Spatio-Temporal Image-Based Texture Atlases for Dynamic 3-D Models

Zsolt Jankó[1] Jean-Philippe Pons[2]
janko@imagine.enpc.fr jean-philippe.pons@cstb.fr

[1] IMAGINE Université Paris-Est and INRIA Rhône-Alpes / LJK, Grenoble, France
[2] IMAGINE Université Paris-Est, CSTB, Sophia-Antipolis, France

Abstract

In this paper, we propose a method for creating a high-quality spatio-temporal texture atlas from a dynamic 3-D model and a set of calibrated video sequences. By adopting an actual spatio-temporal perspective, beyond independent frame-by-frame computations, we fully exploit the very high redundancy in the input video sequences. First, we drastically cut down on the amount of texture data, and thereby we greatly enhance the portability and the rendering efficiency of the model. Second, we gather the numerous different viewpoint/time appearances of the scene, so as to recover from low resolution, grazing views, highlights, shadows and occlusions which affect some regions of the spatio-temporal model. Altogether, our method allows the synthesis of novel views from a small quantity of texture data, with an optimal visual quality throughout the sequence, with minimally visible color discontinuities, and without flickering artifacts. These properties are demonstrated on real datasets.

1. Introduction

Motivation. In the recent years, several effective methods for the automatic generation of spatio-temporal models of dynamic scenes from video have been proposed [1, 3, 7, 8, 10, 11, 15, 17, 18, 19, 21]. However, capturing, processing and displaying the visual attributes, such as color, of such models, has been quite overlooked so far.

Surely, the case of static scenes, *i.e.* 3-D modeling from photographs, has been extensively studied: there exist several established technique [4, 13, 14, 20] for creating image-based texture atlases, while avoiding visual artifacts such as color discontinuities, ghosting or blurring, which typically arise from photometric and geometric inaccuracies (varying light conditions and camera responses, non-Lambertian reflectance, imperfect camera calibration, approximate shape, ...).

But the case of spatio-temporal models presents additional challenges. As the reflectance properties of the scene typically remain constant through time, the time redundancy in the input video sequences is very high. Fully exploiting this redundancy requires to adopt an actual spatio-temporal perspective, beyond independent frame-by-frame texture atlases. While the latter involve a prohibitively huge amount of texture data, *spatio-temporal texture atlases* are expected to be much more concise. Potentially, they could also take advantage of time redundancy to recover from ambiguities and deficiencies in the input data.

Previous work. To our knowledge, there are only a few notable works on spatio-temporal texture atlases. In [17] the authors propose to warp all images in a common planar parameterization of the animated mesh, in order to recover spatially varying surface reflectance properties. However, this parameterization-based approach involves image resampling and loss of visual detail, and requires to carefully position cuts on the surface to avoid unacceptable distortion. In contrast, in the vein of several successful static 3-D methods [4, 13, 14, 20], we take advantage of the projective transformations from the moving surface to the input video frames, which constitute natural and optimal mappings.

In [2, 23] average textures are computed. First, average images are generated considering all the cameras, but separately for each frames. Then, these average images are merged into one final texture, by substituting the texture of invisible patches by the texture from the closest frame where it is visible [2], or by averaging over frames [23]. Due to blending, the results of these methods suffer from undesirable artifacts, such as ghosting or blurring, that we avoid.

The general principle of our method is to compute an optimal partition of the spatio-temporal surface of the scene, into patches associated to the different input images. The problem is cast as a *Markov random field* optimization:

to this extent, our work is most closely related to those of [4, 13] for static scenes. In this paper, we reformulate and extend these works, in order to meet the specific requirements and issues of spatio-temporal scenes.

Contribution. Our method enjoys several remarkable features. First, we drastically cut down on the amount of texture data, and thereby we greatly enhance the portability and the rendering efficiency of the model. Second, we gather the numerous different viewpoint/time appearances of the scene, so as to recover from low resolution, grazing views, highlights, shadows and occlusions which affect some regions of the spatio-temporal model[1]. Altogether, our method allows the synthesis of novel views from a small quantity of texture data, with an optimal visual quality throughout the sequence, with minimally visible color discontinuities, and without flickering artifacts. These properties are demonstrated on real datasets.

