Adaptive Mesh Texture for Multi-View Appearance Modeling

Matthieu Armando, Jean-Sébastien Franco, Edmond Boyer

Adaptive Mesh Texture for Multi-View Appearance Modeling

Matthieu Armando[†] Jean-Sebastien Franco Edmond Boyer
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
firstname.name@inria.fr

Abstract

In this paper we report on the representation of appearance information in the context of 3D multi-view shape modeling. Most applications in image based 3D modeling resort to texture maps, a 2D mapping of shape color information into image files. Despite their unquestionable merits, in particular the ability to apply standard image tools, including compression, image textures still suffer from limitations that result from the 2D mapping of information that originally belongs to a 3D structure. This is especially true with 2D texture atlases, a generic 2D mapping for 3D mesh models that introduces discontinuities in the texture space and plagues many 3D appearance algorithms. Moreover, the per-triangle texel density of 2D image textures cannot be individually adjusted to the corresponding pixel observation density without a global change in the atlas mapping function. To address these issues, we propose a new appearance representation for image-based 3D shape modeling, which stores appearance information directly on 3D meshes, rather than a texture atlas. We show this representation to allow for input-adaptive sampling and compression support. Our experiments demonstrate that it outperforms traditional image textures, in multi-view reconstruction contexts, with better visual quality and memory footprint, which makes it a suitable tool when dealing with large amounts of data as with dynamic scene 3D models.

1. Introduction

Image based 3D shape modeling is the process of building digital models of shapes using real images. It finds applications in many domains, in particular with the new virtual and augmented reality devices and the associated need for 3D contents. In order to represent the reconstructed 3D shapes, the dominant paradigm is to model them as geometric surfaces over which appearance functions are defined. With such a model, both geometric and appearance features fundamentally contribute to convey realism and fidelity to the observed shapes. In this work we focus on the appearance representation, which has received surprisingly little attention from the 3D vision community. In particular, we show that both visual quality and encoding efficiency can be significantly improved by exploring representations beyond traditional 2D texture maps.

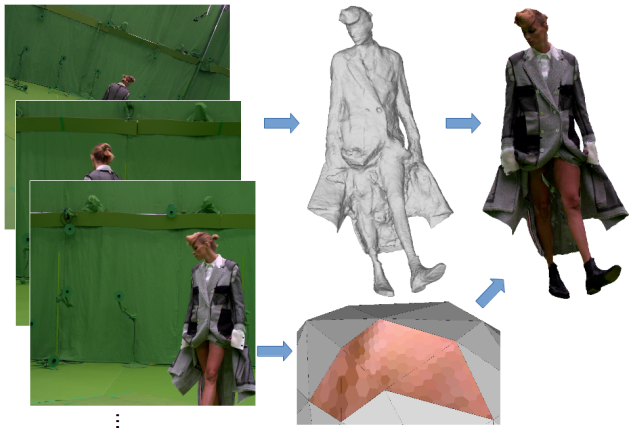

Figure 1: Image-based 3D modeling: given a set of input photographs (left), a geometric mesh is computed (top), along with an appearance function stored within the surface mesh structure (bottom), the main focus of this paper.

Assume we are given a geometric shape model estimated from n images, using for instance a multi-view stereo approach [12]. An optimal solution for the appearance, and with respect to the observed information, is to keep all the original images in the representation. This appears to be inefficient, or even intractable, in many situations where possibly numerous high resolution cameras, and potentially temporal sequences of shapes, are considered. Hence, most 3D visual modeling frameworks usually resort to an appearance representation in the form of a 2D regular grid of texels whose values, typically RGB colors, are estimated from the observed images, given the geometry. The main benefits of these texture maps are their controlled and limited sizes in addition to inheriting most regular 2D image tools, including compression and filtering. On the other hand, such

