A FRAMEWORK FOR IMAGE-BASED ASSET GENERATION AND ANIMATION

Johannes Furch[1], *Anna Hilsmann*[1,2], *Peter Eisert*[1,2] *

[1]Fraunhofer HHI, [2]Humboldt Universität zu Berlin

ABSTRACT

Creating digital animatable models of real-world objects and characters is important for many applications, ranging from highly expensive movie productions to low-cost real-time applications like computer games and augmented reality. However, achieving real photorealism with convincing appearance and deformation behavior requires sophisticated capturing, elaborate manual modeling and time-consuming simulation. This can only be achieved in well funded film productions, while in low-cost applications, animated objects usually lack visual quality. In this paper, we present a new framework for image-based animatable asset generation which avoids these time-consuming processes both in the modeling and the simulation stage. Real-time photo-realistic animation is enabled by the use of captured images and shifting computational complexity to an a-priori training phase. Our paper covers the complete pipeline of content creation, asset generation and representation, and a real-time animation and rendering implementation.

Index Terms— Real-Time, Image-Based Rendering, Animation, Asset Generation, Post-Production

1. INTRODUCTION

Digital animatable replications of real world objects and characters are ubiquitous in many applications, such as film, video games, online conference systems or augmented reality. All of those use-cases aim for photorealism and natural deformation behavior during animation. However, this requirement contradicts with both the ease of modeling and rendering efficiency. Modeling often relies on elaborate manual artistic tweaking of material and animation parameters of 3D computer graphics models. Animation requires complex physics-based simulation of shape deformation (e.g. muscle bulging or cloth wrinkling). Finally, rendering involves costly lighting and reflection simulation. Although this way of modeling and rendering has been established and successfully applied over many years, it also has its drawbacks. Achieving photorealism is still very time-consuming, especially if complex animations and materials are involved. Therefore, convincing results can only be achieved in expensive productions, where long rendering times and expensive manpower can be afforded. In low-cost productions or real-time applications, virtual objects and characters often lack visual quality.

In the past decade, (semi-) automatic methods have been proposed to create digital models directly from real-word objects. We limit the following review of related work to the example of human actors and clothes, as they represent an important class of digital animatable objects for many applications.
Detailed and time consistent 3D models of real human actors can be reconstructed from video sequences using performance capture approaches [1, 2, 3]. However, these approaches do not yet yield animatable characters with realistic shape deformation properties. Interactive control of character movement from motion capture data has been addressed by motion graphs, which smoothly concatenate recorded motion sequences [4, 5, 6]. Other approaches learn shape deformations in the form of vertex displacements directly from examples to reconstruct animatable representations of real objects [7, 8, 9, 10]. All these methods create digital models from real world objects and characters. However, they require very sophisticated capturing setups and in many cases manual editing. Also, they mainly focus on a realistic modeling of shape and deformation but do not account for complex appearance changes during animation. Existing approaches to create satisfying models with realistic reflection properties require even more elaborate setups including light stages [11, 12, 13], so that these methods are mainly used for expensive Hollywood productions.

In contrast to computer graphics models, real camera images naturally provide photo-realistic replications of real-world objects. However, camera images do not offer the same flexibility as animatable computer graphics models as they do not allow performance manipulation and animation. This has recently been addressed in the computer graphics community by image-based animatable models [14, 15]. In these approaches, the model representation consists of real image data showing an object with different animation parameters (e.g. skeleton configurations) and optionally additional meta information extracted from the image data. During animation, the best matching sample images are selected, modified to fit the input animation parameters, and blended into the final result. This approach has several advantages: Since the model is represented by real image data, photorealism comes naturally if no errors are introduced by the image modifications. Content creation can be performed with minimal user intervention and no artistic skills. Rendering costs are low as rendering mainly involves image warping. However, these methods have only recently emerged in the computer graphics literature and have not yet found their way into real (consumer) applications or movie post-productions. In this paper, we present a complete framework for image-based asset generation and animation, which allows straightforward integration into those applications. We base our framework on the theoretical work of [15]. Our main contribution lies in the real-time realization and generalization of this method for a broader class of animatable objects. Furthermore, we show how it can be used for asset generation and integration into the post-production workflow as well as consumer applications.

In the following, Sec. 2 summarizes the image-based rendering approach of [15] and describes our asset generation pipeline from capturing over data extraction and representation to data storage for real-time rendering. Sec. 3 focuses on real-time animation and rendering using this representation. Finally, Sec. 4 presents results and example applications.