2. Problem Formulation and Solution

In this paper, we focus on a representation of dynamic scenes as *animated meshes*. An animated mesh consists in a sequence of meshes with a fixed connectivity (rather than unrelated meshes), whose time-varying vertex positions sample the trajectories of material points. It is a widely used representation in computer graphics, especially in computer animation.

This choice is not restrictive since there exist several methods for producing animated meshes of real dynamic scenes, either directly from video [3, 7, 8, 10, 15, 17, 21], or from time-varying point clouds [16, 22] (the latter being obtained from video or from fast 3-D scanning hardware).

In the following, we consider a dynamic scene, imaged by N calibrated and synchronized video sequences composed of T frames, and approximated by an animated mesh with F polygonal faces. We note:

- $I_{n,t}$, $n \in \{1..N\}$, $t \in \{1..T\}$ the input images,
- $f_{k,t}$, $k \in \{1..F\}$, $t \in \{1..T\}$ the faces of the animated mesh at the different time instants.

2.1. Principle

Our method is based on two central assumptions. The first assumption is that the *reflectance properties* of the surface do not change through time. Please note that this does not exclude *appearance* changes between images, caused by non-Lambertian reflectance, varying shading, shadows and highlights along scene motion, or varying lighting conditions. Once this clarification is made, it appears that an overwhelming majority of real-world scenes fulfill this assumption. The second assumption is that a mesh face corresponds to a same material patch throughout the sequence. The animated mesh representation precisely enforces this property.

Our method exploits these two assumptions to estimate a *normalized appearance* of the surface. Let us consider a face of the animated mesh, and the set of input images in which it is visible. Our rationale is that among these numerous different viewpoint/time appearances of the face, one or several of them are likely to approach ambient lighting conditions, and in particular to be exempt from shadow and highlights. By assigning each face of the animated mesh to one of these adequate input images, we can assemble a spatio-temporal texture atlas which allows the synthesis of *normalized* views of the dynamic scene, from any viewpoint and at any time instant.

Using a constant texture source for a face throughout the sequence has many desirable outcomes. First, we drastically cut down on the amount of texture data. The latter do not scale with the number of frames anymore. Second, if a region of the dynamic scene is visible once (in any time frame, from any input viewpoint), it can be rendered throughout the sequence. If a region is out-of-shadow once, we can discard the shadow throughout the sequence. The same benefits apply to highlights, texture resolution, and so on. Finally, we completely eliminate *flickering artifacts*, *i.e.* small color fluctuations in the animation, to which human eyes are very sensitive.

At this point, a clarification is needed. Rendering regions of the scene that are not visible in any input image at this time seems questionable at first sight: no information about the very geometry of these regions can be expected from the input data. However, most spatio-temporal reconstruction techniques are still able to infer relevant geometry and motion there, by assuming the spatial and temporal coherence of the scene. In turn, our method is able to infer texture in these regions.

The assignment of faces of the animated mesh to input images can be encoded by a labeling function

$$\mathcal{L} : \{1..F\} \rightarrow \{1..N\} \times \{1..T\}, \qquad (1)$$

such that $\forall t$, face $f_{k,t}$ is textured from image $I_{\mathcal{L}(k)}$. To be more exact, faces that are not visible in any input image (in any time frame, from any input viewpoint) are discarded altogether, and are not considered in the above equation and in further discussions. Also, we may want to consider only a representative subset of the original time frames, in order to keep the computational complexity of our method sustainable for very long sequences. The rationale behind the latter simplification is that, with a sufficient number of frames with enough variety, the spatio-temporal texture atlas is very close to optimal, and is not significantly further

[1] One should note that highlights and shadows are necessary for photo-realistic rendering, but they have to be eliminated from the input videos and then synthesized depending on the new environment.

improved by supplemental frames. Remarkably, such an atlas can still be used to synthesize novel views of all original time frames.

The labeling (1) induces a partition of the animated mesh into *patches*. On the one hand, all faces inside a patch get their texture from the same image, so color is continuous across edges interior to the patch. On the other hand, at patch boundaries, *i.e*. at edges between faces with different labels, photometric and geometric inaccuracies in the data, in particular approximate geometry and motion, imperfect camera calibration, are likely to cause visually annoying color discontinuities (*seams*). Our method is able to compensate for most of these perturbations. It computes a labeling which optimizes visual quality, in some sense formally defined below, while minimizing the visibility of seams.

