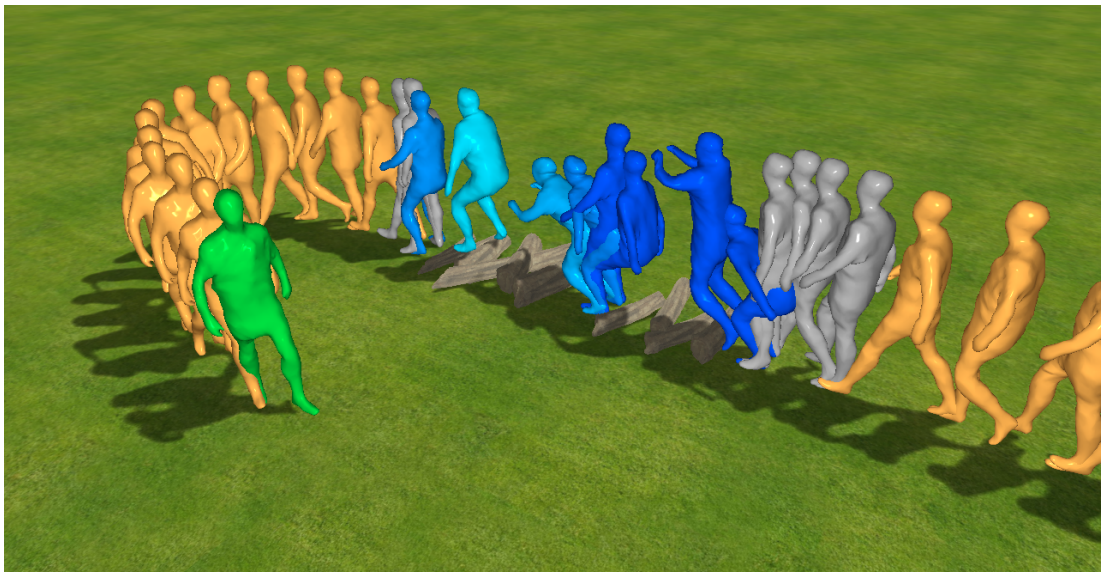


Figure 4.17: *Infantry character interactively controlled to jump over obstacle of varying width, combining motions such as walk, jog, left turn, right turn, short jump and long jump. Meshes coloured depending on parametrisation.*

Figure 4.18: *4DPMG also enables the possibility to create animations that can be rendered into a traditional 2D video footage or still images. Interactive novel human motion can be combined with real footage.*

Chapter 5

4D Video Textures

Visual realism remains a challenging goal in the production of character animation. In film visual-effects, realism is achieved through highly skilled modelling and animation, commonly using captured video to provide an artistic reference. For interactive content a combination of material reflectance maps and static texture are commonly employed, using captured imagery to provide realistic visual detail. Materials may be modulated according to character motion and behaviour to create a more life-like appearance. The use of static materials and texture results in a loss of visual realism compared to the dynamic appearance of people in captured video.

This limitation has motivated interest in video-based rendering to reproduce the detailed dynamic appearance of real-world scenes. Initial image-based approaches [CW93, DTM96, KR97] achieved novel viewpoint synthesis of static scenes using image interpolation techniques. Free-viewpoint video [ZKU*04] enables video-realistic rendering of novel views from multi-view footage, but is limited to replay of the captured performance. The inference from multiple-view video of structured 4D video models, representing the dynamic 3D surface shape and view-dependent appearance over time, allows greater reuse of the captured performance [CTMS03, dST*08, VBMP08, GSdA*09, XLS*11]. The motion of the resulting character models may be modified via skeletal rigging, or direct surface manipulation, and rendered with the captured video, to reproduce a detailed dynamic appearance that conveys a high degree of visual realism.

As discussed in Chapter 4, 4D video geometry may be readily manipulated in such pipelines by blending [CTGH11], concatenating [HHS09, CTGH12a, CTGH13] or editing captured model sequences [ACP03, KG08, VBMP08, TH11]. However the *dynamic appearance* of the character remains that of the original motion, baked into the captured video. For example the wrinkling of skin or clothing during movement remains fixed if the motion is retimed or exaggerated. This limits the extent to which the character motion can be modified whilst maintaining visual realism. Consequently there remains a gap between existing video-based characters in interactive application and full video-realism.

This chapter presents 4D Video Textures (4DVT) which allow the motion and dynamic appearance to be interactively controlled whilst maintaining the visual realism of the source video. The approach is based on the parametrisation of a set of 4D video examples for an actor performing multiple related motions, for example a short and long jump, following the approach introduced in Chapter 3. Appearance for an intermediate motion is produced by aligning and combining the multiple-view video from the input examples to produce plausible video-realistic dynamic appearance corresponding to the modified movement. As the character motion changes so does the dynamic appearance of the rendered video reflecting the change in motion. 4DVT enable real-time character animation with interactive control of motion and viewpoint whilst maintaining the video-realism.

5.1 Related work

Authoring photorealistic character animation has been a popular research topic in the Computer Graphics community for the last two decades. Synthesising novel realistic video is a challenging goal due to the observed surface appearance depending on complex physical properties including illumination, shadows, shape, motion, reflectance, sub-surface scattering and viewpoint.

As discussed in previous Chapters 2 and 3, a common approach to produce human motion clips consists in reusing captured video data. The same idea has been used in

the literature to synthesise photorealistic video characters [SMH05, FNZ*09, XLS*11], exploiting datasets of video footage to render new clips.

The proposed methods can be divided into two group: 2D video-based animation methods, which aim to generate 2D-video characters; and 3D video based methods, whose goal is to author free-viewpoint video character animation that can be rendered from any viewpoint.

5.1.1 2D video-based animation

Video Rewrite [BCS97] introduced the resampling of video frames to synthesise novel video sequences for facial animations. Phonemes from input videos of actors talking were automatically labelled and mouth image frames were reordered to match the query phonemes of the new audio track. Subsequent research on *Video Textures* [SSE00] demonstrated animation of a variety of dynamic scenes by concatenating segments of the source video with transitions between frames with similar appearance and motion. Resulting videos maintain the captured realism of the input clips. 2D-video animation is further investigated using *video sprites* [SE02], an extension of the previous work [SSE00] which provides the user with flexible transition control of the animation. However, the image-based frame-to-frame similarity used by Schöld and Essa [SE02] limits the video footage where the approach can be applied to non-rigid or stochastic motions, and cannot handle changes in appearance due to viewpoint modification and self-occlusion, thus the approach is not suitable for human motion. *Video textures* are further extended by Celly and Zordan [CZ04], using the ratio of height to width for the human silhouette bonding box to identify transitions in a relatively small dataset with manually selected subsets of motions. However, the poor morphing model to generate transitions usually results in ghosting artefacts.

2D approaches for video-based animation preserve the video-realism of the source data in the generation of novel sequences but are limited to a fixed viewpoint and replaying the captured motion. These approaches rely on an appearance-based similarity metric to identify similar frames in the source video for transitions. However, appearance-based similarity may fail for dynamic scenes with self-occlusion, such as video of a

person, due to the similar appearance of occluding parts resulting in erroneous transitions. In order to overcome this limitation, Hornung *et al.* [HDK07] combined skeletal motion capture, MoCap, with traditional 2D images to animate a single frame of an articulated figure. As in skeletal motion graph the 3D MoCap information provides a measure of the similarity between frames which is robust to self-occlusion. Manual correspondences between the image features and 3D motion features are imposed. Limb occlusion in the original images can be handled, however their method requires manual intervention for the initial fitting step and does not handle video footage. *Human Video Textures* [FNZ*09] also combined 2D imagery and 3D skeletal information to produce animations of people by augmenting the video acquisition with 3D motion-capture markers allowing the identification of suitable frames for transitions. This achieves video-realistic animation but is limited to both replaying segments of the captured video restricting the range of movement and viewpoint

Dynamic Textures [CYJ02] enabled the synthesis of new poses and motions. Stable texture patches were identified in the input frames using a coarse geometric proxy of the motion. New texture patches to populate the requested pose were created by modulating a PCA basis of patches. Recent research [HFE13] has introduced *pose space image based rendering* demonstrating photorealistic animation of clothing from a set of 2D images augmented with 3D shape information to support natural transitions. This approach could potentially be extended to concatenation of 4D video sequences to incorporate video-realistic dynamics. Nevertheless, these approaches do not allow interactive viewpoint control.

To overcome the limitation of fixed viewpoint in 2D video concatenation, methods for viewpoint control of video replay are required. Initial image-based approaches [CW93, DTM96, KR97] achieved novel viewpoint synthesis of static scenes using image interpolation techniques. Zitnick *et al.* [ZKU*04] achieved free-viewpoint video that maintain the visual quality of the source video over a limited range of viewpoints by combining multiple synchronised video streams. Lipski *et al.* [LLB*10] propose a space-time interpolation method for complex real-world video-scenes, enabling the synthesis of a limited range of novel viewpoints by combining captured images in a space-time interpolation domain. A tetrahedralisation of the space is used to inter-

polate correspondences between the original footage. Similarly, *unstructured video-based rendering* [BBPP10] uses coarse scene reconstruction from multiple view video captured with moving hand-held cameras to produce non-photorealistic transitions between viewpoints of complex dynamic scenes. More recently, Tompkin *et al.* introduced *Videoscapes* [TKKT12], extending the 2D video concatenation to allow exploration of cities from unstructured video collections.

5.1.2 3D video-realistic animation

Video-realistic animation and free-viewpoint rendering from 4D performance capture has also been investigated. *Starck et al.* [SMH05] present a video-based approach to character animation using a motion graph representation. Sequences of people are represented using 2D spherical geometry images, which provides a temporally consistent structure. A novel spherical matching technique was used to find transition between captured motion clips. *Surface Motion Graphs* [HH09, HHS09] allow motion synthesis from 3D video sequences according to user constraints on movements, position and timing. Transitions, based on shape similarity, are automatically found to satisfy user constraints. These approaches achieve a video-realistic quality similar to free-viewpoint video rendering but animation is limited to replay of the segments of the captured 4D video.

Chapter 4 presents an approach to generate novel 3D mesh sequences via 4D Parametric Motion Graphs (4DPMG), however only geometric information is taken into account. Synthesised mesh motions textured with the original video-footage result in unrealistic appearance due to artefacts such as unnatural texture dynamics, texture ghosting and stretching caused by the mismatch between the novel geometry and the originally captured video textures. To increase realism, flexibility and range of animation, methods to synthesise textures for novel poses are required. In the literature we can find that a common approach to solve this problem is based on texture warping from both distinct poses and multiple viewpoints [ECJ*06, SLAM08]. *Floating Textures* [EDDM*08] introduce a warping method based on real-time optical flow computation that improved projective texturing of a 3D geometry proxy from multi-view camera capture.

A recent advance in free-viewpoint video-realistic character animation that enables the synthesis of novel poses was presented by Xu *et al.* [XLS*11]. *Video-based characters* indexes a database of 4D performance capture sequences to retrieve a sequence of multi-view source images which best match a query motion and rendering viewpoint for each frame. Image-based warping is employed to adjust the retrieved image sequence to the query pose and viewpoint. This results in photo-realistic rendering of novel video sequence of the character which match the target motion and can be composited within a background video. However their method is not suitable for interactive character animation.

The 4D Video Textures (4DVT) approach introduced in this work combines multiple 4D videos to interpolate the detailed video dynamics. The proposed approach maintains the dynamic appearance of the video as the motion changes allowing real-time interactive animation with free-viewpoint video-realistic rendering. 4DVT enables for first time interactive video-realistic visualisation of motions that were never captured from any view point.

5.2 4D Video Textures Animation Pipeline

The pipeline used to create interactive video-real character animation from 4D performance capture in illustrated in Figure 5.1 and comprises the steps listed below:

1. **Multi-camera capture:** Actor performance is captured in a multi-camera chroma-key studio to facilitate foreground segmentation [SMN*09], Figure 5.1(a), as previously discussed in Section 2.2.

2. **4D video models:** A dataset of 3D-mesh sequences is created [SH07b] and temporally aligned [BHKH13] following the approach discussed in Section 2.2.1, building a 4D video dataset. Figure 5.1(b).

3. **Free-viewpoint rendering:** In Figure 5.1(c), photo-visual realism for the reconstructed captured meshes is achieved by traditional free-viewpoint video rendering techniques are used to render photo-realistic characters [BBM*01, CTMS03, SH07b, VBMP08, dST*08, GSdA*09].

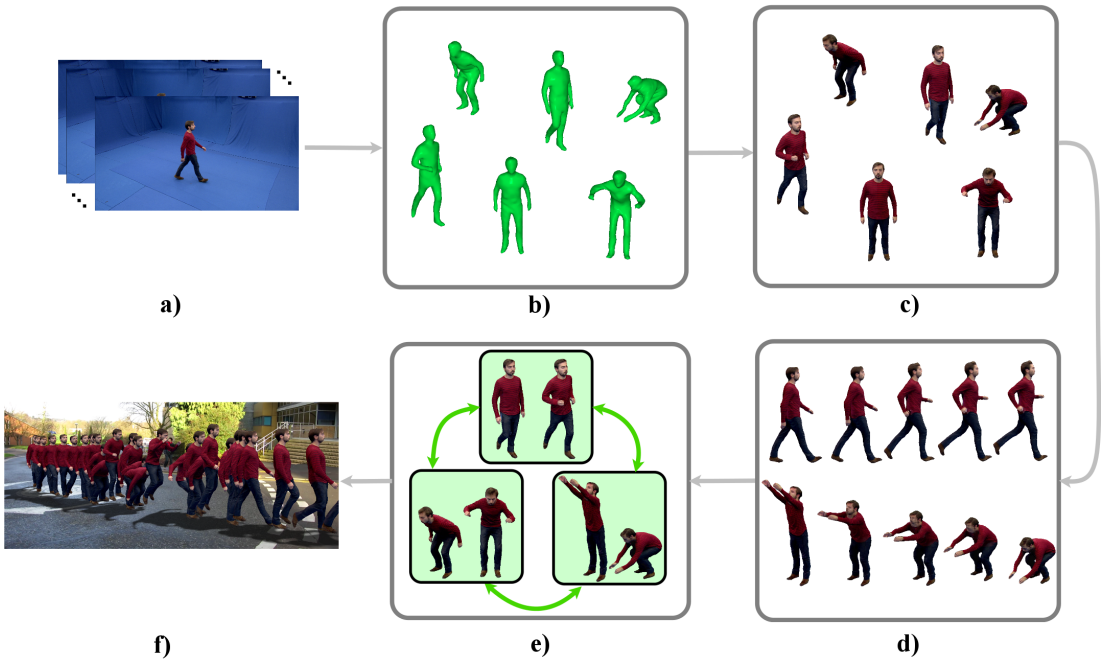


Figure 5.1: *Overview of the proposed framework of 4D Video Textures for interactive animation. (a) Multi-camera view performance capture. (b) Mesh reconstruction and alignment. (c) Standard free-view point rendering. (d) Dynamic texturing with 4D Video Textures. (e) 4D Video Textures Motion Graph. (f) Resulting video-realistic interactive animation.*

4. **4DVT rendering:** Novel 4D video sequences are synthesised by the combination of multiple captured 4D video. The underlying geometric proxy is generated following the approach of Section 3.2. A novel approach for view-dependant alignment of multiple-view video input is used to synthesise the dynamic appearance of the character. Shown in Figure 5.1(d), as the character motion changes, so does the dynamic appearance of the rendered video.