*This work is partially funded by the German Science Foundation, DFG EI524/2-1 and by the European Commission, FP7-288238 SCENE and H2020-644629 AutoPost.

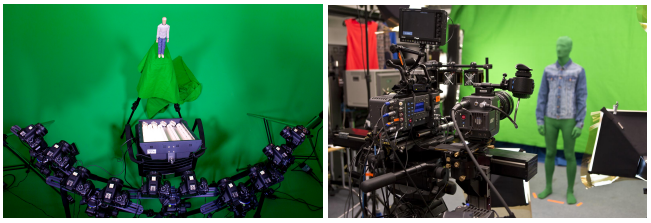

Fig. 1. Synchronized multi-view high-resolution still camera setup and stereo movie camera setup with integrated depth sensor.

2. IMAGE-BASED ANIMATABLE ASSETS

Our image-based assets are constructed from a set of images showing a real object captured under varying conditions and a set of parameters describing these variations, e.g. viewing directions and skeletal poses. Other variations are possible, as long as they can be described by a finite number of parameters. During animation, a set of samples is selected based on parameter similarity to a render input. As the sparsely sampled images will rarely ever exactly fit the render input parameters, the image data has to be modified accordingly by geometric warping and color adjustment. Fast low-resolution pure geometric animation methods (e.g. skeleton subspace deformation, SSD) are used to guide the warping of the images. These methods are simple and fast but do not model characteristic shape deformations and appearance changes of real-world objects, like muscle bulging or cloth wrinkling. This is compensated for by applying spatio-intensity image warps interpolated from a-priorily extracted warps between the sample images. The interpolation is performed in the parameter space of the captured variations. Hence, our image-based asset generation pipeline consists of the following steps: capturing of sample images, extracting meta information (e.g. skeletal pose, warps, ...), and bundling all data into an asset representation for animation and rendering.

2.1. Capturing

Different setups are possible for capturing the image samples, ranging from multi-view synchronized high resolution DSLR cameras to stereo or single view video cameras (Fig. 1). Each setup has its (dis-) advantages depending on the application. Multi-view high resolution still camera setups are comparably cheap and can be used to capture a large number of high resolution views for each single skeletal pose configuration. However, this requires determining the sampling frequency during capturing. Therefore, the absence of significant samples can only be detected in later steps, which might lead to inferior animation results. Using video cameras instead, the parameter space can be densely sampled and an automatic selection of significant samples can be performed during meta data extraction. The advantage of this approach is that the final representation can be augmented by adding samples in any stage of the processing chain. However, high quality movie cameras are very expensive, making it very costly to capture a large number of views. Moreover, even high-end movie cameras still do not meet the resolution of consumer DSLR cameras.

Note that while we limit the variable conditions to camera views and skeletal poses here, the proposed implementations are generic and can be extended to the needs of the application (e.g. by introducing a change in lighting conditions). To ease the subsequent data extraction, the images are captured in front of a green screen that can be removed from the captured sequence using keying methods included in industry standard movie production tools. The output of capturing stage is a multi-view multi-pose dataset of images and associated alpha mattes (Fig. 2).

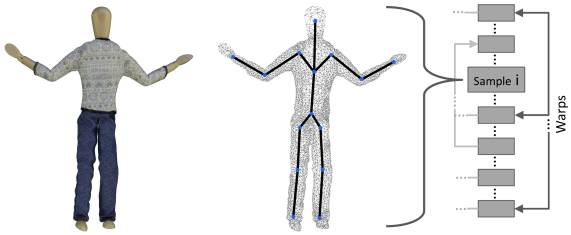

Fig. 2. Illustration of the asset representation: each sample consists of the captured image, an alpha matte, a camera, a 3D-mesh, an animation configuration and image warps to other samples in the database.

2.2. Data Representation and Extraction

The following meta information is extracted from the captured sample images and consolidated into the final asset representation (Fig. 2).

Calibration. The cameras are calibrated prior to capturing, using specifically designed calibration charts [16].

Depth. For coarse animation of the image data, 3D-meshes have to be generated. Depending on the capture setup, 3D point clouds are created either by triangulation of reliable sparse key-point correspondences [17] or by projection of the depth channel of a RGBZ device [18, 19]. 2D meshes are generated for the area covered by the alpha masks, and the 3D positions of the vertices are interpolated from the point cloud.