[†]MSR-INRIA Joint Center

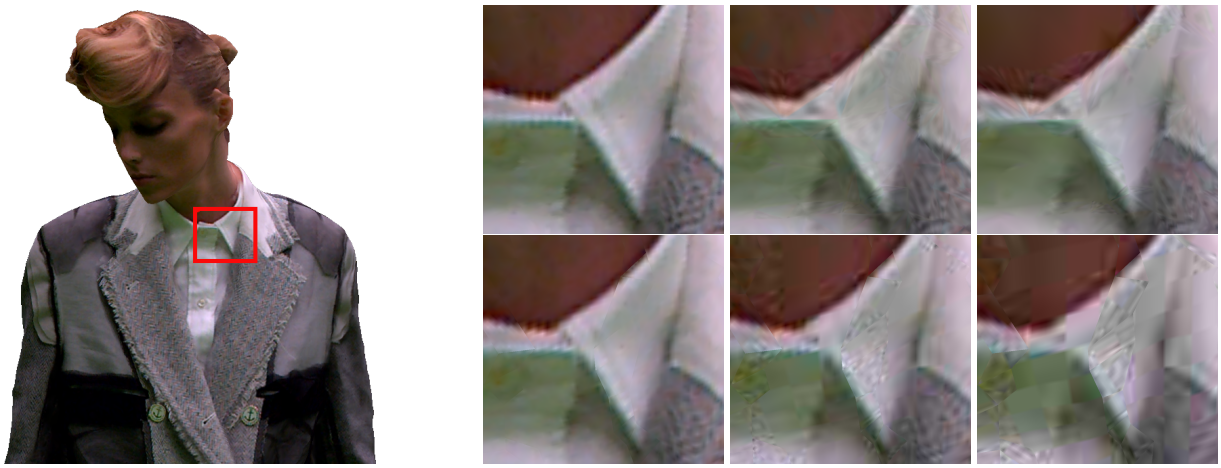

Figure 2: Visualization and surface close-up of our appearance compression scheme: **Top**: mesh texture. **Bottom**: image texture. Left to right: high-resolution texture, texture compressed to decreasing file size.

representations present severe limitations. First, a 3D to 2D mapping is required to associate geometric points on the 3D shape to appearance information in the 2D texture. To this purpose, the 3D shape model is usually cut into charts homeomorphic to discs that are parameterized with texel coordinates. The resulting 2D texture atlases are discontinuous by construction, creating unnecessary seams on the model appearance and making local computations on the appearance function difficult; In addition, locally adjusting the per-triangle texel sampling to the density of pixels oberving each triangle is unpractical, as it is globally bound by the initially chosen texture resolution and would require global optimization of the atlas mapping function.

Instead of an appearance stored in an intermediate 2D grid, we advocate therefore for an appearance stored on the 3D shape surface with the following advantages:

- There is no need for a 3D to 2D mapping that induces discontinuities in the representation.
- Regions on the shape with varying observed pixel densities, as due to camera properties including resolutions and viewpoints, as well as to self-occlusions such as armpits on the human body, can be described with adapted appearance samplings.

To this end we build on the mesh color representation introduced in [24] for synthetic graphical models and propose an adaptive appearance representation for 3D image based reconstruction. We show that this representation outperforms traditional texture atlases with more efficient appearance representations with respect to both precision and size. In addition, we also introduce a dedicated compression method that builds on the JPEG pipeline and allows for effective appearance storage with our representation. The efficiency improvements of the proposed contributions allow for promising image based 3D modeling applications.

2. Related Work

2.1. Appearance Representation.

Besides the widespread texture map representation mentioned before, attempts have been made for representations beyond 2D textures in order to address their limitations. The thorough survey of Tarini *et al*. [18] gives an exhaustive overview on the subject. For instance Ptex [3] used by Walt Disney Animations Studios, and Mesh Colors [24] are particularly noteworthy as they are used in real production pipelines in the movie industry ([10]). In both cases, they use the intrinsic parameterisation of the model, rather than an arbitrary mapping. While Ptex was developed for subdivision surfaces, and only works with quad faces, Mesh colors can be used for triangular meshes which is the 3D model representation we consider in this work. However, works on this model focus yet on the graphics side, either trying to optimize it for rendering [23] or creating 3D painting tools for artists to work with [10]. For image-based modeling in vision applications, texture maps are still the main appearance representation and, to the best of our knowledge, no attempts have been made to replace them.

2.2. Multi-view Appearance Estimation.

Appearance representations are used to store the appearance information for 3D models. In the multi-view reconstruction context, such information is estimated by considering the color cues from several input views. In the case of texture maps, texels are usually backprojected into input images and the corresponding pixel values are combined with: blending approaches that average pixel colors to determine

texel values, *e.g*. [16, 1]; or more recently using superresolution strategies, as in [9, 19, 8], that allow for prior constraints, *e.g*. TV regularization, to be applied on the estimated texture. In any case, the disjoint charts of a texture atlas introduce discontinuities in the texture space that make filtering, regularization, or any non-local computation over the texels difficult to perform, since neighbors must be determined across chart boundaries. In that respect, a representation attached to the shape parametrisation, *e.g*. mesh triangles, is much more efficient with connectivity relationships that are intrinsic. In addition, it enables for adaptive resolutions where a texture has a uniform pixel resolution.

2.3. Appearance Compression.

Reducing the memory footprint is an important issue when streaming or storing 3D contents. This is even more critical with dynamic scenes for which shape and appearance information evolve over time. In the case of 2D textures, appearance information can be optimally compressed using image techniques with nevertheless some differences. Whereas most image compression techniques are optimized for storage and transmission, and thus, focus on visual quality vs. bitrate, texture compression puts more emphasis on random access and decoding speed [2]. State of the art methods include ASTC (adaptive scalable texture compression) [14] and ETC (Ericsson Texture Compression), published as iPACKMAN [17]. Putting aside the concern about random access and decoding speed, and focusing on bit rate instead, state of the art methods are then mostly standard image compression techniques such as JPEG. This is extended to dynamic scenes in [4] where the standard H.264 compression format is used to compress texture frames over time, assuming for that purpose the texture atlas to be fixed over frames. This enables temporal redundancy to be exploited, which is an interesting feature however beyond the scope of this paper. With mesh colors representations, for which appearance information does not live in 2D images but on the model representation, *i.e*. mesh triangles, standard image compression techniques do not directly apply. We propose a strategy that builds on JPEG to compress mesh color representations and to benefit therefore from its visual quality properties without sacrificing memory space with respect to 2D textures.