5. **4DVT Motion Graphs:** A motion graph structure, illustrated in Figure 5.1(e), is built to enable transitions between different 4DVT motions based on appearance as well as shape. This representation is an extension of the 4D Parametric Motion Graph earlier introduced in Section 4.2.

6. **Video-realistic composition:** Finally, shown in Figure 5.1(f), video-realistic character animation can be synthesised and composited together with traditional 2D video footage. Using standard camera tracking algorithms, and a coarse

3D reconstruction of the scenario, a 3D video-realistic character is interactively controlled *inside* 2D video scenario with moving camera.

Steps 1, 2 and 3 are discussed in the previous chapters of this thesis; steps 4 and 5 are discussed in detail in the remaining of this chapter; and step 6 consists in a standard video composition. The main contribution of this chapter is the introduction of 4DVT, a novel method for interactive appearance synthesis that allows for first time both viewpoint and motion control of an interactive video-realistic character.

5.3 4DVT: 4D Video Textures

5.3.1 Definition

A 4D video $F(t) = \{V(t), M(t)\}$ combines multiple view video sequence $V(t) = \{I_c(t)\}_{c=1}^{C}$ with C camera views combined with a 4D proxy of the dynamic surface shape represented as a mesh sequence $M(t)$, where vertices correspond to the same surface point over time. This form of representation has previously been employed for free-viewpoint video rendering of dynamic scenes [CTMS03, SH07b, dST*08, VBMP08]. Free-viewpoint video renders a novel video sequence $I(t, v)$ for a viewpoint v from the set of input videos $V(t)$ using the mesh sequence $M(t)$ as a geometric proxy [BBM*01]. The objective of free-viewpoint video is to maintain the visual realism of the source video whilst providing the flexibility to interactively control the viewpoint. However, free-viewpoint video is limited to replay of the captured performance and does not allow any change in scene motion.

The goal of 4D Video Textures (4DVT) is to allow interactive control of the scene motion and rendering viewpoint whilst maintaining video quality rendering. Previous approaches to video-based character animation from 4D video have proposed techniques for resampling frames or segments of the captured data and warping either the geometry or appearance to match the desired motion [SMH05, XLS*11]. However, these approaches only allow a limited range of motion control and do not take into account the dynamics of the surface appearance resulting in loss of realism in the rendered video.

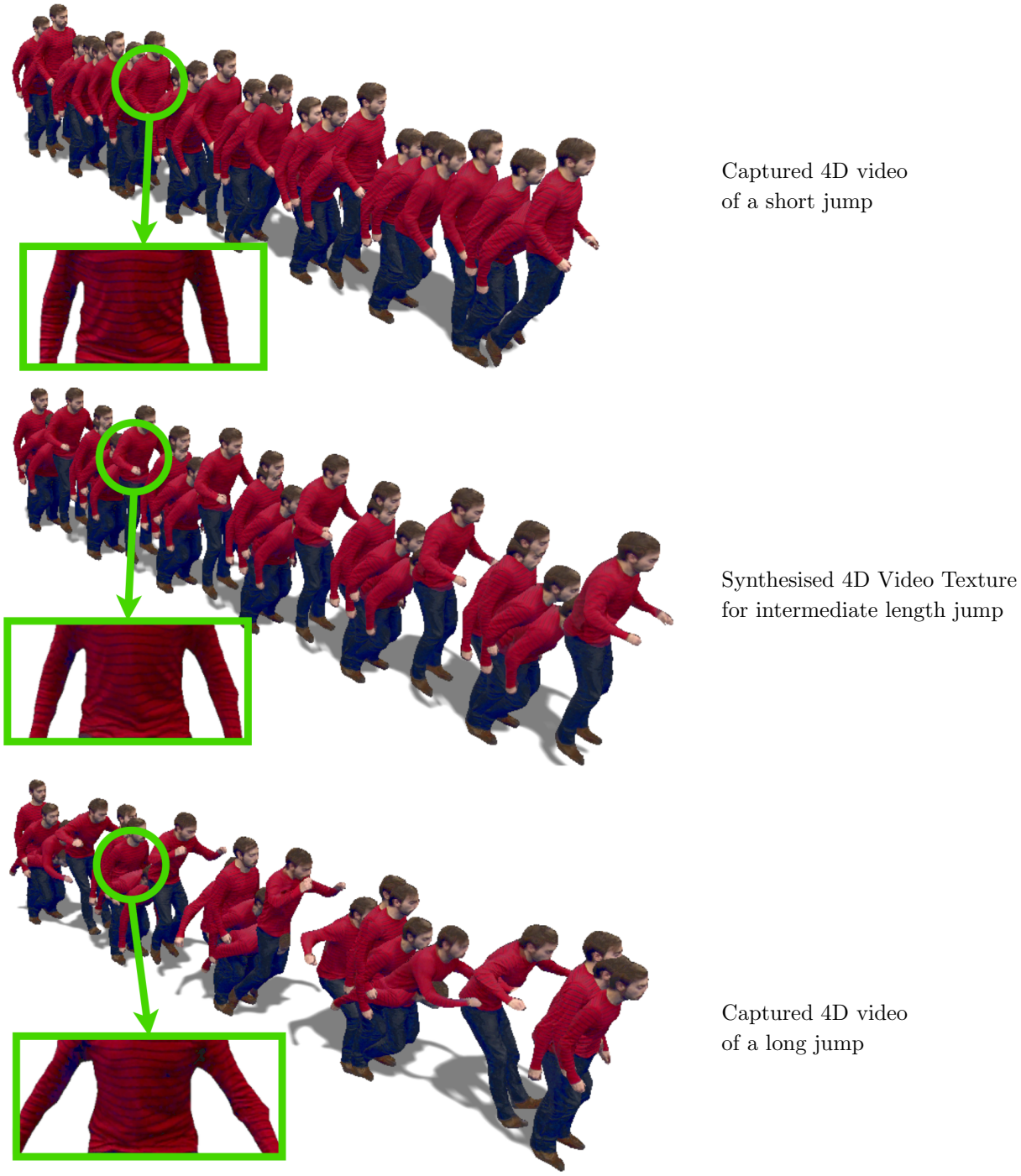


Figure 5.2: *4D Video Textures: Original 4D videos for short (top) and long (bottom) jumps. Centre: 4DVT synthesised medium jump (middle). In green, a close-up to highlight texture details, notice how the the synthesised sequence (middle) maintains visual quality of the captured sequences. Using dataset Dan.*

4DVT introduce a new approach which changes the dynamic appearance as the underlying motion changes to maintain plausible video realism across a wider range of motion. Figure 5.2 illustrates the problem — given two 4D videos as input, how can we render an intermediate motion with a visual-quality comparable to the input motion and plausible dynamic appearance? A straightforward approach is to use the mesh sequence interpolation scheme introduced in Chapter 3, and then we use the original multi-camera video footage to incorporate appearance into the blended geometry. However, this solution will result in unnatural appearance dynamics due to the change in shape surface, as show later in this section in Figure 5.4. Linear blend of the source video textures projected into the blended geometry also fails in providing plausible appearance, suffering from ghosting artefacts due to residual errors of the mesh sequence alignment step as well as changes in appearance across the sequences.

Therefore, the problem is addressed by combining multiple 4D videos $\{F_i(t)\}_{i=1}^{N}$ to render a novel video sequence with interactive control of both the scene motion and viewpoint. The general problem can be stated as follows. Given a set of motion control parameters $\mathbf{w}$ and viewpoint v, we aim to render a novel video $I(t, w, v)$:

$$I(t, \mathbf{w}, v) = h(F_1(t), ..., F_N(t), \mathbf{w}, v), \tag{5.1}$$

where $h(.)$ is a function which combines the source 4D videos according to the specified motion parameters $\mathbf{w}$ and viewpoint v. The rendered video $I(t, \mathbf{w}, v)$ should preserve the visual quality of both the scene appearance and motion. Note that both $\mathbf{w}$ and v can be functions of time t, but for simplicity purposes we write $\mathbf{w}$ and v instead of $\mathbf{w}(t)$ and $v(t)$.

The approach presented in this thesis embeds the set of 4D videos $F_i(t)$ in a parametric motion space $\mathbf{p}$, where $\mathbf{p} = f(\mathbf{w})$ represents high-level motion parameters such as speed and direction. This representation enables interactive control of the motion and viewpoint to render a video output in two stages:

1. Synthesis of a 4D shape proxy $M(t, \mathbf{w})$ by non-linear combination of the input mesh sequences using the approach presented in Section 3.2.

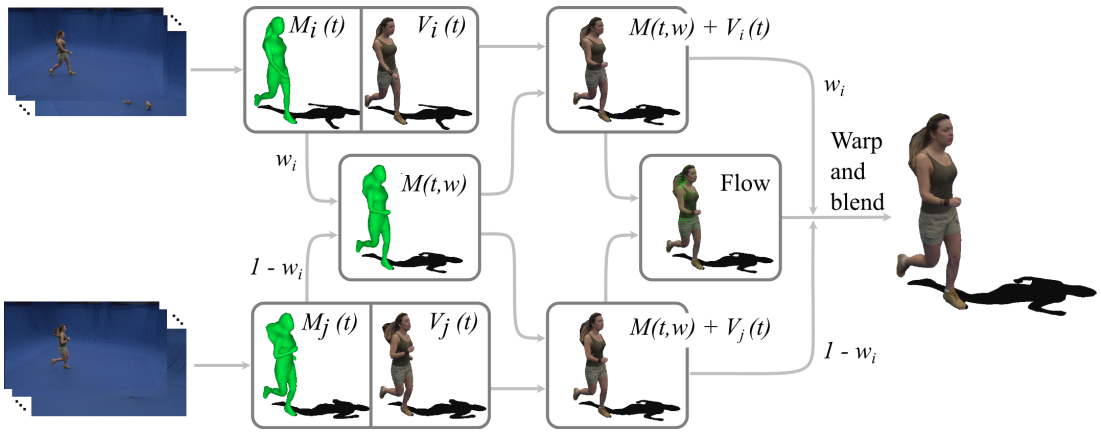


Figure 5.3: *Overview of the proposed 4D Video Texture pipeline. Illustrated using dataset 'Fashion'.*

2. View-dependent rendering of the output video $I(t, \mathbf{w}, v)$ based on real-time alignment of the rendered multi-view source videos $V_i(t)$ using the 4D shape proxy $M(t, \mathbf{w})$.

Figure 5.3 presents an overview of 4DVT rendering pipeline. View-dependent optic flow alignment of two 4D videos is performed based on the 4D proxy mesh $M(t, \mathbf{w})$ to produce the rendered video sequence with the motion $\mathbf{w}$ and dynamic texture appearance. This process is performed in real-time allowing interactive manipulation of both the viewpoint v and motion parameters $\mathbf{w}$. In order to synthesise an appearance that maintains visual realism our approach assumes that the set of source 4D videos $\{F(t)\}$ are of the same subject performing related motions. Throughout this work the proposed framework have been applied to multiple-view video sequences of people performing a variety of related motions, for example walk, run, turn, reach and grab. 4DVT allow the creation of video characters with interactive video and motion control and free-viewpoint rendering which maintain the visual quality and dynamic appearance of the source videos. Intermediate motions are rendered with plausible dynamic appearance for the whole-body, cloth wrinkles and hair motion.

5.3.2 Interactive Control of the 4D Shape Dynamics

To interactively control the scene dynamics the combination of multiple mesh sequences is required to compute a geometric proxy. Given a set of input sequences $M_i(t)_{i=1}^N$ representing the surface shape dynamics for multiple related motions, the objective is to synthesise a novel sequence $M(t, \mathbf{w})$ which reproduces the desired motion defined by a set of high-level motion parameters $\mathbf{w}$. For example, if the inputs are mesh sequences for walk, turn and run motions, the output motion would be parametrized to allow interactive control of speed and direction. In the work presented in this thesis this is achieved using the hybrid non-linear approach presented in Section 3.2, providing real-time interactive control of the motion.

5.3.3 View-dependant 4DVT Rendering

The critical challenge for video quality rendering with control of both motion and viewpoint is the combination of multiple 4D videos $\{F_i\}$ to render a novel output video $I(t, \mathbf{w}, v)$. A naive approach to render the output video $I(t, \mathbf{w}, v)$ for a given set of motion parameters $\mathbf{w}$ and viewpoint v is to use the known mesh correspondence to transfer the multiple view video for each input mesh sequence $M_i(t)$ to the interpolated proxy shape $M(t, \mathbf{w})$ and blend the resulting textures from multiple input motions. However, any misalignment in the underlying geometric correspondence or change in appearance due to the motion will produce blurring and ghosting artefacts.

