Multi-Resolution Texture Coding for Multi-Resolution 3D Meshes

David Fuentes Sánchez, Rafael Pagés Scasso and Francisco Morán Burgos
Grupo de Tratamiento de Imágenes, Universidad Politécnica de Madrid, Spain
Email: dfs@gti.ssr.upm.es

Abstract—We present an innovative system to encode and transmit textured multi-resolution 3D meshes in a progressive way, with no need to send several texture images, one for each mesh LOD (Level Of Detail). All texture LODs are created from the finest one (associated to the finest mesh), but can be reconstructed progressively from the coarsest thanks to refinement images calculated in the encoding process, and transmitted only if needed. This allows us to adjust the LOD/quality of both 3D mesh and texture according to the rendering power of the device that will display them, and to the network capacity. Additionally, we achieve big savings in data transmission by avoiding altogether texture coordinates, which are generated automatically thanks to an unwrapping system agreed upon by both encoder and decoder.

I. INTRODUCTION

Most of the research in MR (Multi-Resolution) representation and coding of 3D meshes has focused to date on the shape of “naked” meshes, i.e., with no textures or other attributes (colors, normals, etc.) attached to them. The first steps in this field were based on bare mesh simplification, and good examples of this are the methods of Schroeder et al. [18], and of Soucy and Laurendeau [20], which are very similar, and similarly useless for MR environments, because their simplifications cannot be undone. It was Hoppe who first introduced the concept of progressive mesh [9], together with the associated and reversible edge collapse vs. vertex split operations. After this, first Garland [8] and later Hoppe himself [10] improved greatly Hoppe’s initial method, both conceptually (Garland’s quadric error metric remains a key idea and tool for mesh simplification) and from an algorithmic efficiency viewpoint.

Later still, many other researchers have designed and reported more efficient techniques for the simplification and/or compression of “naked” 3D meshes: it is worthwhile mentioning, among others, Cohen-Or’s patch-based simplification method [4], Khodakhovsky’s wavelet-based scheme [11], Alliez’s equalization of vertex valence around six (the “regular” one for triangular meshes) [1], and Gandoin’s system [7], based only in exploiting the geometry properties, and later improved by Peng [16].

However, we have been only able to find useful ideas about the progressive coding of textured 3D meshes in a paper by Sander et al. (incl. Hoppe) [17] where they present a technique allowing to map one single texture image onto all the different LODs (Levels Of Detail) of a progressive mesh. They cleverly partition the mesh into charts, parameterize every chart, and pack all of them to form one single texture atlas, common to every mesh LOD. Unfortunately, this means that the full information about that common texture is needed to render even the coarser mesh LODs, which seems an overkill.

The system we propose uses instead different texture LODs for the successive mesh LODs, so the coarsest mesh LOD only needs the coarsest (and “cheapest” to be stored or transmitted) texture LOD to be rendered. Of course, our texture LOD set is coded progressively so, given a specific LOD, only some delta information is needed to recover the immediately finer one. And thanks to a lossless mesh unwrapping algorithm agreed upon by both encoder and decoder [14], we skip altogether the coding of texture coordinates, which are automatically generated with the sole knowledge of the mesh shape.

Its unwrapping algorithm being essential for our system, we feel the need to review briefly different methods for 3D mesh unwrapping. The vast majority, if not all of the most widely used systems introduce some distortion when unwrapping the mesh to create a parametrization of it. Examples of this are: the classic reference in this field, which is probably Floater’s method to smoothen a surface thanks to a shape-preserving parametrization [5]; Lee’s MAPS [12], which was not explicitly designed to unwrap meshes but can be used with this purpose; and Lévy’s method to generate atlases [13] based on Cauchy-Riemann’s equations.

The next Section gives an overview of our system, that we have proposed to and is being considered by MPEG (Moving Picture Experts Group; formally ISO/IEC JTC1/SC29/WG11) within its core experiments on MR3DMC (Multi-Resolution 3D Mesh Compression) [6]. It mainly consists of two combined processes, texture adaptation and skin transplant, that we explain in Sections III and IV respectively. In Section V we elaborate on the two modes we have conceived and explored so far for using our system, and report on the results we have obtained. Finally, Section VI concludes our presentation.

II. PROPOSED TECHNIQUE: OVERVIEW

The main goal of our system is to efficiently code textured MR meshes so that there is no need to build and transmit a texture LOD for every mesh LOD. For the sake of clarity, we focus our explanation below on a simple case with only two LODs, but it remains valid for any number of them.