3. Appearance Models

Without loss of generality we assume that shapes are modeled with 3D triangle meshes. As mentioned earlier their appearances, typically color information, can be represented with 2D image textures, the traditional model, or directly on shapes with mesh textures. In this section we present both models, their parameterizations and sampling properties before considering more specifically in the next section, the 3D image based modeling context and the mesh texture tools we introduce for that purpose.

3.1. Notation

We consider the shape surface $\mathcal{M}$, as obtained with for instance a multi-view stereo reconstruction approach. $\mathcal{M}$ is represented as an ordered set of vertices $\{v_1 \dots v_N\}$ in $\mathbb{R}^3$, and an ordered set of triangles $\{\mathcal{T}_1 \dots \mathcal{T}_M\}$, where $\mathcal{T}_i \in [1, N]^3$ is a triplet of vertex indices. We want to represent the appearance of $\mathcal{M}$ as a *dense* function $C : \mathcal{M} \to \mathcal{C}$ that associates a color from a space $\mathcal{C}$ to every point on the mesh. Typically, $\mathcal{C} = [0 \dots 255]^3$ for a discrete RGB-space color representation. To this aim we parameterize point locations on $\mathcal{M}$ with barycentric coordinates and through the following function F:

$$\begin{aligned} F\colon \{1, ..., M\}\times]0,1[^2 &\to \mathcal{M} \\ (j, a, b) &\mapsto a\ v_x + b\ v_y + (1 - a - b)\ v_z\ , \end{aligned} \tag{1}$$

where $(x, y, z) = \mathcal{T}_j$ are the 3 vertex indices of triangle $\mathcal{T}_j$. The closure of the codomain of F is $\mathcal{M}$. We call F^{-1} the inner parameterization of $\mathcal{M}$. In other words, every point on $\mathcal{M}$ can be uniquely represented by its triangle index j and its barycentric coordinates (a, b), with the exception of vertices and edges. The function $C : \{1, ..., M\}\times]0, 1[^2 \to \mathcal{C}$ associates then a color to any point represented by (j, a, b).

3.2. Image Texture

We recall in this section the widely used image texture solution and discuss its properties.

3.2.1 Parameterization.

Image textures make use of an external mapping between an image and the 3D mesh $\mathcal{M}$ under consideration. $\mathcal{M}$ is divided into k patches $P_1, ..., P_k \subset \mathcal{M}$, such that $\cup_{j=1}^k P_j = \mathcal{M}$. For each patch, a separate function $\phi_i\colon P_i \to C_i \subset \mathbb{R}^2$ is computed independently that maps 3D points in P_i to 2D points in a chart C_i. Charts are chosen so that $\cup_{j=1}^k C_j = T \subset [0, 1]^2$ and $C_i \cap C_j = \emptyset, \forall i \neq j$ (see Fig. 3). The chart building process is thus typically a non trivial and global task, introducing discontinuities such that the whole mapping $\mathcal{M} \to T$ is piecewise continuous. The mapping $\phi = \cup_{j=1}^k \phi_j$ is typically defined per face, using 3 pairs of texture coordinates in T representing the images of its 3 vertices by ϕ. ϕ is then linearly interpolated inside the face. Vertices and edges are duplicated along chart discontinuities. In other words, a discrete function $\phi_0\colon \{v_1 \dots v_N\} \to T$ is defined on the vertices, and is then extended to the whole surface using the inner parameterization F^{-1} (1). An additional drawback of the scheme is thus that it must ensure that color information C, or any other surface function defined over T, should be consistent across charts to avoid visual artifacts and apparent color seams.

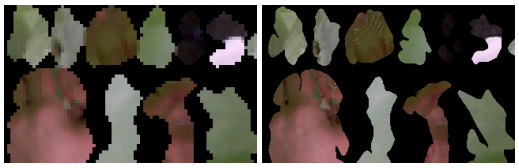

Figure 3: Close-up of the image texture of a mesh, generated with the same texture atlas but different resolutions. (Left: 80×50, Right: 320×200).

3.2.2 Sampling.

T is discretized with a rectangular image grid, inheriting all the associated image tools. The sampling depends on vertical and horizontal resolutions and the intially chosen mapping of triangles to texture space. Some texels (*i.e.* pixels in the texture) inevitably fall partly within and outside the chart union T. This means C must be extrapolated over the chart borders to avoid artifacts. This is even more critical if consistent filtering operations on C are to be performed.

3.2.3 Storage and Compression.

Image textures can be stored as regular image files and buffers, with numerous possible compression schemes, such as JPEG. The mapping function ϕ_0 is usually stored into the geometry file, in the form of texture coordinates. Assume E, N, M are respectivley the number of edges, vertices and faces of $\mathcal{M}$. As shown in [15], for large triangular meshes, the approximation $M \approx 2N$ is considered valid.