The transfer of appearance between 4D videos for a walk and run motion is illustrated in Figure 5.4. Figures 5.4(a) and 5.4(b) show both the captured geometry and captured texture for a walk and run pose, respectively. Figure 5.4(c) shows the walk texture $V_i(t)$ transferred to the jog geometry $M_j(t)$, notice how the texture distorts causing the face to look unrealistic. Analogously, Figure 5.4(d) shows running appearance $V_j(t)$ projected into walk geometry $M_i(t)$, distortion artefacts are also present. Appearance distortion artefacts and deformed face features in figures 5.4(c,d) are caused by the difference in surface dynamics for the two source motions. Figures 5.4(e-h) show the same behaviour in an example using character Dan. Figures 5.5(a,b) show the result of naive linear blending of the transferred appearance into a blended mesh $M(t, \mathbf{w}), \mathbf{w} =$

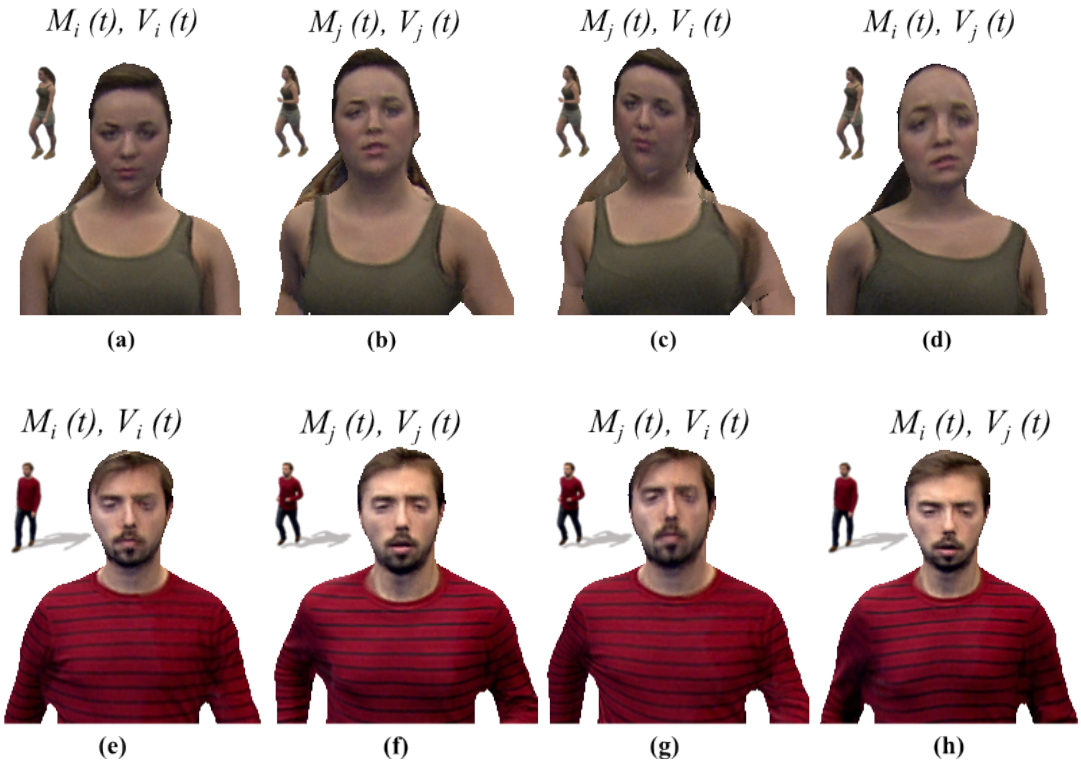


Figure 5.4: *4D video appearance transfer: In top row, using dataset Roxanne: (a,b) show the original rendered 4D video frames for a walk and a run sequence; (c,d) show the result of appearance transfer, notice the distortion artefacts caused by the difference in surface geometry between the source 4D models. In bottom row, using dataset 'Dan': (e,f) show original poses from a walk and a jog motions. (g,h) show the result of appearance transfer, notice distortion and unrealistic deformations.*

$\{0.5\}$, and a heatmap to highlight differences between the blended textures. Even with accurate surface alignment [HBH11] the 4D video appearance changes for different motions due to the dynamics of the skin, clothing and hair. We therefore require an approach to accurately align the dynamic appearance of the input motions. Figures 5.5(c,d) show the 4DVT result achieved using the view-dependent appearance alignment proposed in this thesis and described in detail in the remaining of this chapter.

4DVT mitigates ghosting and distortion artefacts by inferring an aligned video-texture from the multiple-view videos input $V_i(t)$ for rendering. Unfortunately it is not possible to directly pre-compute the alignment between the multiple-view videos for different

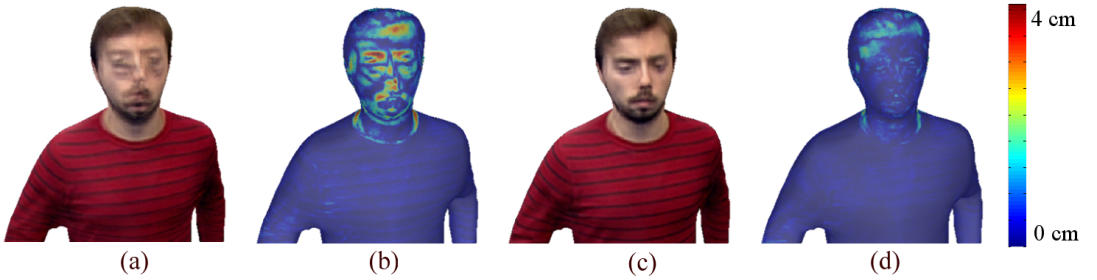


Figure 5.5: *4D video appearance transfer to an intermediate blended geometry ($w = 0.5$): (a,b) result of naive blending, and texture difference map; (c,d) result of 4D video texture view-dependent alignment and blending, and texture difference map. Using dataset Dan.*

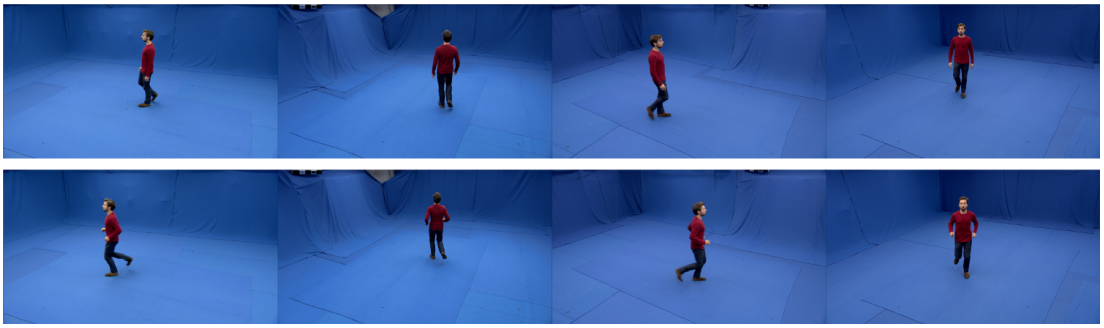


Figure 5.6: *Top: Captured frames from cameras 1, 3, 5 and 7 of a walking pose. Bottom: Captured frames from cameras 1, 3, 5 and 7 of the jog pose corresponding to the walk pose shown in the top row. Direct texture alignment between camera views is not possible due to changes in surface visibility.*

motions as the surface visibility depends on the pose relative to the camera which will change significantly for different motions. Figure 5.6 illustrates an example of this difficulty. Top row shows the captured frames from cameras 1, 3, 5 and 7 of a pose from a walking motion. Bottom row shows the corresponding frames (equivalent poses) from the same cameras of a jog motion. Although the poses captured from each camera are similar, direct alignment pre-computation would fail because of the differences in surface visibility due to changes in pose and location.

An alternative would be to pre-compute a texture map from the multiple view video input and perform alignment in the texture domain. There are two significant problems with this solution: firstly, there may be a significant loss of view-dependent detail in the combination of multiple view video into a single texture map due to errors

in shape and camera calibration, as noted in previous work [EDDM*08]; secondly, alignment in the texture domain is non-trivial due to the distortion of the original surface appearance, discontinuities or seams in the texture map and any errors in the geometric surface alignment. We therefore propose an alternative approach which overcomes these limitations.

Inspired by previous research on *Floating Textures* [EDDM*08] we propose a view-dependent approach for 4D video texture rendering which combines multiple view video input from multiple motions in real-time. Floating Textures proposed a view-dependent optic flow alignment to refine the integration of appearance information from multiple view video. The approach demonstrated high-quality real-time free-viewpoint rendering, avoiding ghosting and blur artefacts which occur due to errors in the shape proxy or camera calibration. However, *Floating Textures* is limited to refinement for a single geometric proxy and corresponding set of multiple view images, thus it is not valid interpolation of multiple motions.

This thesis proposes an approach to render dynamic video textures based on multiple 4D videos of different motions. The set of 4D videos $\{F_i(t)\}_{i=1}^{N}$, with corresponding motion parameters w_i, for a particular motion class are embedded in the parametrised motion space $\mathbf{w}$. Video of novel motions is then rendered by interpolating between the set of 4D videos $\{F_i(t)\}_{i=1}^{N}$ according to the motion parameters $\mathbf{w}$ and viewpoint v. A view-dependent approach is introduced to align the appearance for video sequences with different surface dynamics.

View-dependent rendering with parametrised motion control therefore requires on-the-fly combination of the view-dependent rendering for individual 4D videos $F_i(t, w_i)$. This is achieved by online alignment using optic flow and blending according to the motion parameters of view-dependent appearance. The proxy mesh sequence $M(t, \mathbf{w})$ for the given motion parameters $\mathbf{w}$ is first synthesized from Equatiom 3.7. As the same mesh shape $M(t, \mathbf{w})$ and viewpoint v are used to render each input motion the rendered views share the same geometry and have similar appearance (see full-body miniatures in top-left of subfigures 5.4). Differences in the rendered appearance are primarily due to the different dynamics of the input motion (together with any residual error in the

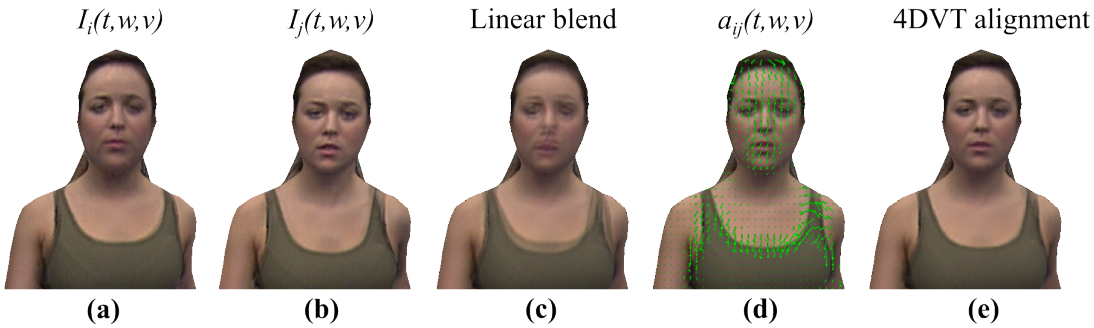


Figure 5.7: *(a),(b) Original walk and run texture projected to a blended geometry* $M(t,w)$, $w = \{0.5\}$, *respectively. (c) Linear blend between (a) and (b), resulting in severe ghosting artefacts. (d) Optic flow [Far03] between (a) and (b). (d) Result of the 4DVT view-dependent texture alignment. Using dataset Roxanne.*

shape reconstruction and mesh sequence alignment).

Our approach to 4D video alignment and rendering based on the proxy mesh sequence $M(t, \mathbf{w})$ comprises the following steps:

Input: multiple view video capture of an actor performing multiple related motions $V_i(t)$, together with the parametrization mesh sequence $M(t, \mathbf{w})$ for user specified parameter $\mathbf{w}$ and rendering viewpoint v.

1. For each input sequence $V_i(t)$ we perform view-dependent rendering of view v using the parametrisation mesh $M(t, w)$ giving a set of image sequences $I_i(t, \mathbf{w}, v)$. See Figure 5.7(a,b).

2. The alignment between each pair of input video sequences is then evaluated using optic flow: $a_{ij}(t, \mathbf{w}, v) = o(I_i(t, \mathbf{w}, v), I_j(t, \mathbf{w}, v))$. See Figure 5.7(c).

3. The combined rendered view is produced by blending of the warped texture view: $I_{out}(t, \mathbf{w}, v) = z(a_{ij}(t, \mathbf{w}, v), I_i(t, \mathbf{w}, v), I_j(t, \mathbf{w}, v), \mathbf{w})$. See Figure 5.5(c).

Output: Parametric video texture $I_{out}(t, \mathbf{w}, v)$

For interactive animation control the image alignment $o(.)$ is required to be performed at run-time. A number of GPU implementations of optic-flow algorithms have been evaluated [Far03, BBPW04, EDDM*08]. We have found the GPU implementation of

the Farneback [Far03] optic flow algorithm (available in OpenCV 2.4 [opea]) to achieve the best results in terms of alignment error with favourable computational performance. Function $z(.)$ implements a per-pixel image warping and blending, weighted according to w:

$$z(a_{ij}, I_i, I_j, w) = I_i^{'}(a_{ij}, w) \cdot w + I_j^{'}(a_{ij}, 1-w) \cdot (1-w), \tag{5.2}$$

where $I^{'}(a, w)$ is the warped image $I()$ according to the flow field a and the weight w such that each pixel $p'(x, y)$ of the warped image I' is computed

$$p'(x, y) = p(x + a_x(x, y) * w, y + a_y(x, y) * w) \tag{5.3}$$

where (x, y) are the pixel coordinates, $a^x(x, y)$ is the magnitude of the flow a in the horizontal axis at (x, y) and a^y is the magnitude of the flow a in the vertical axis at (x, y).

Figure 5.8 presents examples of 4DVT rendering for five characters with intermediate parameter values at intervals $\Delta w = 0.2$ between the two source frames. This illustrates reproduction of detail such as wrinkles for intermediate meshes without blurring and ghosting which occur without view-dependent alignment. Large differences in shape between V_i and V_j for character Dan (row 1) and JP (row 3) are particularly challenging.

Figure 5.9 presents a visual comparison between a linear approach for texture blending and the 4DVT approach presented in this section. Two poses $V_i(t)$ and $V_j(t)$ from the JP dataset are used as a source data. Blended results for weights $\mathbf{w} = \{0.3\}$ and $\mathbf{w} = \{0.6\}$ are presented. Zoomed-in results show that the linear blending approach (top row) fails in aligning input textures, causing the synthesised results suffer from ghosting artefacts. The proposed 4DVT view-dependant optic flow alignment result, highlighted in green in the bottom part of the figure, provide synthetic blended textures that maintain the captured video quality.