2.2. Variational Formulation

More specifically, we adopt a variational formulation: the optimality of a labeling is quantified by an energy functional composed of two terms, measuring the local visual quality and the visibility of seams, respectively:

$$E(\mathcal{L}) = E_{quality}(\mathcal{L}) + \mu E_{seams}(\mathcal{L}) \ , \tag{2}$$

where μ denotes a weighting factor. In the rest of this subsection, we detail some possible definitions of these two energy terms. Subsection 2.3 describes the minimization procedure applied to the energy functional.

The visual quality term $E_{quality}$ is local: it does not consider interactions between neighboring regions of the model. Hence we write it as a sum over faces:

$$E_{quality}(\mathcal{L}) = \sum_{k=1}^{F} \phi_k(\mathcal{L}(k)) \ , \tag{3}$$

where $\phi_k(n,t)$ quantifies how appropriate image $I_{n,t}$ is for texturing faces $f_{k,\cdot}$ of the animated mesh. We can use different criteria to assess this quality.

In previous work on static scenes, several strategies have been used. In [13], the angle between viewing direction and face normal is proposed. In [4], the area of the projected face is advocated instead; in our context, this would write

$$\phi_k(n,t) = -\text{area}\left[\Pi_n(f_{n,t})\right] \ , \tag{4}$$

where Π_n denotes the projection from 3-D space to video sequence n. The latter choice would lead to an easily interpretable definition of visual quality: the total number of *texels* (texture elements) on the animated mesh.

Nevertheless, the above definitions mistakenly identify visual quality with visual detail. They fail to account for photometric aspects such as shadows and highlights. If the 3-D positions of light sources are known, we can easily estimate shadow and highlight regions on the spatio-temporal surface, and define visual quality as the total number of *shadow-free and highlight-free* texels on the animated mesh.

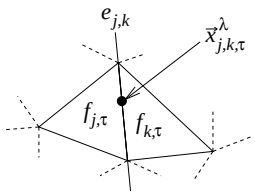


Figure 1. Notations of adjacent faces for term E_{seams}.

As this additional information is not available in practice for all datasets, we propose an alternative approach. Given a face, we compute its average projected area over all images where it is visible. To guarantee sufficient visual detail, we discard all images below average, *i.e*. we set ϕ_k to infinity for these images. For the remaining possible images, we set ϕ_k to the difference between the actual face intensity and the median of intensities. Here again, we compute the median over all images where the face is visible. Unfortunately, we could not test this criterion in this paper, for lack of datasets with significant shadows and highlights.

The second term E_{seams} measures how smoothly the color changes passing from one face to an adjacent. If the two faces take their textures from the same image, *i.e*. their label is the same, then the borderline between them is continuous. On the other hand, neighboring faces with different labels are likely to cause seams at the common edge. In order to minimize seam visibility, the second energy term is defined as the integral along the seams of color discrepancy between bordering images.

Further notations are needed to write a formal expression of this term. (See Figure 1.) Let us denote by $e_{j,k}$ a non-border edge of the animated mesh, adjacent to faces $f_{j,\cdot}$ and $f_{k,\cdot}$. We note $\vec{x}^{\lambda}_{j,k,\tau}$, $\lambda \in [0,1]$ a linear parameterization of $e_{j,k}$ at time τ. Notice that the color at a point $\vec{x}^{\lambda}_{j,k,\tau}$ on an edge does not depend on the time frame τ, but only on the selected input image (n,t). We note $c^{\lambda}_{j,k}(n,t)$ the color at $\vec{x}^{\lambda}_{j,k,\cdot}$, extracted from image $I_{n,t}$:

$$c^{\lambda}_{j,k}(n,t) = I_{n,t} \circ \Pi_n(\vec{x}^{\lambda}_{j,k,t}) \ . \tag{5}$$

With these notations in hand, E_{seams} writes as a sum over the set $\mathcal{E}$ of all non-border edges:

$$E_{seams}(\mathcal{L}) = \sum_{e_{j,k} \in \mathcal{E}} \|e_{j,k}\| \, \psi_{j,k}(\mathcal{L}(j), \mathcal{L}(k)) \ , \tag{6}$$

$$\psi_{j,k}(\ell, \ell') = \int_0^1 \left\| c^{\lambda}_{j,k}(\ell) - c^{\lambda}_{j,k}(\ell') \right\| d\lambda \ . \tag{7}$$

$\|e_{j,k}\|$ is computed as the average length of the edge over all frames.