Animation Parameters. To represent the skeletal pose configuration for each sample image, a skeleton-model of articulated bones is fit to the mesh, e.g. using [20, 21]. For a consistent representation of camera positions and skeletal poses, we place the world coordinate system in the torso joint. Skinning weights for SSD-animation are calculated based on [22]. To simplify the data transfer to the graphics hardware, we limit the number of influencing bones for each vertex to four.

Proximity Graphs. A proximity graph is built connecting samples lying close together in the high-dimensional parameter space (view/skeletal pose). To save storage, a prime objective of the asset creation process is to keep the samples as significant as possible and their number as low as possible. To decrease the amount of required samples, the parameter space can be split into multiple sub-spaces (e.g. limbs of a body) that represent independent regions in the captured images. A proximity sub-graph is constructed for each sub-space. During rendering, the sub-animations are seamlessly blended [15].

Appearance Changes. Pure geometric animation (SSD) cannot model real world deformations and appearance changes. Following [15], sample appearance changes are stored as joint spatio-intensity image warps for each connection in the proximity graph [23]. This results in two displacement values and a photometric change parameter for each mesh vertex.

Data Storage. The image-based asset data is saved in a generic sample-based format that can be extended by additional parameters if desired. Optimized image data (e.g. compressed or downscaled) can be stored for different rendering applications. Meshes and memory-demanding parameters are best saved in binary files to both optimize the amount of required storage space and loading time.

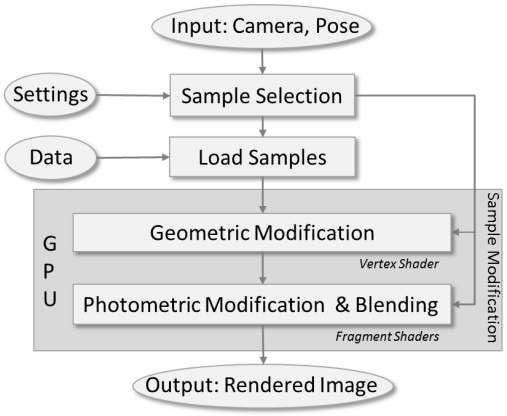

Fig. 3. The image-based asset animation and rendering pipeline.

3. REAL-TIME ASSET ANIMATION AND RENDERING

The input to the renderer consists of parameters for camera view and skeletal pose in the same representation as for the sample images and for conventional character animation methods. This ensures a smooth integration of our approach into established post-production frameworks. Based on the input parameters, a set of samples is selected from the database and processed to synthesize a new image. Configurable settings are the number of samples (s) to be modified and blended into the final result and the number of warps per sample (w) to be used for warp interpolation. In the following, we will introduce our implementation of a real-time renderer following the pipeline illustrated in Fig. 3.

Sample Selection. First, a subset of the samples is selected based on their distances to the input in the parameter space as well as the w smallest distances to their neighbors in the proximity graph (see Fig. 4, a and e). This ensures the selection of samples visually similar to the input configuration and the availability of suitable warps for interpolation. Based on the distances in parameter space, blending and warp interpolation weights are determined (for details see [15]). For each contributing sample, rigid bone transformations (SSD animations) to the input pose are calculated.

Sample Data Loading. Since the sample modification is performed on the graphics hardware, all image (texture) and mesh data has to be transferred to graphics memory. Especially for real-time applications and low cost hardware, this bottleneck becomes relevant. Theoretically, all sample data can be loaded in advance and buffered in graphics memory. However, the data might exceed the available resources. To tackle this, prefetching data can be limited to the required samples if the sequence is known in advance. For more general real-time applications we propose two data scheduling heuristics: (i) Assuming that currently used sample data will be needed again to render subsequent frames, it is maintained in graphics memory for a certain number of frames. (ii) Assuming that samples parametrically close to those recently loaded will be needed to render subsequent frames, these samples are prefetched as soon as resources are available. This could also be adapted to only include probable future candidates based on a prediction of parameter changes. The amount of data to be loaded is determined by the setting parameters s and w as well as the resolution and compression of the stored image and mesh data. All of those variables can be adapted to the application requirements and specific devices.