In general, the geometry information is represented by N 3D points, or $3N$ floating point numbers, and the connectivity is stored as M triplets of vertex indices, *i.e.* $\approx 6N$ integers. For appearance, $3M$ texture coordinate pairs are used, *i.e.* $12N$ floating points in the worst case scenario, or four times the geometry information, although this can be optimized with smart indexing. In the best case, neglecting cross-boundary chart redundancies, this takes $2N$ floating points (one pair of texture coordinates per vertex).

3.3. Mesh Texture

In contrast to image textures, the appearance information is stored on the mesh structure with mesh textures. Consequently, a 2D mapping function is not required and the appearance information is directly sampled over the mesh triangles, as illustrated in Figure 4. These ideas were introduced in [24] and we experiment and extend them with appearances from real images.

3.3.1 Parameterization.

Mesh textures do not require any form of external parameterization. Color information is defined directly on the mesh

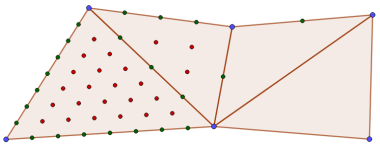

Figure 4: Appearance samples on a mesh texture model with triangles exhibiting varying appearance resolutions: from left to right, R = 8, 4, 2, 1.

structure and not in any other domain. As such, it only makes use of the inner parameterization $F^{-1}$of $\mathcal{M}$ (1).

3.3.2 Sampling.

Sampling is a strong feature of the model governed by a single resolution parameter. Given a triangular face $\mathcal{T}_i$ and its resolution R_i, a mesh texture stores $\frac{(R_i+1)(R_i+2)}{2}$ samples on the locations given by:

$$F(\mathcal{T}_i, \frac{m}{R_i}, \frac{n}{R_i}) \mid 0 \leq m \leq R_i, 0 \leq n \leq R_i - m\,. \quad (2)$$

In other words, the positions of appearance samples within a triangle are parametrized by the barycentric coordinates

$$(\frac{m}{R_i}, \frac{n}{R_i}, 1 - \frac{m+n}{R_i})\,, \quad (3)$$

associated to the triangle vertices. In practice, vertices and edge samples are shared between adjacent faces, and they must be treated separately. The resolution of an edge is defined as the lowest resolution of its two adjacent faces. As in [24], face resolutions can only take values that are powers of two. This makes interpolation easier along edges with a lower resolution than the face.

This choice of representation has several important advantages over image texture sampling. First, the sampling is by construction hexagonal, and hence more tightly packed and less directionally biased than the square sampling of image texture. This is formalized in information theory where hexagonal structures are shown to be optimal quantizers over 2D regular lattices [13]. Second, there is no discontinuity in the appearance model, nor any distortion. This makes filtering more accurate and much easier to perform in practice, as illustrated in Figure 5. Third, sampling frequency can be chosen locally, at the face level, whereas it is fixed globally with image textures (local sampling frequency depends on the vertical and horizontal texture resolution and the mapping function). This is particularly suitable for multi-view 3D modeling, as detailed in the next section. Fourth, local editing or resolution changes do not require a complete resampling or recharting of the mapping.

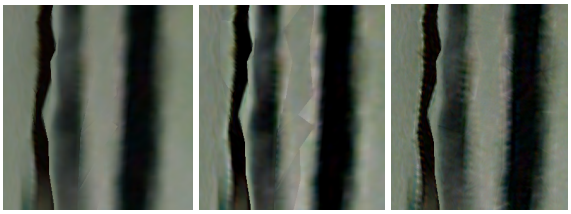

Figure 5: Illustration of a filtering operation with a texture from real images. Three close-ups of a mesh, rendered with: Left: the unfiltered image texture, middle: the same image texture with a simple sharpening filter applied, right: the mesh texture with a sharpening operation applied directly on the mesh. Artifacts appear despite the texture being dilated many times in the image texture. They are due to the seams lying too close to peaks in the color gradient.

3.3.3 Storage and Compression.

Similar to [24], we decouple geometry and color information, to make it suitable for standard graphics pipeline. Each vertex, edge and face stores an index to a single mesh global color array. No compression scheme is yet available for mesh textures and we introduce a novel approach for that purpose in the next section 4.2.

4. Mesh Textures from Real Images

As previously discussed, mesh textures are convenient when modeling shapes using real images. We detail in this section the related aspects in that case, namely the strategies for the appearance sampling and the compression.

4.1. Sampling Strategy

Given the pixel information in the observed images we define an adaptive sampling strategy that optimally exploits the appearance data. To this aim, we first choose a resolution level per triangle before computing the mesh texture. Then, in a post-processing step, we downsample triangles when they can still be accurately interpolated by the next lower resolution level, this up to a given error threshold. The first step depends directly on the pixel density in the input images whereas the second step depends on the amount of information present in the pixels. Figure 6 illustrates the benefits of this sampling strategy: The uniform sampling of texels with the image texture implies that unseen or uniform areas are oversampled, while others could benefit from a denser sampling, *e.g.* the button on the coat. In contrast, the mesh texture representation samples fewer points in the hidden parts of the mesh but has a denser sampling in this highly textured area.