2.3. Optimization Procedure

It must be noted that $\psi_{j,k}$ is a semi-metric: $\forall \ell, \ell', \ell'' \in \{1..N\} \times \{1..T\}$,

- $\psi_{j,k}(\ell, \ell) = 0$,
- $\psi_{j,k}(\ell, \ell') = \psi_{j,k}(\ell', \ell) \geq 0$,
- $\psi_{j,k}(\ell, \ell'') \leq \psi_{j,k}(\ell, \ell') + \psi_{j,k}(\ell', \ell'')$.

This still holds with any metric on colors instead of the usual Euclidean distance in RGB color space.

This has an important practical consequence: it allows us to minimize the energy functional (2) with α-*expansion* [6, 12]. It consists in translating the labeling problem, which is generally NP-hard, to a succession of binary minimum cut problems. Efficient solutions to these min-cut problems are described in [5]. The whole process monotonically decreases the energy and is guaranteed to converge to a *strong local minimum*, thereby ensuring a close-to-optimal seam placement.

In our implementation, we use the graph cuts minimization software by O. Veksler (http://www.csd.uwo.ca/~olga/code.html) and by V. Kolmogorov (http://www.adastral.ucl.ac.uk/~vladkolm/software.html).

2.4. Texture Atlas Creation

We could synthesize novel views from the input video sequences using multiple passes of projective texture mapping [9]. However, the creation of a single rectangular texture map is very desirable: it increases the rendering efficiency and allows to output portable animated 3D formats. To build such a spatio-temporal texture atlas, we consider the binary masks representing the useful regions in the different images: $M_{n,t}$ is the projection in $I_{n,t}$ of the associated patch:

$$M_{n,t} = \Pi_n \left[\bigcup_{\mathcal{L}(k)=(n,t)} f_{k,t} \right] . \quad (8)$$

We first apply a morphological dilation to the masks with a square structural element of a few pixels, in order to provision for automatic texture minifying during rendering. We then compute a connected component decomposition, yielding a list of texture fragments. We pack the latter using a classical first-fit decreasing strategy: we place the fragments in decreasing order of size, at the first available slot found along a scanline search in the atlas. Finally, we set the texture coordinates of the vertices of the animated mesh accordingly. Thus, the final output of our algorithm is compatible with standard 3D viewers.

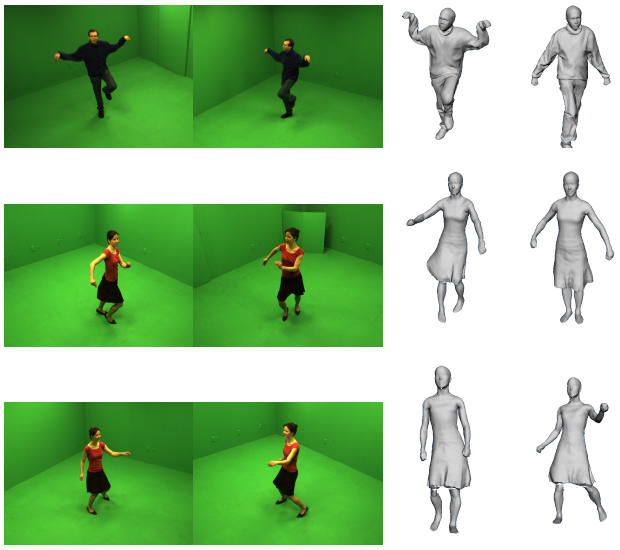


Figure 2. Some views of our test datasets. From left to right: two input images, two views of the animated mesh. From top to bottom: 'crane', 'samba' and 'swing' datasets.

3. Experimental Validation

3.1. Input Datasets

In order to validate our method under real conditions, we tested it on three challenging datasets: 'crane', 'samba' and 'swing', that were made available online by the authors of [21] at http://people.csail.mit.edu/drdaniel/mesh_animation/.