Sample Modification. The image data of the selected samples is modified to fit the input view and skeletal pose by applying the geometric model transformations and the interpolated image warps (see Fig. 4, b-d). Those steps constitute the most calculation intensive part of the animation and are performed on the GPU. For each contributing sample, a processing pipeline consisting of a vertex shader and a fragment shader is executed. The resulting warped images are blended into a texture framebuffer. The following uniforms and attributes are handed to the shaders for processing:

All Samples / Uniform	
Input Camera	3x4 matrix
Per Sample / Uniform	
Image + Alpha Matte	RGBA texture sampler
Sample Camera	3x4 matrix
Bone Transformations	Vector of 3x4 matrices
Blending Weight	Scalar
Warp Weights	Vector
Per Vertex / Attributes	
Position (from 3D Mesh)	3D vector
Skinning Weights	4D vector
Skinning Bone Indices	4D vector
Geo and Photo Warps	(w) 3D vectors
(Sub-Skeleton Blending Weight)	Scalar

Geometric warping is performed in a *vertex shader*. The sample camera is used to project each vertex to the original image position to recover the texture coordinates. Next, the vertex is moved by SSD-animation: the transformations of the four closest bones are combined linearly based on its skinning weights and applied to the vertex. The input camera is applied to project the transformed vertex to the 2D image plane of the render frame. An additional 2D vertex offset is interpolated from the sample's image warps (based on the warp weights) and added to the projected vertex. Based on the same weights, a photometric warp is interpolated and handed to the fragment shader for photometric correction.

In the *fragment shader*, the pixel color is retrieved from the sample image. The color value is then multiplied by the photometric warp parameter, the sample blending weight and optionally the sub-skeleton blending weight and finally added to the texture framebuffer. By design, most alpha values in the final texture framebuffer sum up to 1 in the interior of the object and to values between 0 and 1 in partially transparent regions at the border. However, since there is no consistent mesh for each sample, and sub-skeleton weights are calculated per sample, the corresponding blending weights of the modified samples might not always exactly sum up to 1. We address this by normalizing the pixel color by the inverse of the alpha channel value. To avoid the extension of object borders we set the pixel to 0 if the alpha channel value lies below a threshold. Those steps are performed by a separate *fragment shader* and the result is finally written to the main framebuffer.

4. RESULTS AND APPLICATIONS

We illustrate the overall workflow of our image-based asset generation and animation framework on two datasets, a puppet and a jeans jacket. The puppet was captured in 110 different skeletal poses (Fig. 2) and from 10 different views using the pre-calibrated multi-view still-camera setup shown in Fig. 1 on the left. The jacket was captured using a side-by-side movie camera setup (Fig. 1 right). From this data, 48 image pairs of significant skeletal poses were extracted. The digital assets were tested in a post-production workflow

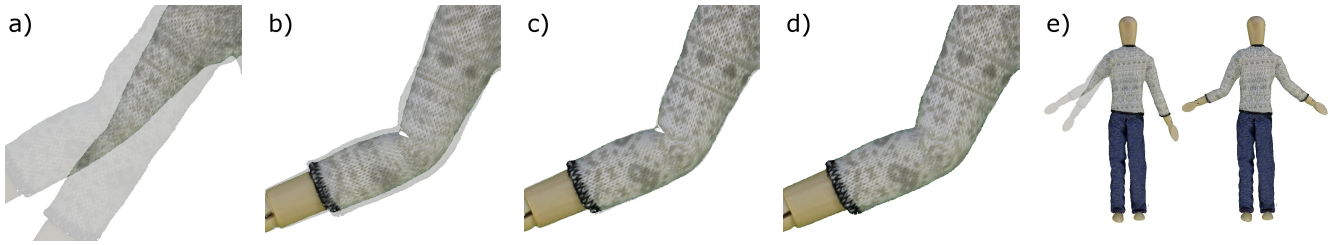

Fig. 4. Details of the different animation steps: a) selected samples, b)-d) sample modification steps: b) animation; c) warps; d) blending correction; e) full selected samples and final result.

Fig. 5. The full NUKE post-production pipeline illustrated on a shot for the shortfilm "Lady in red" (left to right): raw footage; samples selected for the extracted pose skeleton; final output of our renderer; final result, fine tuned using standard compositing tools.

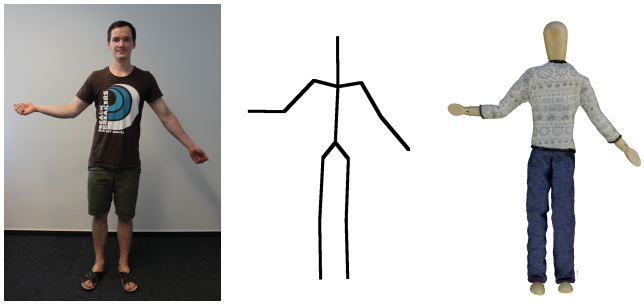

Fig. 6. A user controlling the motion of our doll character.

for movie production as well as in real-time applications using a Microsoft Kinect [18] for motion-capturing.