Figure 6: Right: Two close-up renderings of the same colored mesh, with an image texture (top) and a mesh texture (bottom), using approximately the same number of samples, rendered with closest point interpolation. (Close-ups of the model on the left)

4.1.1 Adaptive Sampling

Our sampling strategy adapts the resolution R_i of the triangle $\mathcal{T}_i$ to the number of pixels that observe this triangle. To this aim, we assume that, for any subpart $S \subset \mathcal{M}$ of the mesh, a function $\mathcal{D}$ gives a measure of the available input pixels relevant to S, dependent on the coloring strategy chosen. For example, a super-resolution strategy can define $\mathcal{D}$ as the total number of pixels, in all selected views, that project onto S. A naive blending approach will define $\mathcal{D}$ as the maximum number of pixels, among all views, that project onto S. In this paper, we implement a strategy that selects a single *best* camera per triangle (see 5.1). Thus, $\mathcal{D}$ measures the number of pixels in this selected view.

Now given the number $\mathcal{D}(\mathcal{T}_i)$ of pixels in triangle $\mathcal{T}_i$ we need to find the appropriate appearance resolution R_i for that triangle. For such a resolution, the number of appearance samples in the triangle is approximately:

$$\frac{(R_i-2)(R_i-1)}{2} + \frac{3}{2}(R_i-1) + \frac{3}{6} = \frac{1}{2}R_i^2\,, \quad (4)$$

counting samples within the triangle as full, samples over edges as shared with 2 triangles and samples on vertices as shared, on average, among 6 triangles. Given this number, for each triangle $\mathcal{T}_i$, we choose the smallest possible resolution R_i that is a power of two and such that:

$$\frac{1}{2}\lambda R_i^2 < \mathcal{D}(\mathcal{T}_i)\,, \quad (5)$$

where the parameter λ can be chosen depending on the ex-

pected rendering quality. Typically, $\lambda = 2$ in our experiments.

4.1.2 Downsampling

Real world objects often present large regions with more or less uniform appearances. In such a region, irrespective of the resolution R_i, the appearance samples of a triangle will carry redundant color information. In order to account for that, we downsample triangles that fall into this category. More precisely, for each triangle $\mathcal{T}_i$, we consider all sample points associated to the current resolution level and compare their color values with the ones they would get by just interpolating the next lower-resolution level. Looking back at equations (2) and (3), these are samples with barycentric coordinates m odd, or n odd, or both. Denoting $o = R - m - n$, we compute the average difference as:

$$E = \frac{8}{3n(n+1)} \times (\Sigma_m + \Sigma_n + \Sigma_o)\,, \tag{6}$$

with:

$$\Sigma_m = \sum_{m \text{ even},\, n,o \text{ odd}} d\big(s_{(m,n,o)}, \frac{s_{(m,n-1,o+1)} + s_{(m,n+1,o-1)}}{2}\big)\,, \tag{7}$$

where $s_{(m,n,o)}$ is the color of the sample with barycentric coordinates (m, n, o) and d is the L_2 distance in corresponding color space. If E is less than a threshold T_{ds}, we downsample the triangle appearance and verify again at the next lower resolution. In our experiments, $T_{ds} \in [0, 80]$ for varying visual quality (with color values in $[0, 255]$).

4.2. Mesh Texture Compression

One of the important features with image textures is the ability to apply efficient compression schemes and to significantly reduce the amount of data while keeping good visual quality. In order to make mesh textures an efficient appearance modeling tool as well, we also give them this ability.

4.2.1 Strategy

Image compression is a widely studied problem for which optimal solutions have been proposed. Since mesh textures store color information in 2D (triangular) lattices, we believe that the existing schemes already provide efficient solutions with, furthermore, standard components. Most state-of-the-art lossy image compression and texture compression methods, including JPEG, ASTC or ETC, follow a block-based approach, first introduced by Delp and Mitchell with Block Truncating Coding in 1979 [5], in which they decompose the image into rectangular blocks and encode each block independently. We draw inspiration from JPEG [20] and adapt it to mesh textures. We first give an outline of our method below. The full process is then detailed in the next subsection (4.2.2).

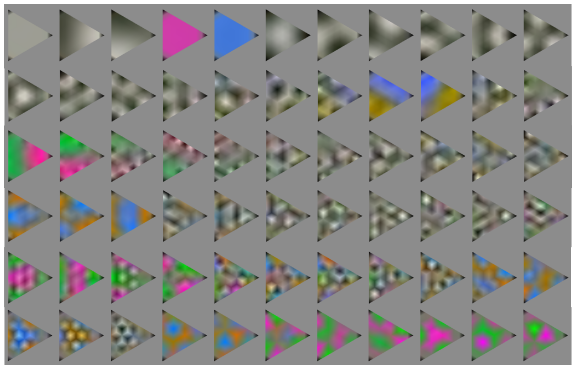

Figure 7: Example of the PCA decomposition for triangles of resolution 8 for one of our test meshes. Each triangle represents one component, from most to least relevant (*i.e.* biggest to smallest eigenvalue) in reading order. The first few components (which are most preversed by the quantization step) encode low frequency information.