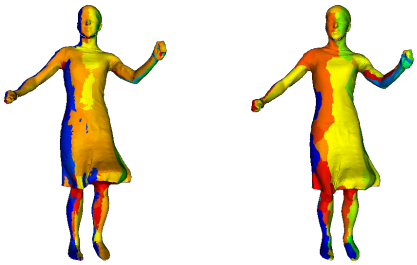
Each dataset consists of 8 calibrated and synchronized 1600×1200 color video streams. 'crane' and 'samba' are composed of 175 frames. 'swing' is composed of 150 frames. The datasets also include an animated mesh of the dynamic scene, of approximately 10K vertices and 20K faces, obtained in accordance with [21]. Figure 2 displays some views of these datasets.

3.2. Compared Methods

On each of the datasets, we compare four different methods, which we call 'single-frame', 'single-frame optimized', 'multi-frames' and 'multi-frames optimized'.

The 'single-frame' and 'single-frame optimized' methods both compute independent frame-by-frame texture atlases. The 'single-frame' method greedily maps each face to the highest-quality input image of the current frame. The 'single-frame optimized' achieves a trade-off between visual quality and visibility of seams, similarly to [4, 13].

The 'multi-frames' and 'multi-frames optimized' methods both compute a single spatio-temporal texture atlas. To limit computational expense, we build this atlas from a subset of the input time frames, namely 10 uniformly distributed time frames, with no noticeable deterioration of the



'multi-frames' 'multi-frames optimized'

Figure 6. Color-coded partition of the surface obtained on the 'samba' dataset, with two different methods.

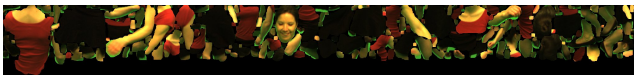


Figure 7. A sample spatio-temporal texture atlas.

results. The 'multi-frames' method greedily maps each face to the highest-quality input image among all viewpoints and all selected time frames:

$$\mathcal{L}(k) = \arg\max_{n,t} \phi_k(n,t) \ .$$

Finally, the 'multi-frames optimized' method is the one described in this paper. Let us mention that the 'multi-frames optimized' method degenerates to the 'multi-frames' method when setting the weighting factor μ of E_{seams} to zero.

In all these experiments, we use the number of texels (4) as the visual quality measure.

3.3. Results

The results of the four aforementioned methods on the 'crane', 'samba' and 'swing' datasets are compared in Figures 3, 4 and 5, respectively. Due to space limitations, we only show three views for each method and each dataset, synthesized at three different time instants. A more convenient and more comprehensive visual comparison can be conducted using the videos available in our supplemental material.

Also, for illustration purposes, Figure 6 displays the partition of the surface obtained on the 'samba' dataset with the 'multi-frames' and the 'multi-frames optimized' methods. Finally, Figure 7 displays a sample spatio-temporal texture atlas, obtained from the 'swing' dataset.

3.4. Discussion

These results clearly demonstrate one major advantage of multi-frames methods over single-frame methods: the ability to cover temporarily hidden regions as well, which results in significantly smaller untextured regions in novel views.

Another drawback of single-frame methods is also apparent in these results, particularly in the accompanying videos: the incoherence of surface partition across time frames produces small color fluctuations, known as flickering artifacts, to which human eyes are very sensitive.

The poor results of the 'multi-frames' method reveal the important geometric inaccuracy of the input datasets (approximate calibration, geometry and motion). This inaccuracy prevents from naively assembling a satisfactory spatio-temporal texture atlas from input images that are very distant, in viewpoint or in time. Remarkably, our method ('multi-frames optimized') turns out to be robust to such imperfect data, indicating that a global optimization of surface partition is essential for the success of our approach.

At the same time, our experiments highlight a limit of spatio-temporal texture atlases: if very detailed geometry and motion, such as creases of clothes and facial expression, are not modeled in the animated mesh, they cause a violation of our central 'constant reflectance' assumption. As a result, these variations are averaged out by our approach. This explains the visually intriguing static faces visible in our results.