Figure 5.10 shows close-up views for two examples of 4DVT rendering, each example interpolated between two source 4D video frames. Full body renders of the source poses are shown in the right and left columns. Notice the smooth change in wrinkle detail as the motion transfers from the right pose to the left. The rendered dynamic appearance detail changes with the underlying surface motion allowing realistic rendering as

Figure 5.8: *4D Video Textures results from Section 5.3.3. Left and right columns are the input free-viewpoint renders of the two poses of each character that are going to be interpolated. Columns 2, 3, 4 and 5 are the synthesised results of the proposed approach $I_{out}(t, w, v)$ for weight values of vector w of 0.8, 0.6, 0.4 and 0.2 respectively. Notice how the synthesised novel poses $I_{out}(t, w, v)$ maintain the visual quality of the input data $V_i(t)$ and $V_i(t)$. Using datasets Dan (top row), Roxanne (2nd row), JP (3rd row), Infantry (4th row) and Knight (bottom row).*

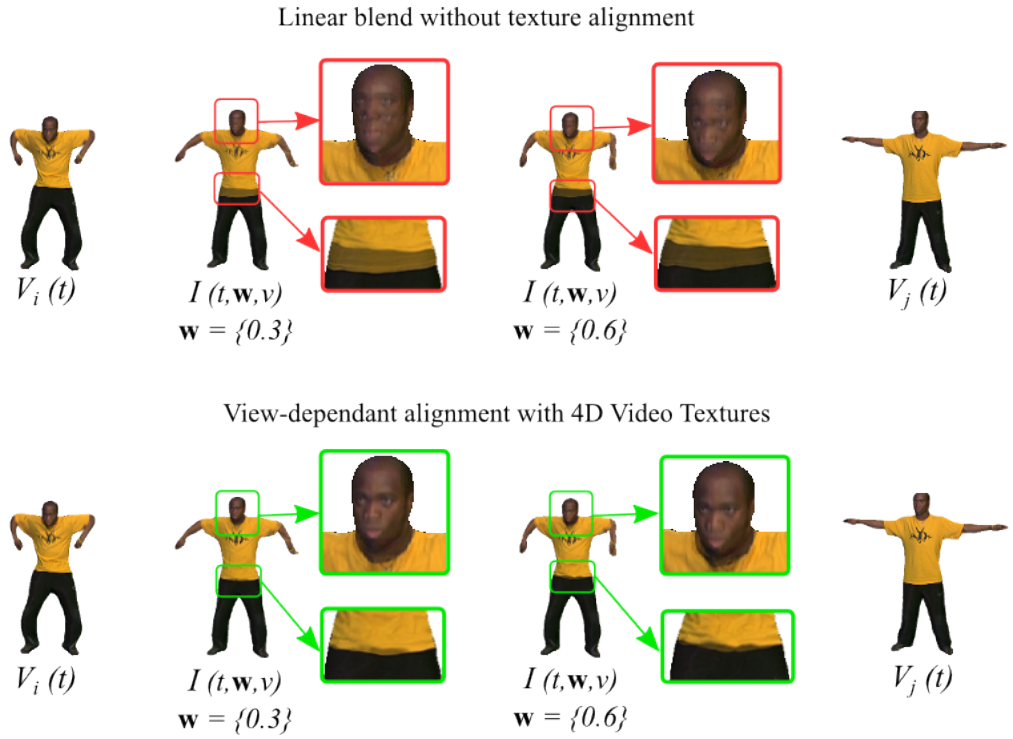


Figure 5.9: *Poses $V_i(t)$ (left column) and $V_j(t)$ (right column) are interpolated using blending weights* $\mathbf{w} = \{0.3\}$ *and* $\mathbf{w} = \{0.6\}$. *Top row shows the results using a linear approach with no texture alignment, zoomed-in details (red boxes) highlight ghosting and alignment artefacts. Bottom row shows the result achieved using the 4DVT alignment approach introduced in this chapter. Zoomed-in details (green boxes) show no texture artefacts.*

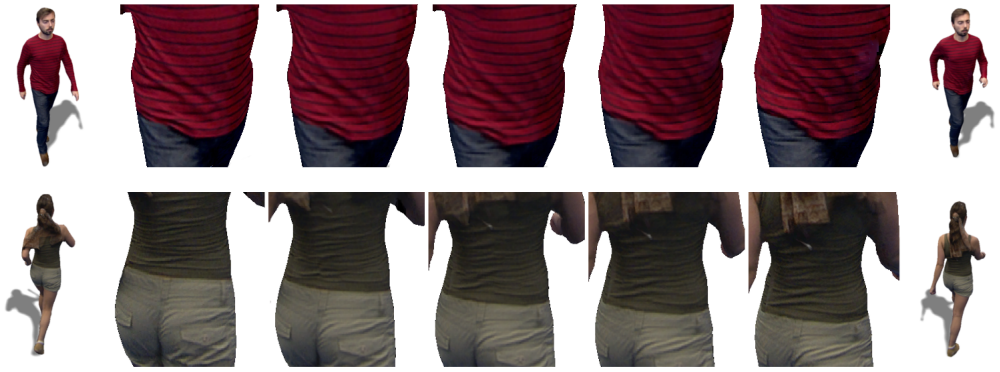


Figure 5.10: *Top-left, top-right: Input 4D video poses taken from walk and jog sequences, dataset Dan. Bottom-left, bottom-right: Input 4D video poses taken from jog and walk sequences, dataset Roxanne. Centre columns, close-up detail of the 4DVT texture alignment approach to interpolate between the source poses. Notice how appearance details smoothly transfers between poses, enabling the synthesis of in-between poses that maintain the captured visual quality.*

the character movement is interactively controlled. In principle the proposed 4DVT framework could also be applied to other dynamic scenes where it is desirable to control motion and viewpoint such as a trees, water, or cloth response to wind.

5.3.4 4DVT Motion Graphs

The approach introduced in Section 5.3.3 enables video-realistic rendering of novel motions from a set of 4D videos of related motions. This allows the construction of a motion class $R(\mathbf{p})$ parametrised by a set of high-level motion parameters $\mathbf{p}$ to control the within class motion. To fully exploit 4D video datasets, video-based character animation within a 3D scene requires the combination of multiple motion classes R_j in a *Motion Graph* structure [KGP02] to encapsulate motion clips and potential transitions between them.

As discussed in Chapter 4, motion graphs [KGP02] have been extended in the literature to represent transitions between parametrised motion classes for skeletal data [HG07] and recently also for surface motion sequences [SMH05, SH07b, HHS09, CTGH13, CTGH12a]. Since 4DVT enables the synthesis of video-realistic parametric animations, a motion graph structure that exploits appearance information is required to represent the potential transitions between 4DVT motion classes.

In this section we address the synthesis of texture for 4D Parametric Motion Graphs [CTGH12a, CTGH13]. This combines the approach for view-dependant alignment of appearance from different motions introduced in the previous section with a parametric motion graph structure. The motion graph is extended to take appearance information into account for transitions between different parametrised motions. Adding appearance information into the originally proposed 4DPMG cost function, described in Equation 4.2, will help in finding transitions based not only in shape similarity, but also in appearance similarity. This results in synthetic animations that avoid glitches and discontinuities in both geometry and texture domain.

Interactive animation requires on-the-fly evaluation of a graph path P for transition between the current motion class $R^s(\mathbf{p}^s)$ and a target motion class $R^d(\mathbf{p}^d)$ with a rendering viewpoint v. On-the-fly optimisation of the graph path is performed over a

discrete trellis of parameter values in the source and target motion class according to the following cost function:

$$P_{opt} = \arg\min_{P \in \Omega} \left(E_S(P) + \lambda_1 E_A(P, v) + \lambda_2 E_L(P) \right), \tag{5.4}$$

where Ω is the set of possible graph paths and $E_S(P)$ and $E_A(P, v)$ evaluate the cost of path P in terms of difference in shape and view-dependent appearances for view v at the transition respectively, and $E_L(P)$ is the latency, or path length, between the current motion state, $R^s(\mathbf{p}^s)$, and the target motion $R^d(\mathbf{p}^d)$. λ_i are weights to balance the relative importance of each term.

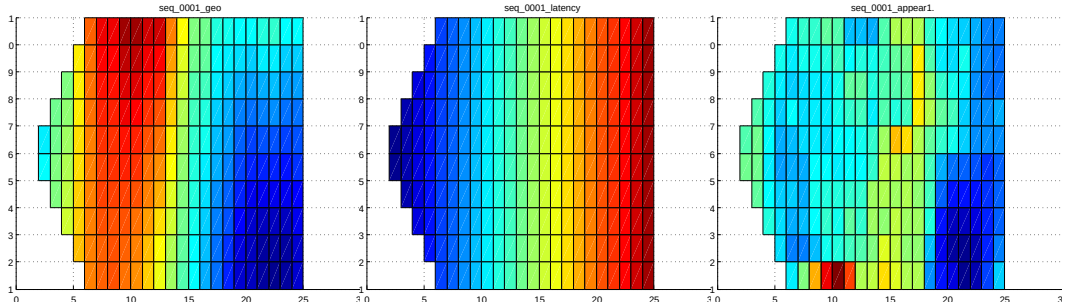
Analogous to 4DPMG, the latency cost $E_L(P)$ for a path P is measured as the path length $|P|$ between the source $R^s(\mathbf{p}^s)$ and target $R^d(\mathbf{p}^d)$ motions. The shape cost, $E_S(P)$, previously defined in Equation 4.6 measures the change in shape and motion at the transition between motion classes.

While shape $E_S(P)$ and latency $E_L(P)$ costs in Equation 5.4 are viewpoint independent, the appearance cost $E_A(P, v)$ depends on the viewpoint v which is interactively controlled by the user and exploits the view-dependent optic flow introduced in Section 5.3.3. The appearance cost $E_A(P, v)$ measures the change in appearance at the transition between motion class for rendering viewpoint v.

$$E_A(P, v) = \|a_{ij}(v)\|^2 \tag{5.5}$$

where $a_{ij}(v)$ is the view-dependent optic flow $a_{ij}(v) = o(I_i(v), I_j(v))$ between the rendered images $I_i(v) \in R^s$ and $I_j(v) \in R^d$. Figure 5.11 illustrates the cost function terms for the motion space R^s transitioning to a frame in the target motion space R^d. Shape and latency costs are independent of the viewpoint, whereas the appearance cost depends on the rendered appearance for a given viewpoint, v, which must be evaluated at run-time. Real-time implementation of the appearance cost $E_A(P, v)$ is achieved using a GPU optic flow implementation (see Section 5.3.3 for details).

The influence of viewpoint v in the appearance cost $E_A(P, v)$ is illustrated in Figure 5.12, which presents a visualisation of the appearance similarity for a transition between a walk/jog node and a jump node created with dataset Dan. Figures 5.12a, 5.12b and 5.12c show a 4DVT character $I(t, v_i, w)$ with $t = 0.6$, $w = 0.5$ rendered from arbitrary



(a) *Shape* **(b)** *Latency* **(c)** *Appearance*

Figure 5.11: *Illustration of the motion graph costs trellis presented in Equation 5.4 for a random transition, time vs. parameter value: (a) Shape similarity $E_S(P)$; (b) Latency $E_L(P)$; and (c) Appearance similarity $E_A(P, v)$.*

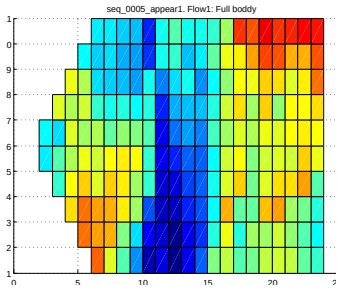
viewpoints v_1, v_2 and v_3 respectively. Figures 5.12d, 5.12e and 5.12f illustrate the appearance similarity trellis for each of the views. Notice that dark blue areas of viewpoint v_1, which indicate high appearance similarity, differ significantly from v_2 and v_3. Such severe difference in appearance similarity between viewpoints is caused by *what* is actually being rendered in each particular v_i. While v_1 renders a frontal full-body view of the character, viewpoints v_2 and v_3 render close-ups of the upper body, thus ignoring other body parts for appearance similarity computation. Consequently, if the target mesh in the motion space R^d has a large shape difference, evaluation of the shape similarity $E_S(P)$ will result in a low score; however, if viewpoint v_i is zoomed-in into a region of the character with high appearance similarity, such as the face in this example, $E_A(P, v)$ will result in a high score. This observation demonstrates that the appearance term $E_A(P, v)$ incorporated into the proposed cost function of Equation 5.4 adds meaningful information when searching for transition candidates. Since shape and appearance similarity are not directly correlated it is important to consider both in the minimisation problem to be solved when transitioning between parametric motion spaces.

An example of this observation is illustrated in Figure 5.13, which shows the result of requesting a transition between a walk/jog and a walk to stand motion from two different viewpoints v_i and v_j. The optimal transition from viewpoint v_i, which shows a view of the full-body as display in Figure 5.13b, is found at $t_s = 0.95$, $w_s = \{0.1\}$

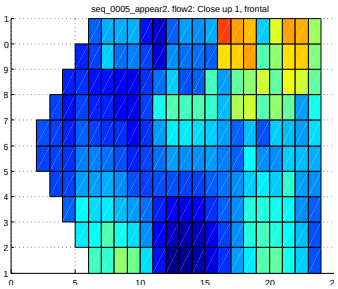
(a) *Viewpoint* v_1

(b) *Viewpoint* v_2

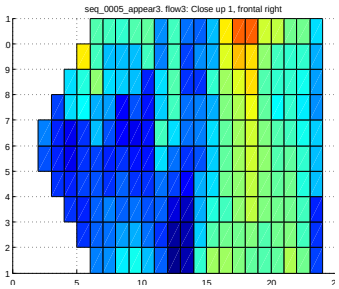
(c) *Viewpoint* v_3



(d) *Appearance cost trellis from* v_1



(e) *Appearance cost trellis from* v_2



(f) *Appearance cost trellis from* v_3

Figure 5.12: *Top row, left to right: 4D Video character* $I(t, w, v_i)$*,* $t = 0.6$*,* $w = \{0.5\}$ *from viewpoints* v_1, v_2 *and* v_3 *respectively. Using walk and jog sequences from dataset Dan. Bottom row: Appearance costs trellis computed for each of the view points using the appearance similarity measure from Equation 5.5. Notice how the candidates' appearance similarity changes depending on* v_i*.*

and $t_d = 0.1$, $w_d = \{0\}$. If the viewpoint changes to v_j, which shows a close-up view of the face as shown in Figure 5.13b, the result of the optimisation changes, because the appearance cost $E_a(t, v)$ is view-dependant. In this case, the optimal transition is found at $t_s = 0.28$. Notice that at this time-stamp the shape similarity is significantly lower, but this score is compensated by the high score in appearance similarity from that particular viewpoint v_j, leading to a different optimal transition point with respect the solution found at v_i.

Once all similarities are evaluated, the transition path P is optimised over a trellis of depth l_{max} with a discrete parameter sampling $\Delta w = \frac{w_{max} - w_{min}}{10}$ in the source and

(a) *Transition found between a walk/jog motion and a walk to stand motion from viewpoint v_i using 4DVT Motion Graphs. Left: transition pose found from walk/jog space ($t_s = 0.94$). Centre: target transition pose from walk to stand motion. Right: Optical flow between the two linked poses.*

(b) *Transition found between a walk/jog motion and a walk to stand motion from viewpoint v_j. Left: source transition pose in the walk/jog space ($t_s = 0.28$). Centre: target transition pose in the walk to stand motion. Right: Optical flow between the two linked poses.*

Figure 5.13: *Detail of the transitions generated using a 4DVT Motion Graph between a walk/jog motion and a walk to stand motion, from two different viewpoints. Top row shows the two linked frames found using the optimisation from Equation 5.4 at viewpoint v_i. Bottom row shows the transition found between the same two motions from viewpoint v_j. Notice that changes in viewpoint affect the result of the optimisation (different t value) performed by Equation 5.4 because the appearance cost $E_a(t, v)$ is view dependant.*

target motion space for all possible transition paths Ω using Dijkstra's shortest path algorithm. Despite the large number of potential transition paths the optimal path P_{opt} can be evaluated in real-time for $l_{max} < 10$ which is sufficient to produce visually seamless motions.

5.4 Results and Evaluation

Evaluation was performed over five 4D performance capture datasets captured under two studio configurations at 25 *fps*, as described in Section 2.2. The resulting interactive animations ran at $\sim$ 10 *fps* on a GeForce GTX 680 GPU equipped Pentium 2.5Ghz quad core PC.

Figures 5.14 to 5.23 show a variety of results generated using 4DVT, rendered every 10th frame for visualisation purposes. Animated versions of these figures as well as further video-results are available in the supplementary video of this thesis. Visual quality of the synthesised sequences is comparable to the rendering of captured free-viewpoint video sequences.