Vector space – The triangles of a mesh texture are not 2D regular grids but they constitute anyway natural 'block' candidates. However, block sizes are predetermined and cannot be chosen arbitrarily: we need to be able to process blocks of varying size. Since we only allow resolution levels that are powers of two, the number of potential block sizes is limited (typically 4 or 5). We process each block size (*i.e.* each vector space) independently.

Space transform – In our case, blocks are triangular rather than rectangular and we cannot apply DCT directly. While solutions have been proposed to adapt DCT to triangular shapes (*e.g.* [6]), we opted for a different approach based on PCA decompositions, this after having tested both methods. While this transformation has little physical meaning compared to DCT, it is an effective tool for identifying the few components that encompass most of the variations in the data. Besides, we believe the main components also tend to cover low-frequency variations that are most relevant to the human visual system. (See figure 7).

Discarding irrelevant components – Similar to JPEG, we downsample chroma components. Coefficients in the PCA basis are quantized based on a specific quantization matrix.

Entropy coding – Huffman coding and run-length encoding: We choose to compute Huffman tables for each case (*i.e.* mesh), rather than use predefined ones, as in standard JPEG. We compute separate Huffman tables for each resolution level. The PCA decomposition gives us a natural writing order for coefficients (as opposed to the zigzagging needed for JPEG).

4.2.2 Encoding and Implementation.

We present here the full encoding pipeline in more details. For our compression scheme, edge colors are duplicated. While this introduces redundancy in the data, it allows us to compress edges' color information within the 2D signal of the triangles. More importantly, it completely eliminates the need for indexing between geometry and color data. Vertex colors are stored separately and left uncompressed. The encoding process is as follows:

• Triangles are reordered per resolution level R, from highest to lowest. Each set of triangles with a given resolution is processed independently.

• RGB data is transformed into YC_bC_r. Chroma components are downsampled to the next lower resolution. This means eliminating between $1/2$ (if $R = 2$) and $3/4$ (asymptotic behaviour when R increases) of chroma samples.

• A PCA decomposition is computed for each value of R.

• The coordinates of faces in this new space are quantized. We choose quantization vectors of the form $Floor(1 + ai + bi^2)$ times a normalization factor, where i is the PCA component index. i ranges from 0 for the main component, to $(D_T - 1)$, and a and b are chosen empirically (in our experiments, $a = 1$, and $b \in [0.01, 0.1]$ for varying bitrates).

• The mean vector and eigen vectors of the PCA decomposition are also quantized (*e.g.* on 12 bits per coefficient).

• Barring some implementation details, we then follow the standard JPEG pipeline, except that coefficients are not subdivided into several channels. Resolution changes are written using a specific JPEG marker. They are followed by the quantization table and the list of PCA eigenvectors (and mean), signaled by their own marker as well. Then,we write the Huffman tables used for this part of the data, and finally, the entropy-coded data.

The geometry data is written in another file. It includes a list of vertices (3D points) with one color per vertex, and a list of faces, *i.e.* three vertex indices. Faces are written following the order used in the compressed color file.

The color data of each face takes up a varying number of bytes in the encoded signal, which cannot be predicted. Thus, for rendering, data must be decoded before being loaded on memory (which is also the case with JPEG textures, but not with ETC or ASTC, for example ([17, 14]).

5. Evaluation

In order to demonstrate the benefit of mesh texture in the context of multi-view shape reconstruction, we report in this section on comparisons between image and mesh textures given reconstructed shapes and their observed images. The criteria we consider are visual quality or fidelity to the original images and size which, we believe, are the most critical properties of the appearance in our context.

We first detail the appearance function that associates a color to any point on the shape surface (section 5.1).. This function is used with both image and mesh textures in the comparisons. We then render views of our models for comparisons, both in terms of sampling (section 5.2) and compression ratio (section 5.3) and with respect to the criteria mentioned. Additional experiments with more data are included in the supplementary material.

5.1. Appearance Function

Given a mesh and its associated observed images from different viewpoints, we compute a continuous function that gives a color value for each point on the surface mesh. Various strategies can be considered for that purpose, from simple blending to super-resolution approaches. Our objective is primarily to compare appearance representations and not to evaluate appearance function, we opt therefore for an effective best view strategy that select for each mesh face the *best* view available. The key aspects are as follows:

Computing visibility: Each input image I_i is upsampled, and visibility for every (face,view) pair is computed at the sub-pixel level, which gives more precision on the available appearance information.

Assigning Views: In real acquisition setups, the camera properties are inaccurate and the input views are consequently misaligned and inconsistent. Averaging pixel information over different images is therefore likely to severely blur the appearance. Following many works in that respect, *e.g.* [11, 7, 22], we assign a single view to each face of the shape model. In practice, we select the most informative view with the largest number of visible subpixels.