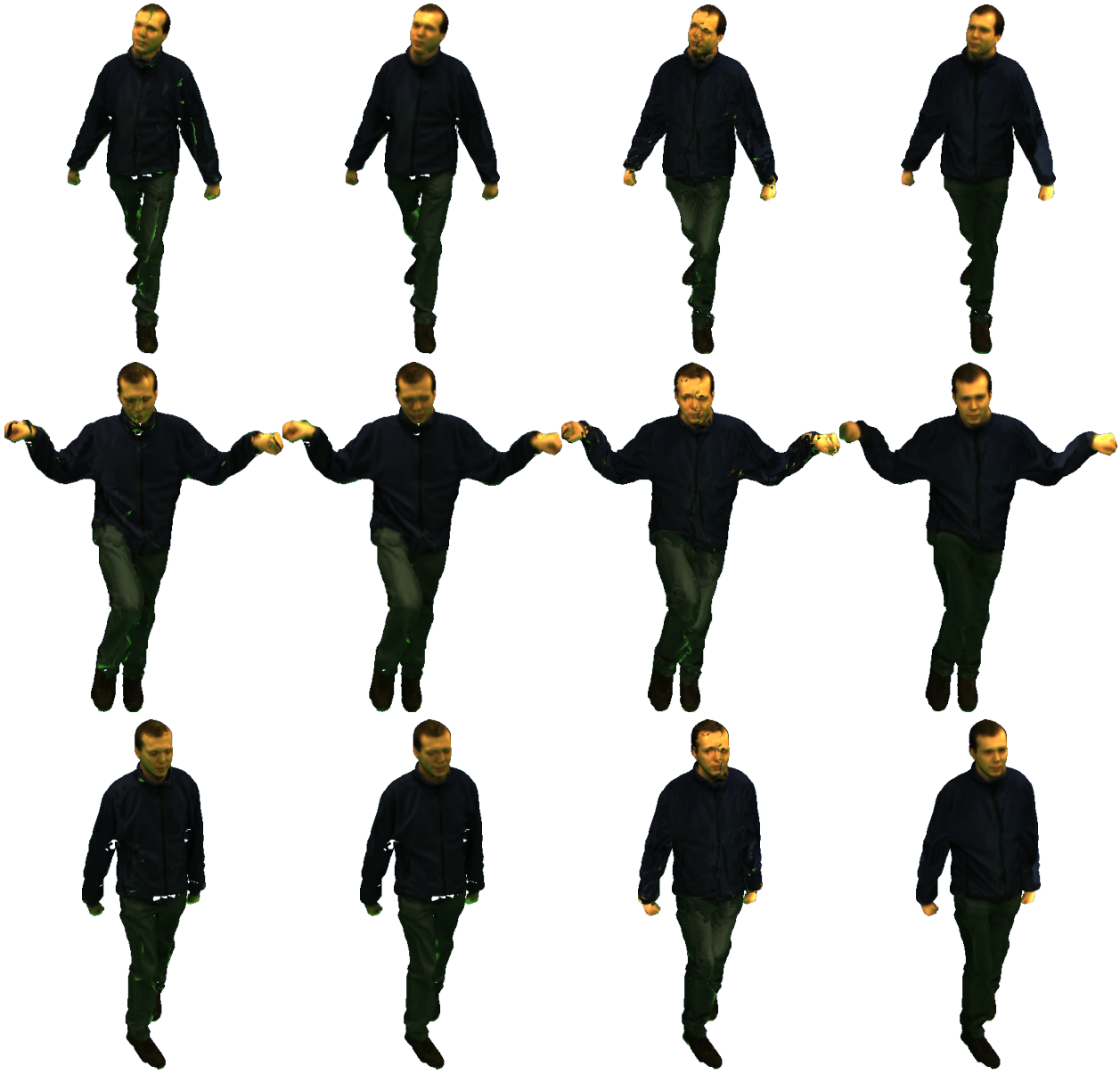
Besides, we should note that our method does not explicitly take photometry into account, and although our method could evolve in this direction, in this paper we deliberately adopt an orthogonal approach: we estimate a plausible diffuse map of the scene by automatically discarding complex photometric effects. Our method, though it has limitations, is simple and easy to implement, and our results show significant advantages compared to frame-by-frame texturing.

4. Conclusion

We have proposed a method to create high-quality spatio-temporal texture atlas for dynamic 3-D model. The texture map is built using a set of calibrated video sequences by exploiting its high redundancy. We represent the dynamic scene as animated mesh with fixed connectivity, and assume the reflectance properties of the surface to be constant through time, which is valid for most real-world scenes.

The main contribution is twofold. First, the amount of texture data is drastically reduced by eliminating redundancy, which greatly accelerates rendering and helps portability. Second, using several different viewpoint/time appearances of the scene, we can recover from low resolution, grazing views, and reduce the annoying effects of highlights, shadows and occlusion. We have demonstrated the advantages of our method compared to single-frame texturing on real datasets.