Figure 5.14 shows a long path interactively travelled using Dan character. Notice how the resulting animation seamlessly combines walk, jog, turn and jump motions. Similarly, Figure 5.15 shows an animation interactively generated using the Knight character, who wears a loose a metallic upper-body chainmail armour with a leather cape on top. In the last part of the animation the character switch from a walk motion into a jog motion, notice how the highly non-rigid surface of the cape is successfully reproduced by our 4DVT approach. Figure 5.16 presents character Dan jumping across benches with variable in-between distance. 4DVT rendering in combination with photo-realistic CG object results in plausible video-realistic animations. Infantry character animation is illustrated in Figures 5.17, Figures 5.18 and 5.19. Observe the seamless transitions from a walk/jog motion into a jump motion present in all Infantry figures.

In order to stress the realism of the results generated using the proposed 4DVT approach Figure 5.19 explicitly highlights synthesised poses using a semi-transparent green layer. Originally captured textures are marked in blue. No significant differences in visual quality are observed.

2D standard video footage can be combined with 4DVT to generate fully video-realistic real-world scenarios with interactive character animation. In Figure 5.20 Roxanne character performs a synthesised walk to jog motion in an outdoor background. Similarly, in Figures 5.21 and 5.22 character Dan is animated in outdoor scenes combining walk,

Figure 5.14: *Character Dan interactively animated using 4DVT Motion Graphs in a virtual scenario. Combines walk, jog, left turn, right turn, short jump and long jump captured motions.*

jog, turn, long jump and short jump motions. Notice the highly realistic video-texture, including hair and wrinkle details, generated using the proposed approach. Refer to the supplementary video of this thesis for further animated results, including multiple challenging 4DVT characters composited with hand-held recorded videos.

Finally, although it is not a main goal of this research, limited object interaction is also possible using our approach. Figure 5.23 shows character Dan grabbing boxes of variable width, which is set by the user. The character's parametrisation to reach the requested size is automatically found by solving $w_i = f^{-1}(p)$, where in this particular example p is the width of the box and w_i is the blending weight of the R motion class that controls the width of a grab motion.

5.4.1 User study

A cohort of 51 non-expert participants undertook a public web-based survey to evaluate aesthetic preference for videos and stills generated by the 4DVT framework. Participants were asked to compare 15 separate rendering pairs, each pair consisting in one 4DVT rendered model and one source FVVR captured model. Refer to Appendix C for a detailed list of the renders used in the survey.

Participants were asked to indicate their preference for one rendering or the other on a 5 point scale. The scale spanned strong preferences for either configuration (1 or 5),

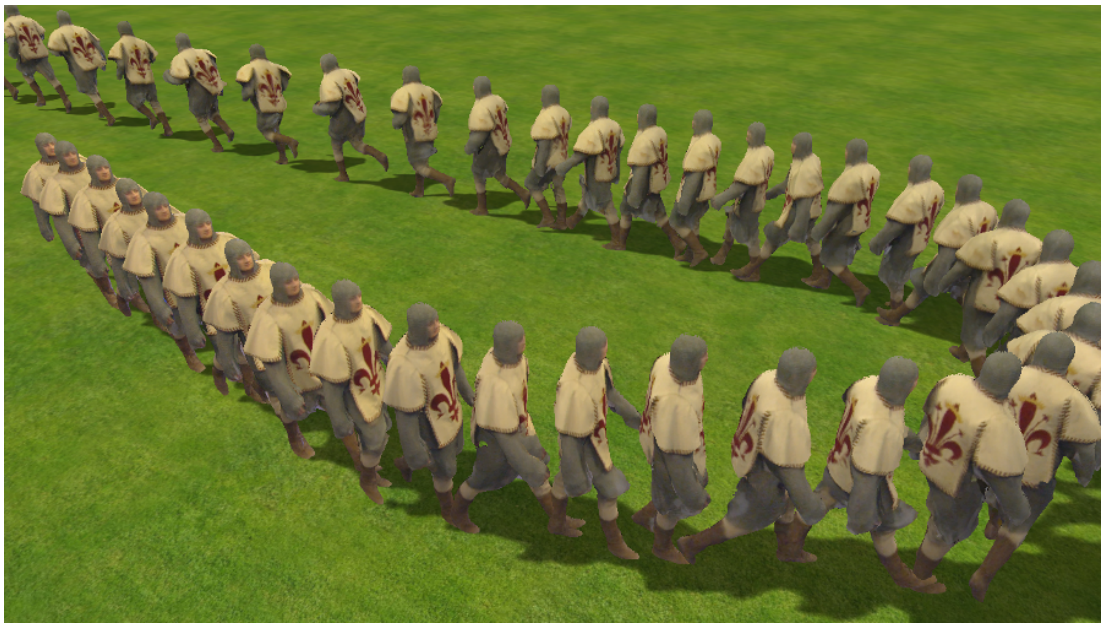


Figure 5.15: *Knight character animated using our 4DVT approach. Notice the significantly non-rigid clothing: metallic upper-body chainmail armour and a leather cape on top. 4DVT successfully reproduces complex non-rigid surface and appearance deformations.*

Figure 5.16: *Character Dan jumping across benches with variable in-between distance. Photo-realistic computer generated objects may be incorporated into the virtual scenario to increase the realism of the animation. Notice the highly detailed texturing, reproducing surface dynamics such as cloth wrinkling and hair movement.*

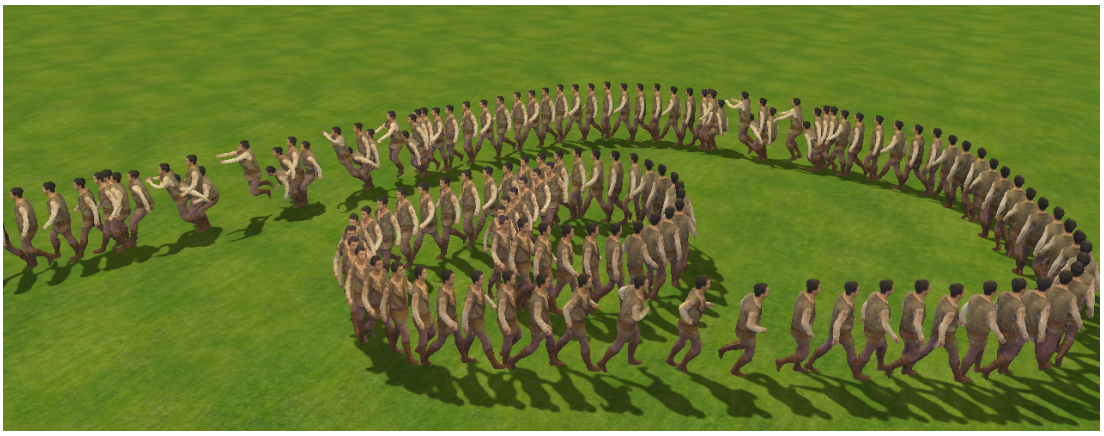


Figure 5.17: *Infantry character animated using 4DVT Motion Graph. Notice the different range of motions and transitions, including walk to jog motion, a range of different turn curvatures and varying jump lengths. 4DVT fully exploits 4D video databases to synthesise large novel range of motions.*

Figure 5.18: *Infantry character animated through 4DVT Motion Graphs jumping over obstacles of varying length. Notice the detail in face and clothing.*

Figure 5.19: *In order to stress the difference between captured 4D video models and 4DVT synthesised characters, a green semi-transparent layer is rendered on the top of 4DVT models in this figure. Blue is used to highlight the captured data. Notice how the difference in pose and appearance visual quality is hardly perceptible between captured and generated characters.*

Figure 5.20: *Character Roxanne animated performing a walk to jog motion in a virtual scenario created from a hand-held recorded video. The extraction of the camera calibration parameters of the captured 2D video enables the animation of 4DVT characters 'in' real video footage, obeying the original depth and scene properties.*

Figure 5.21: *Character Dan animated using 4DVT Motion Graphs, combining walk, jog, left turn, and a range of jumps. The extraction of the camera calibration parameters of the captured 2D video enables the animation of 4DVT characters 'in' real video footage, obeying the original depth and scene properties.*

Figure 5.22: *Character Dan animated using 4DVT Motion Graphs, combining walk, jog, left turn, and a range of jumps. Notice the appearance detail in face and and cloth wrinkling.*

Figure 5.23: *4DVT enables limited object interaction. In this example, character Dan is animated to grab a box of variable width, interactivelly controlled by the user.*

and no preference (3). Questions were put to participants in a randomised order, and identical pairs were also inserted as a control. For such pairs, we observed preferences of 3.15 ± 0.67 indicating natural variations in response.

We evaluated the perceived difference between real FVVR captured footage and synthesised 4D captured footage using 4DVT. Users were shown a real 4D captured character at a given time, alongside a character interpolated from that 4D footage using our proposed algorithm. The interpolated character approximately matched the captured one. Results were gathered for 15 sequences and are summarised per subject in Figure 5.24. On the scale, strong preference for real footage was indicated at 5, and 4D video textures at 1. Overall the output is judged to be approximately equal (3.3 ± 1.10) indicating the plausibility of the synthetic character. Interestingly characters Infantry and Knight are from datasets captured using studio setup B, and are at a lower resolutions than the other two performances captured in studio setup A. The increased noise in the optic flow field resulting from, and combined with, the lower resolution imagery from studio B datasets may have led to a greater preference for real imagery in these cases (3.60 ± 1.03). Nevertheless, the overall mean is approximately neutral (3.30) with a high standard deviation (1.10) indicating little perceivable difference between synthetic output and real imagery.

Finally we asked participants to indicate their opinion of character animation produced by our system on a scale 1-5, where very artificial was indicated at 1, and very realistic at 5. The overall score of 3.97 (± 0.69) further indicates a good level of aesthetic plausibility in our synthesised output.

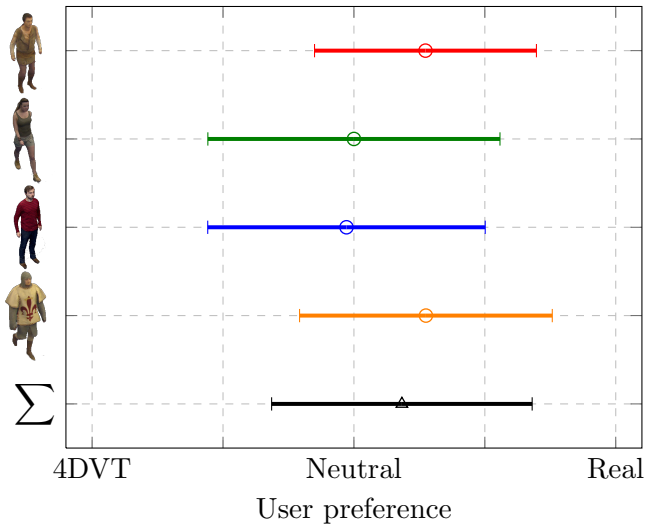


Figure 5.24: *Preference for 4DVT synthesis vs. Real Capture.*

5.4.2 Discussion and Limitation

Results and evaluation presented show that the proposed 4DVT rendering achieves visual quality of dynamic appearance similar to the captured video. Quality of the 4DVT renderings is dependent on a number of factors: high-res multiple view video capture without motion blur; 4D video reconstruction and robust temporal alignment; and accurate optical flow alignment in 4DVT rendering. Visual artefacts in either the captured video or 4D reconstruction will degrade the quality of the final rendering. Small residual errors from incorrect geometric reconstruction and alignment can be observed in the Knight and Infantry character video due to the relatively complex clothing deformations and lower resolution capture. Errors in the online optical flow alignment will result in blur or ghosting artefacts.

In addition to improvements in the 4D video reconstruction pipeline, our approach presents two main limitations. First, the computational overhead of the online optical flow. Although this is a performed in real-time using state-of-the art GPU implementations [opea], it limits the rendering to a single character and requires high-performance GPU. Pre-computation of the alignment is desirable, however, this is non-trivial due to the difference in character pose and camera views in each motion. Alignment computation in the 4DVT space is also non-trivial due to the discontinuities in texture.

A possible solution is pre-alignment in a set of canonical rendered views, however this may result in a loss of quality.

The second main limitation is the loading time and memory requirements for the multi-view camera capture source frames. Our approach requires the original HD imagery to be preloaded into the GPU memory in order to to compute 4DVT appearance at run time. Average data transfer timing from an optical hard drive to GPU memory for an HD image frame with PNG compression requires up to 0.25 *sec.*, Therefore the imagery of a 30 frames sequence captured in a 8 HD camera studio requires up to 60 seconds of loading time. Furthermore, current standard graphics cards are provided with up to 6GB memory, heavily limiting the amount of frames that our approach can handle. Methods for free-viewpoint texture compression are required to both speed-up loading time and reduce memory requirements.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

State-of-the-art methods for 3D surface reconstruction enable the representation of real-world captured motions using a temporally-coherent mesh structure that deforms over time [SH07b, VBMP08, dST*08, GSdA*09, BHKH13], referred to as 4D video. This thesis addresses the problem of exploiting 4D video datasets with the goal of rendering novel video-realistic character animations that maintain the visual-realism of the source videos and allow interactive control of the motion. In order to achieve this goal, a number of problems have been tackled through the research presented in this thesis.

Initial research focused on building a parametrised motion space by combining the underlying mesh sequence geometry from multiple related motions to obtain interactive control of the motion. Inspired by previous research on skeletal motion capture (MoCap) [RCB98], which synthesised novel skeletal data by interpolation of joint angles, methods for 3D mesh sequence synthesis based on pose blending were investigated. In particular, three problems are addressed in Chapter 3. First, sequence time-warping to align the key events across multiple motions. This avoids artefacts such as foot skating or unrealistic motions when blending different sequences. The second problem addressed is 3D mesh blending. Linear interpolation of vertex positions is demonstrated to be computationally efficient, but may result in unrealistic deformations or

mesh collapse if there are significant differences in shape. Non-linear mesh editing techniques based on differential representations that encode local surface shape enable plausible surface deformations, however these methods are too computationally expensive for real-time animation [Sor06, BS08]. Therefore, in order to achieve real-time performance with realistic deformations, a hybrid non-linear blending approach is introduced. The proposed method enables online approximation of the non-linear result by combining linear vertex blending and a set of precomputed non-linear displacement maps. Quantitative evaluation of the proposed hybrid approach shows an average vertex displacement error of online interpolated meshes 3 mm. A number of figures and animated video results (supplementary material) are included to qualitatively evaluate the interactive parametric animation control achieved using the proposed approach. These results show no perceptual difference between the originally reconstructed 4D motions and the synthesised parametric sequences, thus concluding that the proposed method maintains the realism of the source data. Finally, the third problem addressed in Chapter 3 is the need for high-level parametric control to provide the user with meaningful motion parameters, such as walking speed or length of the jump. This is solved by learning a mapping function from the high-level user-specified motion parameters to the corresponding blend weights required to generate the desired motion. This gives a closed-form non-linear function to interactively control the animation using high-level parameter. In summary, the combination of the methods proposed for the problems addressed in Chapter 3 (sequence time-warping, mesh blending and high-level parametric control) enable for the first time interactive control of the motion of a virtual character built from 4D performance capture sequences which are semantically similar in motion. This allows the construction of 4D parametric motion spaces at runtime. Resulting animations maintain the dynamic surface detail present in the source 4D video sequences and allow real-time interactive motion control.