For movie production industries, it is important to integrate new technological developments into existing workflows. Therefore, we developed an image-based asset animation rendering plugin for the industry standard compositing software NUKE [24]. Due to very long rendering times for conventional animation, designated software (like Maya [25]) commonly provides ready-made animation sequences as layered render passes for compositing. The efficiency of our image-based asset animation approach allows closer integration of those two post-production steps. This follows a recent trend in the movie production industry to include 3D functionalities into compositing software. Tools like ncam [26] take the integration even further by providing computer graphics (CG) content previews during shooting. Our real-time rendering approach can be facilitated for this purpose when combined with a motion picture RGBZ camera [19]. In contrast to CG previews, image-based characters can already be included in the final photorealistic quality at this early production stage. The VFX breakdown in Fig. 5 illustrates how the close integration of our plugin and established compositing tools can be used to produce a seamless clothing overlay. All of the processing steps were performed within NUKE, while a clean plate was used for the removal of the background clothing.

Real-time photorealistic animatable asset rendering is also useful for many other applications such as online conference systems (e.g. [27]), computer games (e.g. [28]) and virtual mirrors (e.g. [29]). For online conference systems and computer games, the avatar can be a replication of the user or any other precaptured character asset (e.g. the doll in Fig. 6). The character asset can be arranged inside a virtual space and observed from arbitrary viewpoints, while the simple parametrization of the animation input drastically reduces the amount of data that has to be transmitted between the users as compared to video streams in conventional online conference systems. Emerging augmented-reality computer games like "Night Terrors" [28] composite precaptured videostreams to achieve photorealism despite hardware limitations. However, this approach limits the assets to precaptured content. For such applications, our asset generation and real-time animation rendering approach can provide more flexibility in character design and behavior while retaining the simpicity of content creation by facilitating the same devices already used for capturing. Virtual mirror applications provide a result simmilar to the post-production example illustrated above. For such applications, a simple game controller like the Microsoft Kinect can be used to provide the animation input. Fig. 6 shows a person controlling the motion of our virtual doll character facilitating the Kinect and the real-time renderer introduced above at 30 fps.

5. CONCLUSION

We presented a full framework for image-based asset creation as well as real-time animation and rendering. By using real image data and shifting computational complexity to an a-priori training phase, photo-realistic real-time animation is reduced to low-cost geometric animation, warp interpolation and image warping. The input data is the same as for conventional animation methods, allowing a direct integration into established frameworks. This paper presents a proof of concept to show how traditional computer graphics (CG) animation can efficiently be combined with image-based rendering (IBR) to exploit the benefits of both worlds. The general concept can be shifted more to either the CG or IBR side. For example, in the presented approach only a per-view surface of the character is captured for each sample. Therefore, self-occluding body parts cannot be modified without unnaturally distorting others. This could be resolved by exploiting a consistent 3D representation.

6. REFERENCES

[1] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur, "Markerless garment capture," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 99:1–99:9, Aug. 2008.

[2] Daniel Vlasic, Pieter Peers, Ilya Baran, Paul Debevec, Jovan Popović, Szymon Rusinkiewicz, and Wojciech Matusik, "Dynamic shape capture using multi-view photometric stereo," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 174:1–174:11, 2009.

[3] Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W. Sumner, and Markus Gross, "High-quality passive facial performance capture using anchor frames," *ACM Trans. Graph.*, vol. 30, pp. 75:1–75:10, August 2011.

[4] Okan Arikan and D. A. Forsyth, "Interactive motion generation from examples," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 483–490, 2002.

[5] Lucas Kovar, Michael Gleicher, and Frédéric Pighin, "Motion graphs," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 473–482, 2002.

[6] Dan Casas, Margara Tejera, Jean-Yves Guillemaut, and Adrian Hilton, "4d parametric motion graphs for interactive animation," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2012, pp. 103–110.

[7] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis, "Scape: Shape completion and animation of people," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 408–416, 2005.

[8] Brett Allen, Brian Curless, Zoran Popović, and Aaron Hertzmann, "Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis," in *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2006, pp. 147–156.