Acknowledgment. This work was supported by the French National Agency for Research (ANR) under grant ANR-06-MDCA-007.



'single-frame' 'single-frame optimized' 'multi-frames' 'multi-frames optimized'

Figure 3. Results on the 'crane' dataset. See text for more details.

References

[1] E. Aganj, J.-P. Pons, F. Ségonne, and R. Keriven. Spatio-temporal shape from silhouette using four-dimensional Delaunay meshing. In *Proc. ICCV'07*, 2007. 1

[2] N. Ahmed, E. de Aguiar, C. Theobalt, M. Magnor, and H.-P. Seidel. Automatic generation of personalized human avatars from multi-view video. In *Proc. VRST'05*, pages 257–260, 2005. 1

[3] N. Ahmed, C. Theobalt, C. Rössl, S. Thrun, and H.-P. Seidel. Dense correspondence finding for parametrization-free animation reconstruction from video. In *Proc. CVPR'08*, 2008. 1, 2

[4] C. Allène, J.-P. Pons, and R. Keriven. Seamless image-based texture atlases using multi-band blending. In *Proc. ICPR'08*, 2008. 1, 2, 3, 4

[5] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004. 4

[6] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001. 4

[7] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view

'single-frame' 'single-frame optimized' 'multi-frames' 'multi-frames optimized'

Figure 4. Results on the 'samba' dataset. See text for more details.

video. In *Proc. ACM SIGGRAPH*, 2008. 1, 2

[8] E. de Aguiar, C. Theobalt, C. Stoll, and H.-P. Seidel. Marker-less deformable mesh tracking for human shape and motion capture. In *Proc. CVPR'07*, 2007. 1, 2

[9] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proc. SIGGRAPH*, pages 11–20, 1996. 4

[10] Y. Furukawa and J. Ponce. Dense 3D motion capture from synchronized video streams. In *Proc. CVPR'08*, 2008. 1, 2

[11] B. Goldlücke and M. Magnor. Space-time isosurface evolution for temporally coherent 3D reconstruction. In *Proc. CVPR'04*, volume 1, pages 350–355, 2004. 1

[12] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *Proc. ECCV'02*, pages 65–81, 2002. 4

[13] V. S. Lempitsky and D. V. Ivanov. Seamless mosaicing of image-based texture maps. In *Proc. CVPR'07*, 2007. 1, 2, 3, 4

[14] C. Rocchini, P. Cignoni, C. Montani, and R. Scopigno. Acquiring, stitching and blending diffuse appearance attributes on 3D models. *The Visual Computer*, 18(3):186–204, 2002. 1

[15] J. Starck and A. Hilton. Surface capture for performance based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007. 1, 2

'single-frame' 'single-frame optimized' 'multi-frames' 'multi-frames optimized'

Figure 5. Results on the 'swing' dataset. See text for more details.

[16] J. Süßmuth, M. Winter, and G. Greiner. Reconstructing animated meshes from time-varying point clouds. *Computer Graphics Forum (Proc. SGP'08)*, 27(5):1469–1476, 2008. 2

[17] C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, and H.-P. Seidel. Seeing people in different light-joint shape, motion, and reflectance capture. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):663–674, 2007. 1, 2

[18] K. Varanasi, A. Zaharescu, E. Boyer, and R. P. Horaud. Temporal surface tracking using mesh evolution. In *Proc. ECCV'08*, volume 2, pages 30–43, 2008. 1

[19] S. Vedula, S. Baker, and T. Kanade. Image-based spatio-temporal modeling and view interpolation of dynamic events. *ACM Trans. on Graphics*, 24(2):240–261, 2005. 1

[20] L. Velho and J. Sossai Jr. Projective texture atlas construction for 3D photography. *The Visual Computer*, 23(9):621–629, 2007. 1

[21] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3), 2008. 1, 2, 4

[22] M. Wand, P. Jenke, Q. Huang, M. Bokeloh, L. Guibas, and A. Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *Proc. SGP'07*, pages 49–58, 2007. 2

[23] G. Ziegler, H. Lensch, M. Magnor, and H.-P. Seidel. Multi-video compression in texture space. In *Proc. ICIP'04*, volume 4, pages 2467–2470, 2004. 1