Blending views: The view assignment can be seen as weighing the input views by a function ω_0 that depends on the position on the surface and the view index. Choosing a view j for face $\mathcal{T}$ means

$$\forall i \in \{1, ..., I\}, \forall \mathbf{p} \in]0, 1[^2 \,,$$
$$\omega_0(\mathcal{T}, \mathbf{p}, i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \,. \end{cases} \quad (8)$$

To avoid visible seams between triangles, we compute an estimate of ω_0 for each vertex by averaging its value on adjacent faces. Weights are then interpolated smoothly between vertices, *i.e.* for the image i, a face $t = \{v_1, v_2, v_3\}$, a location $\mathbf{p} = (a, b, c)$ within t, we use the function ω defined by:

$$\omega(x(t, \mathbf{p}), i) = a\, \omega_0(v_1, i) + b\, \omega_0(v_2, i) + c\, \omega_0(v_3, i) \,. \quad (9)$$

Finally, the color C of each surface point is defined by

$$\forall x \in \mathcal{M}, C(x) = \sum_{i=1}^{I} \Big(I_i\big(\pi_i(x)\big) \times \omega(x, i) \Big) \,, \quad (10)$$

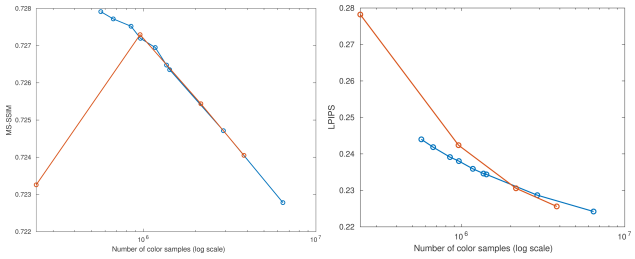

Figure 8: Result on the test mesh: score averaged over all viewpoints. (red) Image textures, (blue) Mesh textures.

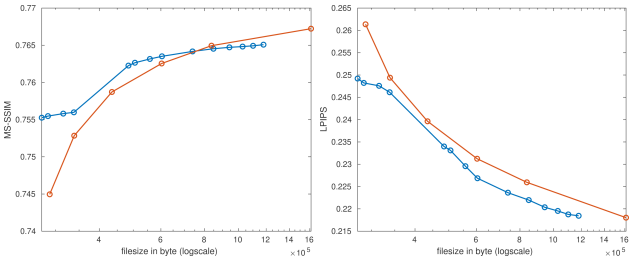

Figure 9: Result on the test mesh: score averaged on different viewpoints. Image textures shown in red, mesh textures in blue. The original image texture takes 5.6 MB (3072×3072) for 2940k texels. The original mesh texture takes 5.5 MB, for 3007k color samples.

where $\{\mathcal{I}_1 \dots \mathcal{I}_I\}$ are the input images with their respective 3D-2D projection operators $\{\pi_i\}$.

5.2. Sampling

We evaluate our method on real data, captured with multi-camera platforms. The input data and the reconstructed mesh are provided by the authors of [12]. It provides 64 different input views of the same scene, and includes an image texture, computed with a method based on conformal maps. We use a leave-one-out evaluation strategy, removing a given input view, and comparing it against the projection of appearances computed using the remaining views. We sample the function C defined in equation (10) with varying parameters. Charts in image textures are dilated to prevent artifacts and for a fair comparison with mesh textures. We use two different metrics for such comparison: the Multi-scale Structural Similarity (MS-SSIM) from [21], and the more recent Learned Perceptual Image Patch Similarity (LPIPS) from [25]. We compare images within the mesh silhouette only in the images. Figure 8 shows some numerical results. MS-SSIM measures a similarity, thus higher scores are better, contrary to LPIPS which measures the perceptual difference. In this case, MS-SSIM seems ill-suited for a meaningful analysis, given the very small range of variation. At high resolution, sampling is not a limiting factor, and the error is mostly due to the imperfect geometry and color function C. As we progressively decrease the number of samples, LPIPS shows our representation tends to retain more visual information.

5.3. Compression Evaluation

To evaluate our compression method, we pick a high-resolution texture image and texture mesh of the same mesh, with a similar number of samples and a similar score on both metrics. We compress them with varying quantization matrices (or compression ratio for JPEG), and compare the results, using the same process as in the previous section, except that this time, we use a single model that we backproject on different input views. Figure 9 demonstrates that our representation outperforms the image texture, especially at low bitrates. Figure 2 displays the artifacts inherent to both methods. If we zoom in, we can already notice small discontinuities with the image textures, because of seams in the texture in that case. As we compress the texture with increasingly high compression ratios, blocky artifacts start to appear along the seams, and finally, on the whole mesh. By comparison our mesh texture method yields triangular blocky artifacts that seem less perceptible, probably because they follow an irregular pattern. Besides, our PCA decomposition basis is computed specifically for the mesh, contrary to the more general DCT decomposition.