The hybrid approach for mesh interpolation presented in Chapter 3 requires the source sequences to be similar in motion (for example, walk and jog) in order to achieve realistic poses. With the objective of relaxing this requirement, Chapter 4 introduces a representation and methods for 4D video geometry manipulation. Inspired by existing approaches for skeletal MoCap [KGP02, HG07] and 3D mesh [HHS09] sequence con-

catenation, a novel graph representation is introduced, referred to as a 4D Parametric Motion Graphs (4DPMG), that allows transition between 4D parametric spaces for multiple motion classes (i.e. walk and jump) by blending and concatenating captured sequences. Each node in a 4DPMG is a parametric space created by the combination of similar motions, as described in Chapter 3. The edges of the graph represent transitions between two parametric spaces. Shape and motion similarity across all input frames are computed and stored off-line in a matrix **S**. This enables online transitions between any pair of parametrised meshes, ensuring both fast response (low transition latency) and smooth transitions. Realistic online interpolation and seamless concatenation of 4D video mesh sequences enable 4DPMG to achieve for the first time interactive and continuous motion control of a virtual character built from 4D performance capture. This is demonstrated by quantitatively evaluating the latency and shape similarity of a large set of randomly generated transitions. Qualitative evaluation is illustrated in a number of figures depicting different animations interactively created using 4DPMG. The supplementary video includes a number of online synthesised animations that combine walk, jog, different styles of jumps, reach and grab for 4 different characters. These results demonstrate that the animations synthesised through 4DPMG maintain the dynamic surface details of the originally reconstructed data. The proposed framework overcomes the limitations of previous methods for 3D mesh animation, which were based on sequence concatenation and do not allow interactive control of the motion [SMH05, HHS09], by enabling both the construction of parametric spaces of meshes and the computation of transitions between them at runtime.

Chapter 5 addresses the remaining challenge to achieve video-realistic rendering. As the character motion is interactively controlled by combining multiple 4D videos, direct rendering from the multi-view source videos will result in incorrect appearance dynamics and visual artefacts such as blur and ghosting when images from multiple motions are combined. Therefore, an approach is required to combine appearance from multiple 4D videos to render dynamic appearance matching the character motion. A novel method for free-viewpoint appearance synthesis is introduced, referred to as 4D Video Textures (4DVT), which combines the source 4D videos to synthesise novel view-dependant textures that change the dynamic appearance as the underlying

parametrised motion changes. Exploiting the known vertex-to-vertex correspondence across multiple sequences, the parametrised 4D geometric proxy is first textured using view-dependant rendering for each of the source multi-view videos. Optical flow is computed between the rendered images for the desired view and used to align and warp the dynamic appearance from 4D videos of different motions to obtain high-quality visual output which matches the parametrically controlled motion. Finally, the transition cost function introduced in the 4DPMG representation in Chapter 4 is extended by the addition of an appearance term. This helps to find better transition candidates by considering not only latency and shape similarity, but also appearance similarity. A user study is performed to evaluate our approach for free-viewpoint texture alignment. 51 individuals were asked to compare and score a set of pairs of renders, each of them containing an original 4D video frame and a synthesised 4DVT frame, depending on their preference. Results confirmed that no significant perceptual difference existed between source free-viewpoint rendering and synthesised 4DVT. Chapter 5 successfully provides the last piece of the puzzle to enable interactive video-realistic character animation from 4D performance capture. The proposed method for motion synthesis and viewpoint-dependant texture alignment advances the state-of-the-art for video-realistic character animation by enabling real-time interactive control of character motion with video-realistic rendering of dynamic appearance.

In summary, this thesis demonstrates the synthesis of video-realistic character animation by the combination and reutilisation of 4D video data. Results from chapters 3 and 4 demonstrate that a plausible geometric 3D-mesh proxy of the motion can be generated by the online interpolation of captured mesh sequences. Novel techniques for both mesh parametrisation and concatenation of parametric sequences have been introduced. These techniques overcome limitations of previous approaches for character animation, allowing for first time interactive flexible control of a virtual character built from multi-camera capture. Furthermore, this work demonstrates that realistic dynamic character appearance that matches a parametric mesh model can be provided by the combination of synchronised multi-view imagery. The proposed method for multi-view parametric texture alignment, referred to as 4D Video Textures, enables for the first time the synthesis of video-realistic interactive characters built from 4D perfor-

mance capture datasets. The results presented in this thesis represent a step forward in the area of character animation, enabling interactive video-realistic rendering. Media production industries, such as TV, film and video-gaming, could potentially benefit from the methods presented in this thesis to generate digital doubles.

6.2 Future Work

The work presented in this thesis successfully achieved our initial goal: the creation of video-realistic interactive character animation. However, a number of open problems remain for future research.

Firstly, it is important to highlight that the quality of the presented results is limited by the quality of the 4D video performance capture. Currently this includes visual artefacts due to errors in reconstruction and temporal mesh sequence alignment. Improvements in 4D video capture would result in improved quality in the rendered animations.

As discussed in Section 5.4.2, the main limitations of the 4D Video Textures approach are related to the size of the input image data, which has a significant negative impact in both the loading time and memory requirements of our framework. The current implementation requires to pre-load all source multi-camera imagery (consisting in 8 to 10 HD images per frame, at 25 *fps*, in tested datasets) into GPU memory to enable real-time texture retrieval for optical flow and texture synthesis computation. Recent research [VH13] has investigated a novel layered representation for free-viewpoint rendering that helps in alleviating the current memory overhead. This representation has been combined with the 4DVT approach presented in this thesis, successfully improving the performance of the framework. At the time of writing this document, a paper describing this work is under submission. Furthermore, free-viewpoint rendering has recently been demonstrated on mobile platforms [IVG*13], bringing video-realistic animations into low-performance devices such as mobile phones and tablets. Future research could investigate the incorporation of these novel methods into the presented framework to reduce the current requirement for high-performance machinery.

An off-the-shelf OpenCV [opea] optical flow implementation that exploits GPU compu-

tational capabilities using NVIDIA CUDA [cud] is used to achieve online performance of the 4DVT approach for texture alignment. This optical flow is the computational bottleneck at runtime, restricting the current framework to a single character. In order to relax the limitation, future research could investigate methods for optical flow computation speed up or offline alignment of the appearance across multiple motions to remove the need for online optical flow computation.

To fully exploit the video-realistic characters synthesised using the proposed 4DVT approach, this thesis has presented a number of rendered results composited with standard 2D video footage. However, our approach does not currently incorporate any colour correction or relighting step to manipulate the final illumination properties of the texture, leading to possible unrealistic compositions due to differences between the scene illumination and the captured character. Recent research on relightable human performances [LWS*13] could be used to extend our proposed 4DVT pipeline to address this limitation. Similarly, reflectance estimation could be also incorporated.

Robust object interaction remains as an open problem. Limited object interaction is demonstrated in this thesis, but it is still far from providing a reliable system. Future research could investigate methods for 3D-mesh character interaction with both synthetic objects and 4D video objects. Similarly, multi-character interaction can be also investigated.

Appendix A

Proof Equation 4.5

This appendix provides a proof of the result defined in Equation 4.5. Let us consider two parametrised meshes M^s and M^d with respective vectors of vertex coordinates $X^s = \sum_i w_i^s X_i^s$ and $X^d = \sum_j w_j^d X_j^d$ where X_i^s,X_j^d denote the vectors of vertex coordinates for the temporally aligned input meshes M_i^s, M_j^d in each parametrised space and w_i^s, w_j^d denote their blending proportions. For conciseness and without loss of generality, we omit the time component and assume weights have been normalised such that $\sum_i w_i = \sum_j w_j = 1$.

Lemma Let us define a function d measuring the distance between pairs of meshes. All meshes having being preliminary temporally aligned to guarantee a similar number of vertices and connectivity, d can be defined as the squared Euclidean distance between mesh vertices, namely:

$$d(M_i, M_j) = \|X_i - X_j\|^2. \tag{A.1}$$

Note that for simplicity, we did not normalise by the number of vertices which remains constant across all mesh sequences after alignment. The distance between any two meshes M^s and M^d must satisfy the following property:

$$d(M^s, M^d) \leq \sum_i \sum_j w_i^s w_j^d d(M_i^s, M_j^d). \tag{A.2}$$

Proof We start by expanding each term of Equation A.2. We have:

$$
\begin{aligned}
d(M^s, M^d) &= \|X^s - X^d\|^2, && \text{(A.3)}\\
&= \|\sum_i w_i^s X_i^s - \sum_j w_j^d X_j^d\|^2, && \text{(A.4)}\\
&= \|\sum_i w_i^s X_i^s\|^2 - 2(\sum_i w_i^s X_i^s)\cdot(\sum_j w_j^d X_j^d) + \|\sum_j w_j^d X_j^d\|^2, && \text{(A.5)}\\
&= \|\sum_i w_i^s X_i^s\|^2 - 2\sum_i\sum_j w_i^s w_j^d X_i^s \cdot X_j^d + \|\sum_j w_j^d X_j^d\|^2, && \text{(A.6)}
\end{aligned}
$$

and

$$
\begin{aligned}
\sum_i\sum_j w_i^s w_j^d d(M_i^s, M_j^d) &= \sum_i\sum_j w_i^s w_j^d \|X_i^s - X_j^d\|^2, && \text{(A.7)}\\
&= \sum_i\sum_j w_i^s w_j^d \left(\|X_i^s\|^2 - 2X_i^s\cdot X_j^d + \|X_j^d\|^2\right), && \text{(A.8)}\\
&= \sum_i\sum_j w_i^s w_j^d \|X_i^s\|^2 - 2\sum_i\sum_j w_i^s w_j^d X_i^s\cdot X_j^d + \sum_i\sum_j w_i^s w_j^d \|X_j^d\|^2, && \text{(A.9)}\\
&= \sum_j w_j^d \sum_i w_i^s \|X_i^s\|^2 - 2\sum_i\sum_j w_i^s w_j^d X_i^s\cdot X_j^d + \sum_i w_i^s \sum_j w_j^d \|X_j^d\|^2, && \text{(A.10)}\\
&= \sum_i w_i^s \|X_i^s\|^2 - 2\sum_i\sum_j w_i^s w_j^d X_i^s\cdot X_j^d + \sum_j w_j^d \|X_j^d\|^2. && \text{(A.11)}
\end{aligned}
$$

The middle term on the right hand-side of Equations A.6 and A.11 remains identical. In order to prove Equation (A.2) it is therefore sufficient to show that the following two inequalities hold:

$$\|\sum_i w_i^s X_i^s\|^2 \leq \sum_i w_i^s \|X_i^s\|^2, \tag{A.12}$$

and

$$\|\sum_j w_j^d X_j^d\|^2 \leq \sum_j w_j^d \|X_j^d\|^2. \tag{A.13}$$

We carry out the proof for Equation A.12, the proof for Equation A.13 being identical. We start by writing explicitly the squared norm of the vector of vertex coordinates as a function of individual mesh coordinates:

$$\|\sum_i w_i^s X_i^s\|^2 = \sum_k (\sum_i w_i^s X_i^s(k))^2. \tag{A.14}$$

Noting that w_i^s are by definition non negative, the previous equation can be written in the form:

$$\|\sum_i w_i^s X_i^s\|^2 = \sum_k (\sum_i \sqrt{w_i^s}(\sqrt{w_i^s} X_i^s(k)))^2. \tag{A.15}$$

Applying Cauchy-Schwarz inequality, we then obtain:

$$\begin{aligned} (\sum_i \sqrt{w_i^s}(\sqrt{w_i^s} X_i^s(k)))^2 &\leq (\sum_i w_i^s)(\sum_i w_i^s X_i^s(k)^2), && \text{(A.16)} \\ &= \sum_i w_i^s X_i^s(k)^2, && \text{(A.17)} \end{aligned}$$

and it follows from Equation A.15 that:

$$\begin{aligned} \|\sum_i w_i^s X_i^s\|^2 &\leq \sum_k \sum_i w_i^s X_i^s(k)^2, && \text{(A.18)} \\ &= \sum_i w_i^s (\sum_k X_i^s(k)^2), && \text{(A.19)} \\ &= \sum_i w_i^s \|X_i^s\|^2. && \text{(A.20)} \end{aligned}$$

This completes the proof. *square*

Result Let us now define a function s measuring the similarity between pairs of meshes as a function of their distance d previously introduced:

$$s(M_i, M_j) = 1 - \frac{d(M_i, M_j)}{\max(d(M_i, M_j))}. \tag{A.21}$$

The similarity between any two meshes M^s and M^d must satisfy the following property:

$$s(M^s, M^d) \geq \sum_i \sum_j w_i^s w_j^d s(M_i^s, M_j^d). \tag{A.22}$$

This provides a sufficient condition to identify similar meshes between two parametrised space based on the similarity between pairs of input meshes defining each parametrised space.

Proof From Equation A.2 we have:

$$1 - \frac{d(M^s, M^d)}{\max(d(M_i, M_j))} \geq 1 - \frac{\sum_i \sum_j w_i^s w_j^d d(M_i^s, M_j^d)}{\max(d(M_i, M_j))}, \tag{A.23}$$

$$= (\sum_i w_i^s)(\sum_j w_j^d) - \sum_i \sum_j w_i^s w_j^d \frac{d(M_i^s, M_j^d)}{\max(d(M_i, M_j))}, \tag{A.24}$$

$$= \sum_i \sum_j w_i^s w_j^d - \sum_i \sum_j w_i^s w_j^d \frac{d(M_i^s, M_j^d)}{\max(d(M_i, M_j))}, \tag{A.25}$$

$$= \sum_i \sum_j w_i^s w_j^d \left(1 - \frac{d(M_i^s, M_j^d)}{\max(d(M_i, M_j))}\right). \tag{A.26}$$

From the definition of s, it follows that:

$$s(M^s, M^d) \geq \sum_i \sum_j w_i^s w_j^d s(M_i^s, M_j^d). \tag{A.27}$$

This completes the proof. □

It should be noted that the result was stated in the case of shape similarity only. The extension to shape and motion similarity is trivial since these are both expressed in terms of a Euclidean distance between pairs of N_v-dimensional vectors.

Appendix B

4DPMG OpenGL Environment

A C++ OpenGL [opeb] environment illustrated in Figure B.2 was built in order to implement the 4D Parametric Motion Graph (4DPMG) approach presented Chapter 4. The application allows the user to load any 4DPMG defined in a customised XML format. Once the data is loaded, the user can interactively control the character and check the current configuration through an intuitive OpenGL graphic interface.

B.1 XML file format

The application loads the motion graph information such as number of nodes and sequences embedded in each node from a customised XML file, using the following tags:

<nodes>

Indicates that a new 4DPMG is created, and nodes will be defined.

<node>

Indicates that a new node in the current 4DPMG is created.

<sequences>

Defines the sequences used to create the current node.

<sequence>

Defines a new sequence, pointing to the path where meshes are stored.

<offsets>

Indicates that the offsets of the sequences will be read.