[9] Carsten Stoll, Juergen Gall, Edilson de Aguiar, Sebastian Thrun, and Christian Theobalt, "Video-based reconstruction of animatable human characters," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 139:1–139:10, 2010.

[10] T. Neumann, K. Varanasi, N. Hasler, M. Wacker, M. Magnor, and C. Theobalt, "Capture and statistical modeling of arm-muscle deformations," *Comput. Graph. Forum (Proc. Eurographics)*, vol. 32, no. 2, pp. 285–294, 2013.

[11] O. Alexander, M. Rogers, W. Lambeth, M. Chiang, and P. Debevec, "Creating a photoreal digital actor: The digital emily project," in *Visual Media Production, 2009. CVMP '09. Conference for*, 2009, pp. 176–187.

[12] Paul Debevec, "The light stages and their applications to photoreal digital actors," in *SIGGRAPH Asia*, Singapore, 2012.

[13] Oleg Alexander, Graham Fyffe, Jay Busch, Xueming Yu, Ryosuke Ichikari, Andrew Jones, Paul Debevec, Jorge Jimenez, Etienne Danvoye, Bernardo Antionazzi, Mike Eheler, Zybnek Kysela, and Javier von der Pahlen, "Digital ira: Creating a real-time photoreal digital actor," in *ACM SIGGRAPH 2013 Posters*, 2013.

[14] F. Xu, Y. Liu, C. Stoll, J. Tompkin, G. Bharaj, Q. Dai, H.-P. Seidel, J. Kautz, and C. Theobalt, "Video-based characters - creating new human performances from a multi-view video database," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 30, no. 4, pp. 32:1–32:10, 2011.

[15] A. Hilsmann, P. Fechteler, and P. Eisert, "Pose space image based rendering," *Computer Graphics Forum*, vol. 32, no. 2, pp. 265–274, 2013.

[16] P. Eisert, "Model-based camera calibration using analysis by synthesis techniques.," in *Proc. International Workshop on Vision, Modeling, and Visualization*, 2002, pp. 307–314.

[17] J. Furch and P. Eisert, "An iterative method for improving feature matches," in *International Conference on 3D Vision - 3DV*, June 2013, pp. 406–413.

[18] Barak Freedman, Alexander Shpunt, Meir Machline, and Yoel Arieli, "Depth mapping using projected patterns," 2009.

[19] Thomas Hach and Johannes Steurer, "A novel rgb-z camera for high-quality motion picture applications," in *Proc. of the 10th European Conference on Visual Media Production*, 2013.

[20] Philipp Fechteler, Anna Hilsmann, and Peter Eisert, "Kinematic icp for articulated template fitting," in *Proceedings of the 17th International Workshop on Vision, Modeling and Visualization (VMV2012)*, Magdeburg, Germany, 2012.

[21] Pushmeet Kohli and Jamie Shotton, *Key Developments in Human Pose Estimation for Kinect*, Springer, 2013.

[22] Ilya Baran and Jovan Popović, "Automatic rigging and animation of 3d characters," *ACM Trans. Graph.*, vol. 26, no. 3, 2007.

[23] A. Hilsmann and P. Eisert, "Joint estimation of deformable motion and photometric parameters in single view video," in *ICCV NORDIA Workshop*, 2009, pp. 390–397.

[24] The Foundry, "NUKE," www.thefoundry.co.uk/products/nuke, 2015.

[25] Autodesk, "Maya: Comprehensive 3D animation software," www.autodesk.com/products/maya, 2015.

[26] Ncam Technologies Limited, "ncam," www.ncam-tech.com, 2015.

[27] Philipp Fechteler, Anna Hilsmann, Peter Eisert, Sigurd Van Broeck, Christoph Stevens, Julie Wall, Michele Sanna, Davide A. Mauro, Fons Kuijk, Rufael Mekuria, Pablo Cesar, David Monaghan, Noel E. O'Connor, Petros Daras, Dimitrios Alexiadis, and Theodore Zahariadis, "A Framework for Realistic 3D Tele-Immersion," in *Proceedings of the 6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications (MIRAGE2013)*, Berlin, Germany, 6th-7th June 2013.

[28] Novum Analytics, "Night Terrors - Augmented Reality Game," http://www.novumanalytics.com/, 2015.

[29] A. Hilsmann and P. Eisert, "Tracking and retexturing cloth for real-time virtual clothing applications," in *Mirage 2009 - Computer Vision/Computer Graphics Collaboration Techniques and Applications*, Rocquencourt, France, May 2009.