6. Conclusion

We have studied the benefits of representing color information directly on 3D meshes in the context of multi-view image-based appearance modeling. More specifically, we have compared the use of our proposed mesh texture pipeline, with the more widespread use of texture maps. First, taking advantage of its sampling flexibility over the surface triangles, we were able to formulate a smart locally-adaptive sampling strategy, which we show to be more efficient than uniform sampling in the image-based appearance modeling context. Second, we introduce a novel compression strategy dedicated to our mesh-embedded color information, and show that it is effective enough to outperform classic texture storage in terms of the perceptual fidelity vs storage capacity tradeoff, especially at low bit rates. The results obtained validate this scheme as a practical solution, even when dealing with large amounts of data. For future work, it would be interesting to implement compression strategies with a fixed bit rate, so that meshes could be rendered directly from their compressed mesh texture. Additionally, one could try to build temporally consistent mesh textures, to represent the appearance of a moving object across several frames in a compact way. Finally, we would like to combine this framework with a state-of-the-art superresolution algorithm, to see if additional quality performance can be achieved with this representation.

References

[1] A. Baumberg. Blending Images for Texturing 3D Models. *Bmvc*, pages 404–413, 2002. 3

[2] A. C. Beers, M. Agrawala, and N. Chaddha. Rendering from Compressed Textures. *Siggraph 1996*, page 4, 1996. 3

[3] B. Burley and D. Lacewell. Ptex: Per-face texture mapping for production rendering. *Computer Graphics Forum*, 27(4):1155–1164, 2008. 2

[4] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics*, 34(4):69:1–69:13, 2015. 3

[5] E. J. Delp and O. R. Mitchell. Image Compression Using Block Truncation Coding. *Communications, IEEE Transactions on*, 27(9):1335–1342, 1979. 6

[6] J. J. Ding, Y. W. Huang, P. Y. Lin, S. C. Pei, H. H. Chen, and Y. H. Wang. Two-dimensional orthogonal DCT expansion in trapezoid and triangular blocks and modified JPEG image compression. *IEEE Transactions on Image Processing*, 22(9):3664–3675, 2013. 6

[7] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, and D. Cohen-Or. Seamless montage for texturing models. *Computer Graphics Forum*, 29(2):479–486, 2010. 7

[8] B. Goldlücke, M. Aubry, K. Kolev, and D. Cremers. A super-resolution framework for high-accuracy multiview reconstruction. *International Journal of Computer Vision*, 106(2):172–191, 2014. 3

[9] B. Goldluecke and D. Cremers. Superresolution texture maps for multiview reconstruction. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1677–1684, 2009. 3

[10] T. Lambert. From 2D to 3D painting with mesh colors. *ACM SIGGRAPH 2015 Talks on - SIGGRAPH '15*, pages 1–1, 2015. 2

[11] V. Lempitsky and D. Ivanov. Seamless mosaicing of image-based texture maps. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007. 7

[12] V. Leroy, J.-S. Franco, and E. Boyer. Shape reconstruction using volume sweeping and learned photoconsistency. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1, 8

[13] D. J. Newman. The hexagon theorem. *IEEE Transactions on Information Theory*, 28:1398–1402, 1982. 4

[14] J. Nystad, A. Lassen, A. Pomianowski, S. Ellis, and T. Olson. Adaptive scalable texture compression. *ACM SIGGRAPH / Eurographics conference on High-Performance Graphics*, pages 105–114, 2012. 3, 7

[15] J. Peng, C. S. Kim, and C. C. Kuo. Technologies for 3D mesh compression: A survey. *Journal of Visual Communication and Image Representation*, 16(6):688–733, 2005. 4

[16] C. Rocchini, P. Cignoni, and C. Montani. Multiple textures stitching and blending on 3D objects. *Proceedings of the 10th Eurographics conference on Rendering*, pages 119–130, 1999. 3

[17] J. Ström and T. Akenine-Möller. i PACKMAN: high-quality, low-complexity texture compression for mobile phones. *ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, pages 177–182, 2005. 3, 7

[18] M. Tarini, C. Yuksel, and S. Lefebvre. *Rethinking Texture Mapping*, volume 1691. 2016. 2

[19] V. Tsiminaki, J.-S. Franco, and E. Boyer. High Resolution 3D Shape Texture from Multiple Videos. *Proceedings ofIEEE Conference on Computer Vision and Pattern Recognition*, pages 1502–1509, 2014. 3

[20] G. K. Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992. 6

[21] Z. Wang, E. Simoncelli, and A. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. IEEE, 2000. 8

[22] L. Xu, E. Li, J. Li, Y. Chen, and Y. Zhang. A general texture mapping framework for image-based 3D modeling. *Proceedings - International Conference on Image Processing, ICIP*, pages 2713–2716, 2010. 7

[23] C. Yuksel. Mesh color textures. In *Proceedings of High Performance Graphics on - HPG '17*, pages 1–11, New York, New York, USA, 2017. ACM Press. 2

[24] C. Yuksel, J. Keyser, and D. H. House. Mesh colors. *ACM Transactions on Graphics*, 29(2):1–11, 2010. 2, 4, 5

[25] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 8