<offset>

Defines path of the text file containing the local offsets of each mesh of the current sequence. Used to locate the virtual character in the scenario.

<rotations>

Indicates that files containing the root rotations of each sequence will be read.

<rotation>

Defines the path to the text file containing the rotations of each mesh of the current sequence. If the motion being read is a straight motion with no rotations, it can be indicated with the keyword *NO*.

An example of an actual XML file used in this work is shown in Figure B.1. In this particular case, a simple two-node 4DPMG is defined, each node containing two sequences with no root rotations in any of them. Notice how the proposed XML format defines both the nodes and the sequences in each of them, but not the transition points, which are found online depending on the current state of the graph, using the approach introduced in Section 4.4.

B.2 C++ OpenGL GUI

An OpenGL [opeb] application shown in Figure B.2 has been developed in this work to provide full animation control to the user. The user can interactively control the character using the keyboard, and also visualise the current state of the 4DPMG, Figure B.3a, and the current parametric state of the node, Figure B.3b. A menu bar located on the top of the main window allows the configuration of visualisation options such as camera location, lighting, shadows, etc. A second menu bar, located in the bottom of the main screen, prints out live information regarding the application state, such as

```
<?xml version="1.0"?>
<nodes>
  <node>
    <sequences>
      <sequence>/media/seq_14_24_43_walk/</sequence>
      <sequence>/media/seq_14_38_07_jog2/</sequence>
    </sequences>
    <offsets>
      <offset>/media/seq_14_24_43_walk/offsets.txt</offset>
      <offset>/media/seq_14_38_07_jog2/offsets.txt</offset>
    </offsets>
    <rotations>
      <rotation>NO</rotation>
      <rotation>NO</rotation>
    </rotations>
  </node>
  <node>
    <sequences>
      <sequence>/media/seq_15_36_08_shortJump/</sequence>
      <sequence>/media/seq_15_37_34_longJump/</sequence>
    </sequences>
    <offsets>
      <offset>/media/seq_15_36_08_shortJumpoffsets.txt</offset>
      <offset>/media/seq_15_37_34_longJump/offsets.txt</offset>
    </offsets>
    <rotations>
      <rotation>NO</rotation>
      <rotation>NO</rotation>
    </rotations>
  </node>
</nodes>
```

Figure B.1: *Sample of the XML file format used to load a 4DPMG into the C++ OpenGL application.*

frame rate or latest options enabled. On the right-hand side of the window, a number of extra options including save current animation, store current mesh, show previous meshes, show travelled path and show wireframe can be enabled through checkboxes and radial buttons.

The proposed OpenGL interface also incorporates input fields in which the user can introduce parameters that are required for the 4DPMG approach. These options, all previously discussed in Section 4.4, include depth of trellis, destination point in the target node, parameters Δt and $\Delta \mathbf{w}$, among others.

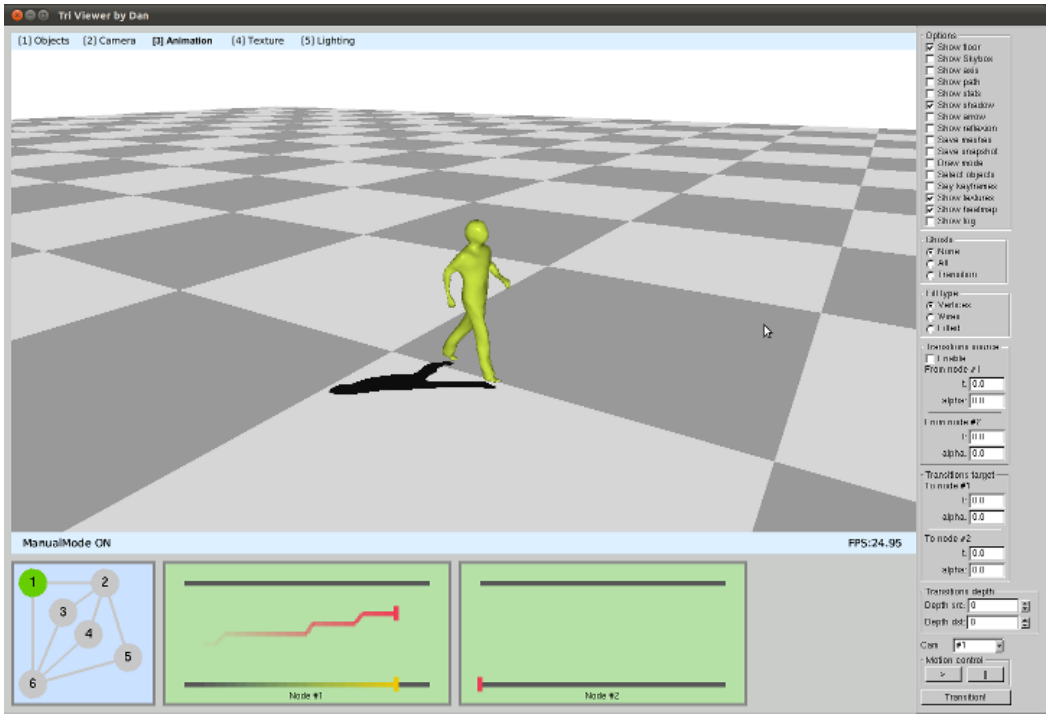
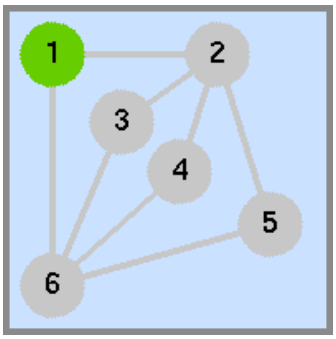
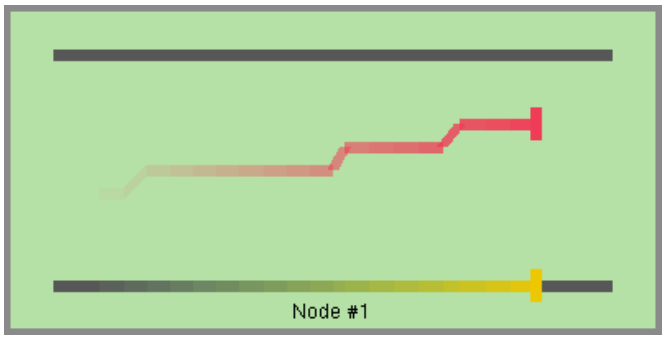


Figure B.2: *C++ OpenGL application implemented to test the approach introduced in this chapter. On top, the main 3D scenario where the character is rendered. On bottom-left, the current 4DPMG, shown in detail in Figure B.3a. Bottom-centre, in green, diagrams depicting the current parametric state of the node, shown in detail in Figure B.3b.*



(a) *4DPMG current state visualisation. The active node is highlighted in green.*



(b) *Visualisation of the current state of the parametric space. Red and yellow lines depicts the current and previous configurations for all parameters w_i.*

Figure B.3: *Diagrams used to visualise the state of the 4DPMG in real time.*

Appendix C

4DVT User Study

A user study was conducted in order to evaluate the user perception of the results of the 4D Video Textures (4DVT) approach introduced in this thesis. A cohort of 51 non-expert participants undertook a public web-based survey to evaluate aesthetic preference for videos and stills generated by the 4DVT framework. Participants were asked to compare 15 separate rendering pairs, each pair consisting in one 4DVT rendered model and one source FVVR captured model. User had to rate each pair selecting one of the following options:

1. Strongly prefer the left hand image.
2. Prefer the left hand image.
3. Neutral / no preference.
4. Prefer the right hand image.
5. Strongly prefer the right hand image.

Figures C.1 to C.6 illustrate examples of the render pairs evaluated by the users. Results of the user study are discussed in Section 5.4.1.

Figure C.1: *User study pair #1*

Figure C.2: *User study pair #2*

Figure C.3: *User study pair #3*

Figure C.4: *User study pair #4*

Figure C.5: *User study pair #5*

Figure C.6: *User study pair #6*

Bibliography

[ACOL00] Alexa M., Cohen-Or D., Levin D.: As-rigid-as-possible shape interpolation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), SIGGRAPH '00, pp. 157–164.

[ACP02] Allen B., Curless B., Popović Z.: Articulated body deformation from range scan data. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 21*, 3 (July 2002), 612–619.

[ACP03] Allen B., Curless B., Popović Z.: The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 22*, 3 (July 2003), 587–594.

[AF02] Arikan O., Forsyth D. A.: Interactive motion generation from examples. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 21*, 3 (2002), 483–490.

[AFO03] Arikan O., Forsyth D. A., O'Brien J. F.: Motion synthesis from annotations. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 22*, 3 (July 2003), 402–408.

[AM00] Alexa M., Müller W.: Representing animations by principal components. *Computer Graphics Forum 19*, 3 (2000), 411–418.

[AMH01] Ahmed A., Mokhtarian F., Hilton A.: Parametric motion blend-

ing through wavelet analysis. In *Eurographics 01. Proceedings of Short Presentations* (2001), pp. 347–353.

[ATR*08] Ahmed N., Theobalt C., Rossl C., Thrun S., Seidel H. P.: Dense correspondence finding for parametrization-free animation reconstruction from video. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (2008), pp. 1–8.

[BBK07] Bronstein A., Bronstein M., Kimmel R.: Calculus of Nonrigid Surfaces for Geometry and Texture Manipulation. *IEEE Transactions on Visualization and Computer Graphics 13*, 5 (2007), 902–913.

[BBM*01] Buehler C., Bosse M., McMillan L., Gortler S., Cohen M.: Unstructured lumigraph rendering. In *Proceedings of SIGGRAPH 2001* (New York, NY, USA, 2001), ACM, pp. 425–432.

[BBPP10] Ballan L., Brostow G. J., Puwein J., Pollefeys M.: Unstructured video-based rendering: interactive exploration of casually captured videos. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 29* (July 2010), 87:1–87:11.

[BBPW04] Brox T., Bruhn A., Papenberg N., Weickert J.: High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)* (May 2004), Springer, pp. 25–36.

[BCS97] Bregler C., Covell M., Slaney M.: Video Rewrite: Driving Visual Speech with Audio. In *Proceedings of ACM SIGGRAPH* (1997), pp. 1—8.

[BCvdPP08] Beaudoin P., Coros S., van de Panne M., Poulin P.: Motion-motif graphs. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), SCA '08, Eurographics Association, pp. 117–126.

[BHH11] Budd C., Huang P., Hilton A.: Hierarchical Shape Matching for Temporally Consistent 3D Video. In *Proceedings of International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)* (Hangzhou, China, 2011).

[BHKH13] Budd C., Huang P., Klaudiny M., Hilton A.: Global non-rigid alignment of surface sequences. *International Journal of Computer Vision 102*, 1-3 (Mar. 2013), 256–270.

[BM98] Bregler C., Malik J.: Tracking people with twists and exponential maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 1998), IEEE Computer Society, pp. 8–.

[Bow00] Bowden R.: Learning statistical models of human motion. In *IEEE Workshop on Human Modelling, Analysis and Synthesis, CVPR* (2000).

[BS08] Botsch M., Sorkine O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (2008), 213—230.

[BW95] Brundelin A., Williams L.: Motion signal processing. In *Proceedings of ACM SIGGRAPH* (1995), pp. 97–104.

[CBI09] Cagniart C., Boyer E., Ilic S.: Iterative mesh deformation for dense surface tracking. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)* (2009), pp. 1465–1472.

[CBI10a] Cagniart C., Boyer E., Ilic S.: Free-Form Mesh Tracking: a Patch-Based Approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010), pp. 1339–1346.

[CBI10b] Cagniart C., Boyer E., Ilic S.: Probabilistic deformable surface tracking from multiple videos. In *Proceedings of the 11th European conference on Computer Vision* (2010), ECCV'10, pp. 326–339.

[CBK05] Cheung G. K. M., Baker S., Kanade T.: Shape-from-silhouette across time part ii: Applications to human modeling and markerless motion tracking. *International Journal of Computer Vision 63*, 3 (2005), 225–245.

[CHL01] Chong K., Han Y., Lam T.: Concurrent threads and optimal parallel minimum spanning trees algorithm. *Journal of the ACM 48*, 2 (2001), 297—323.

[CKBH00] Cheung G. K., Kanade T., Bouguet J.-Y., Holler M.: A real time system for robust 3d voxel reconstruction of human motions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2000), vol. 2, IEEE, pp. 714–720.

[CTGH11] Casas D., Tejera M., Guillemaut J.-Y., Hilton A.: Parametric Control of Captured Mesh Sequences for Real-Time Animation. In *Proceedings of the 4th International Conference on Motion In Games (MIG)* (2011), vol. 7060 of *Lecture Notes in Computer Science*, Springer, pp. 242–253.

[CTGH12a] Casas D., Tejera M., Guillemaut J.-Y., Hilton A.: 4D Parametric Motion Graphs for Interactive Animation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2012), I3D '12, pp. 103–110.

[CTGH12b] Casas D., Tejera M., Guillemaut J.-Y., Hilton A.: Parametric Animation of Performance-Captured Mesh Sequences. *Journal of Visualization and Computer Animation 23*, 2 (2012), 101–111.

[CTGH13] Casas D., Tejera M., Guillemaut J.-Y., Hilton A.: Interactive Animation of 4D Performance Capture. *IEEE Transactions on Visualization and Computer Graphics 19*, 5 (2013), 762–773.

[CTMS03] Carranza J., Theobalt C., Magnor M., Seidel H.-P.: Free-Viewpoint Video of Human Actors. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 22*, 3 (2003), 569–577.

[cud] NVIDIA CUDA (Compute Unified Device Architecture). http://www.nvidia.com/cuda.

[CW93] Chen S. E., Williams L.: View interpolation for image synthesis. In *Proceedings of ACM SIGGRAPH* (1993), pp. 279–288.

[CYJ02] Cobzas D., Yerex K., Jgersand M.: Dynamic textures for image-based rendering of fine-scale 3d structure and animation of non-rigid motion. In *Proceedings of Eurographics* (2002), pp. 1067–7055.

[CZ04] Celly B., Zordan V. B.: Animated People Textures. In *Proceedings of International Conference on Computer Animation and Social Agents (CASA)* (2004).

[dATM*04] de Aguiar E., Theobalt C., Magnor M., Theisel H., Seidel H. P.: M3: marker-free model reconstruction and motion tracking from 3d voxel data. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on* (2004), pp. 101–110.

[DBR00] Deutscher J., Blake A., Reid I. D.: Articulated Body Motion Capture by Annealed Particle Filtering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR* (2000), IEEE Computer Society Press, pp. 2126–2133.

[dST*08] de Aguiar E., Stoll C., Theobalt C., Ahmed N., Seidel H.-P., Thrun S.: Performance Capture from Sparse Multi-view Video. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 27*, 3 (2008), 98:1–98:10.

[DTM96] Debevec P. E., Taylor C. J., Malik J.: Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proceedings of ACM SIGGRAPH* (1996), pp. 11–20.

[ECJ*06] Einarsson P., Chabert C.-F., Jones A., Ma W.-C., Lamond B., Hawkins T., Bolas M., Sylwan S., Debevec P.: Relighting human

locomotion with flowed reflectance fields. In *Proceedings of Eurographics Symposium on Rendering (2006)* (June 2006), pp. 183–194.

[EDDM*08] Eisemann M., De Decker B., Magnor M., Bekaert P., de Aguiar E., Ahmed N., Theobalt C., Sellent A.: Floating textures. *Computer Graphics Forum (Proceedings of Eurographics) 27*, 2 (Apr. 2008), 409–418.

[ESH*12] Elhayek A., Stoll C., Hasler N., Kim K.-I., Seidel H.-P., Theobalt C.: Spatio-temporal motion tracking with unsynchronized cameras. In *IEEE Computer Vision and Pattern Recognition (CVPR)* (Rhode Island, USA, June 2012), IEEE, pp. 1870–1877.

[Far03] Farnebäck G.: Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis* (Gothenburg, Sweden, June-July 2003), LNCS 2749, pp. 363–370.

[FNZ*09] Flagg M., Nakazawa A., Zhang Q., Kang S.-B., Ryu Y., Essa I., Rehg J.: Human Video Textures. In *ACM / SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2009), pp. 199–206.

[GD96] Gavrila D., Davis L. S.: 3-d model-based tracking of humans in action: a multi-view approach. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (1996), pp. 73–80.

[GGSC96] Gortler S. J., Grzeszczuk R., Szeliski R., Cohen M. F.: The lumigraph. In *Proceedings of the SIGGRAPH* (New York, NY, USA, 1996), ACM, pp. 43–54.

[Gle99] Gleicher M.: Animation from observation: Motion capture and motion editing. *Computer Graphics 33*, 4 (Nov. 1999), 51–54.

[GSdA*09] Gall J., Stoll C., de Aguiar E., Theobalt C., Rosenhahn B., Seidel H. P.: Motion capture using joint skeleton tracking and sur-

face estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR* (2009), pp. 1746–1753.

[HBH11] Huang P., Budd C., Hilton A.: Global temporal registration of multiple non-rigid surface sequences. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2011), pp. 3473–3480.

[HDK07] Hornung A., Dekkers E., Kobbelt L.: Character animation from 2d pictures and 3d motion data. *ACM Transactions on Graphics 26*, 1 (2007).

[HFE13] Hilsmann A., Fechteler P., Eisert P.: Pose space image based rendering. *Computer Graphics Forum (Proceedings of Eurographics) 32*, 2 (2013), 265–274.

[HFP*00] Herda L., Fua P., Plänkers R., Boulic R., Thalmann D.: Skeleton-based motion capture for robust reconstruction of human motion. In *Proceedings of the Computer Animation* (Washington, DC, USA, 2000), CA '00, IEEE Computer Society, pp. 77–.

[HG07] Heck R., Gleicher M.: Parametric Motion Graphs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (2007), I3D '07, pp. 129–136.

[HH09] Huang P., Hilton A.: Surface Motion Graphs for Character Animation from 3D Video. In *SIGGRAPH 2009: Talks* (2009), pp. 56:1–56:1.

[HHS09] Huang P., Hilton A., Starck J.: Human Motion Synthesis from 3D Video. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 1478 – 1485.

[HHS10a] Huang P., Hilton A., Starck J.: Shape Similarity for 3D Video Sequences of People. *International Journal of Computer Vision 89*, 2-3 (September 2010), 362–381.

[HHS10b] Huang P., Hilton A., Starck J.: Shape similarity for 3d video sequences of people. *Int. J. Comput. Vision 89* (Sept. 2010), 362–381.

[Hog83] Hogg D.: Model-based vision: a program to see a walking person. *Image and Vision Computing 1*, 1 (1983), 5 – 20.

[HRT*09] Hasler N., Rosenhahn B., Thormahlen T., Wand M., Gall J., Seidel H. P.: Markerless motion capture with unsynchronized moving cameras. In *IEEE Computer Vision and Pattern Recognition (CVPR)* (2009), pp. 224–231.

[HSL*06] Huang J., Shi X., Liu X., Zhou K., Wei L.-Y., Teng S.-H., Bao H., Guo B., Shum H.-Y.: Subspace gradient domain mesh deformation. *ACM Transactions on Graphics (Proceedings of ACM SIGGCAT) 25*, 3 (2006), 1126–1134.

[IF04] Ikemoto L., Forsyth D. A.: Enriching a motion collection by transplanting limbs. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), SCA '04, Eurographics Association, pp. 99–108.

[IVG*13] Imber J., Volino M., Guillemaut J.-Y., Fenney S., Hilton A.: Free-viewpoint video rendering for mobile devices. In *Proceedings of the 6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications* (2013), MIRAGE '13, pp. 11:1–11:8.

[KG04] Kovar L., Gleicher M.: Automated Extraction and Parameterization of Motions in Large Date Sets. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 23*, 3 (2004), 559—568.

[KG08] Kircher S., Garland M.: Free-Form Motion Processing. *ACM Transactions on Graphics 27*, 2 (2008), 1–13.

[KGP02] Kovar L., Gleicher M., Pighin F.: Motion Graphs. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 21*, 3 (2002), 473–482.

[KR97] Kanade T., Rander P.: Virtualized Reality: Constructing Virtual Worlds from Real Scenes. *IEEE MultiMedia 4*, 2 (1997), 34—47.

[KTT*12] Kim K. I., Tompkin J., Theobald M., Kautz J., Theobalt C.: Match graph construction for large image databases. In *Proceedings of European Conference on Computer Vision (ECCV)* (2012), Springer, pp. 272–285.

[Lau94] Laurentini A.: The visual hull concept for silhouette-based image understanding. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 16*, 2 (1994), 150–162.

[LCF00] Lewis J. P., Cordner M., Fong N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), SIGGRAPH '00, pp. 165–172.

[LCR*02] Lee J., Chai J., Reitsma P. S. A., Hodgins J. K., Pollard N. S.: Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 21*, 3 (July 2002), 491–500.

[LLB*10] Lipski C., Linz C., Berger K., Sellent A., Magnor M.: Virtual Video Camera: Image-Based Viewpoint Navigation Through Space and Time. *Computer Graphics Forum 29*, 8 (2010), 2555–2568.

[Lok01] Lok B.: Online model reconstruction for interactive virtual environments. In *Proceedings of ACM / SIGGRAPH Symposium on Interactive 3D Graphics* (2001), I3D '01, pp. 69–72.

[Low04] Lowe D.: Distinctive image features for scale invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91—110.

[LSLCO05] Lipman Y., Sorkine O., Levin D., Cohen-Or D.: Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 24*, 3 (July 2005), 479–487.

[LvdP96] Lamouret A., van de Panne M.: Motion synthesis by example. In *Eurographics International Workshop on Computer Animation and Simulation EGCAS,* (1996), Springer Verlag, pp. 199–212.

[LWS*13] Li G., Wu C., Stoll C., Liu Y., Varanasi K., Dai Q., Theobalt C.: Capturing relightable human performances under general uncontrolled illumination. In *Computer Graphics Forum (Proccedings of EUROGRAPHICS)* (2013), vol. 32, pp. 275–284.

[MBC01] Mizuguchi M., Buchanan J., Calvert T.: Data driven motion transitions for interactive games. In *Eurographics 2001 Short Presentations* (2001), vol. 2, p. 6.

[MBR*00] Matusik W., Buehler C., Raskar R., Gortler S. J., McMillan L.: Image-based visual hulls. In *Proceedings ACM SIGGRAPH* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 369–374.

[MC12] Min J., Chai J.: Motion graphs++: a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA) 31*, 6 (2012).

[Men99] Menache A.: *Understanding Motion Capture for Computer Animation and Video Games*, 1st ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[MG01] Moeslund T. B., Granum E.: A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding 81*, 3 (Mar. 2001), 231–268.

[MH03] Mitchelson J., Hilton A.: Hierarchical tracking of multiple people. In *British Machine Vision Conference* (2003).

[MHK06] Moeslund T. B., Hilton A., Krüger V.: A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding 104*, 2-3 (2006), 90–126.

[MK05] Mukai T., Kuriyama S.: Geostatistical Motion Interpolation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 24*, 3 (2005), 1062–1070.

[MTG97] Moezzi S., Tai L.-C., Gerard P.: Virtual view generation for 3D digital video. *MultiMedia, IEEE 4*, 1 (1997), 18–26.

[MTH00] Molina-Tanco L., Hilton A.: Realistic synthesis of novel human movements from a database of motion capture examples. In *In Workshop on Human Motion (HUMO)* (2000), pp. 137–142.

[opea] OpenCV (Open Computer Vision Library). http://www.opencv.org.

[opeb] OpenGL (Open Graphics Library). http://opengl.org.

[PB02] Pullen K., Bregler C.: Motion capture assisted animation: texturing and synthesis. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 21*, 3 (July 2002), 501–508.

[Per95] Perlin K.: Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics 1*, 1 (1995).

[PG96] Perlin K., Goldberg A.: Improv: a system for scripting interactive actors in virtual worlds. In *Proceedings of SIGGRAPH* (New York, NY, USA, 1996), ACM, pp. 205–216.

[Pop07] Poppe R.: Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding 108*, 1-2 (Oct. 2007), 4–18.

[Pot87] Potmesil M.: Generating octree models of 3d objects from their silhouettes in a sequence of images. *Journal of Computer Vision, Graphics and Image Processing 40*, 1 (Oct. 1987), 1–29.

[PSS02] Park S. I., Shin H. J., Shin S. Y.: On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2002), SCA '02, ACM, pp. 105–111.

[RCB98] Rose C., Cohen M., Bodenheimer B.: Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5 (1998), 32–40.

[RISC01] Rose III C. F., Sloan P.-P. J., Cohen M. F.: Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum (Proceedings of Eurographics) 20*, 3 (2001), 239–250.

[Roh93] Rohr K.: Incremental recognition of pedestrians from image sequences. In *Proceedings of Computer Vision and Pattern Recognition, CVPR* (1993), pp. 8–13.

[SCD*06] Seitz S., Curless B., Diebel J., Scharstein D., Szeliski R.: A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2006), pp. 519—528.

[SD99] Seitz S., Dyer C.: Photorealistic Scene Reconstruction by Voxel Coloring. *International Journal of Computer Vision 35*, 2 (1999), 151–173.

[SE02] Schödl A., Essa I. A.: Controlled animation of video sprites. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), pp. 121–127.

[SH05] Starck J., Hilton A.: Spherical matching for temporal correspondence of non-rigid surfaces. In *Proceedings of IEEE International Conference on Computer Vision* (2005), vol. 2, pp. 1387–1394.

[SH07a] Starck J., Hilton A.: Correspondence labelling for wide-timeframe free-form surface matching. In *Proceedings of IEEE International Conference on Computer Vision* (2007), pp. 1–8.

[SH07b] Starck J., Hilton A.: Surface Capture for Performance Based Animation. *IEEE Computer Graphics and Applications 27(3)* (2007), 21–31.

[SLAM08] Stich T., Linz C., Albuquerque G., Magnor M.: View and time

interpolation in image space. *Computer Graphics Forum (Proceedings of Pacific Graphics) 27*, 7 (2008), 1781–1787.

[SMH05] Starck J., Miller G., Hilton A.: Video-Based Character Animation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 49–58.

[SMN*09] Starck J., Maki A., Nobuhara S., Hilton A., Matsuyama T.: The multiple-camera 3-d production studio. *IEEE Transactions on Circuits and Systems for Video Technology 19*, 6 (2009), 856–869.

[SO06] Shin H. J., Oh H. S.: Fat graphs: constructing an interactive character with continuous controls. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), pp. 291–298.

[Sor06] Sorkine O.: Differential Representations for Mesh Processing. *Computer Graphics Forum 25*, 4 (2006), 789–807.

[SP04] Sumner R. W., Popović J.: Deformation transfer for triangle meshes. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 23*, 3 (2004), 399–405.

[SRC01] Sloan P.-P. J., Rose III C. F., Cohen M. F.: Shape by example. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (2001), I3D '01, pp. 135–143.

[SSE00] Schodl A., Szeliski R. amd Salesin D., Essa I.: Video Textures. In *Proceedings of ACM SIGGRAPH* (2000), pp. 489—498.

[SZGP05] Sumner R. W., Zwicker M., Gotsman C., Popović J.: Mesh-based inverse kinematics. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 24*, 3 (July 2005), 488–495.

[TH11] Tejera M., Hilton A.: Space-time Editing of 3D Video Sequences. In *Conference for Visual Media Production (CVMP)* (2011), pp. 148–157.

[TKKT12] Tompkin J., Kim K. I., Kautz J., Theobalt C.: Videoscapes: exploring sparse, unstructured video collections. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 31*, 4 (July 2012), 68:1–68:12.

[TM10] Tung T., Matsuyama T.: Dynamic Surface Matching by Geodesic Mapping for Animation Transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010), pp. 1402–1409.

[TMSS02] Theobalt C., Magnor M., Schuler P., Seidel H. P.: Combining 2D feature tracking and volume reconstruction for online video-based human motion capture. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications* (2002), pp. 96–103.

[VBK05] Vedula S., Baker S., Kanade T.: Image-based spatio-temporal modeling and view interpolation of dynamic events. *ACM Transactions on Graphics 24*, 2 (Apr. 2005), 240–261.

[VBMP08] Vlasic D., Baran I., Matusik W., Popović J.: Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 27*, 3 (2008), 97:1–97:9.

[VBR*05] Vedula S., Baker S., Rander P., Collins R., Kanade T.: Three-Dimensional Scene Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 3 (2005), 475–480.

[VH13] Volino M., Hilton A.: Layered view-dependent texture maps. In *Proceedings of the 10th Conference on Visual Media Production (CVMP)* (London, 2013).

[WH97] Wiley D., Hahn J.: Interpolation Synthesis for Articulated Figure Motion. In *IEEE Virtual Reality International Symposium* (1997), pp. 157—160.

[WP95] Witkin A., Popovic Z.: Motion Warping. In *Proceedings of ACM SIGGRAPH* (1995), pp. 105–108.

[XLS*11] Xu F., Liu Y., Stoll C., Tompkin J., Bharaj G., Dai Q., Seidel H.-P., Kautz J., Theobalt C.: Video-based Characters - Creating New Human Performances from a Multi-view Video Database. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 30*, 4 (July 2011), 32:1–32:10.

[XYA06] Xu J., Yamasaki T., Aizawa K.: Motion editing in 3d video database. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission, 3DPVT* (2006), IEEE, pp. 472–479.

[XZY*07] Xu W., Zhou K., Yu Y., Peng Q., Guo B.: Gradient domain editing of deforming mesh sequences. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 26*, 3 (2007), 84.

[ZKU*04] Zitnick C., Kang S., Uyttendaele M., Winder S., Szeliski R.: High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 23*, 3 (Aug. 2004), 600–608.

[ZS08] Zhao L., Safonova A.: Achieving good connectivity in motion graphs. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation* (July 2008), pp. 127–136.