Our inputs, highlighted in red on the leftmost end of Figure 1, are a “dressed” fine 3D mesh and its corresponding fine texture, and a “naked” coarse mesh (with no texture). This naked coarse mesh must be obtained at the encoder end...
through some mesh simplification algorithm and must reach the decoder, probably thanks to some MR 3D mesh coding and transmission scheme that will also make the fine mesh eventually available at the decoder. But naked 3D mesh LOD extraction and coding, although related to our work, are not discussed in this paper. What we do explain instead is how we obtain in the encoder (i) the coarse texture to be transmitted to the decoder, to reconstruct the dressed coarse mesh, and (ii) the texture refinement to be equally transmitted later if it is indeed sensible to reconstruct the dressed fine mesh.

Our most innovative idea consists in using the same mesh unwrapping algorithm in both encoder and decoder [14] to create an adapted texture atlas, thus avoiding the need to transmit any texture coordinates information at all. We could use any unwrapping algorithm providing always identical texture atlas masks when unfolding a given mesh, as Sorkine [19] did, but ours has the advantage of not introducing any distortion.

This texture adaptation process is applied to all texture LODs, starting with the finest, that we will eventually reconstruct at the end of the decoding process. So we unwrap the fine mesh LOD and create the corresponding adapted fine texture atlas. Then, we create the texture atlas for the coarse mesh by unwrapping it as well and, for every pixel \( p_c \) in the coarse texture atlas, we find out the exact position of the 3D point \( P_c \) on the coarse mesh associated to the center of \( p_c \), and perform a “corresponding color seek” (see Section IV) to find the 3D point \( P_f \) on the fine mesh whose color (in fact, that of its corresponding fine texture pixel \( p_f \)) should be “transplanted fine to coarse” (i.e., copied) to \( p_c \).

Once this is done for every pixel in the coarse texture atlas mask, the whole dressed coarse mesh is ready to be transmitted. It is worth noticing again that we do not need to transmit any texture coordinates nor any other texture data but the image itself, as we can automatically generate that information in the decoder.

After this, and assuming that the naked fine mesh has been received, we reconstruct its texture, this time thanks to a "coarse to fine skin transplant". Doing this for every pixel in the texture which will be mapped onto the fine mesh will give us an image with the exact same mask that the adapted fine texture had, but with less details (since the pixels have been taken from a coarser texture), so the texture refinement data is needed to recover a finer quality.

### III. Texture Adaptation

The texture adaptation process consists of three substeps:

1) **New texture atlas size calculation.** This is the most critical step in our whole process, because it is the one where some distortion can be introduced. We adjust the dimensions (number of pixels) of the adapted texture atlas and, through an input factor called \( \text{scale} \), we control its quality. Of course, the better the quality we require, the worse the compression ratio we achieve.

2) **Mesh unwrapping and new texture atlas mask creation.** This step is fully accomplished by our unwrapping algorithm [14], which provides a pixel mask to be filled in with colors later. Among other data, it provides a matrix with the same dimensions as the texture atlas, telling us which triangle every pixel belongs to.

3) **New texture atlas coloring.** Once we have the mask from the unwrapper, and the topological and geometric information of the 3D mesh, for every pixel in the mask, we calculate its barycentric coordinates in the triangle it belongs to and obtain thanks to them the color that the corresponding pixel had in the original (fine) texture, and copy it onto our new (coarse) texture. Finally, to avoid seams across patches when rendering the model, a dilation process is performed on the new texture atlas.

### IV. Skin Transplant

Skin transplant is the core process of the codec system we propose. In it, a texture atlas is generated for a naked mesh from a dressed one, thanks to which it is possible to transplant/copy its skin/texture to the first mesh. Below we focus on the fine
to coarse skin transplant that takes place at the encoder, but it can be easily understood that the same reasoning is applicable for the coarse to fine transplant performed by the decoder.

Conceptually, this process consists of the same three substeps as the one above: 1) New texture atlas size calculation; 2) Mesh unwrapping and new texture atlas mask creation; and 3) New texture atlas coloring. We elaborate on the third step only because the first two steps are almost identical to those of the texture adaptation process, except for the following detail in 1): instead of taking as an input parameter the scale factor, for the skin transplant we use a combination of another input factor named child and the ratio between the number of triangles of the recipient vs. donor meshes, so that smaller texture atlases are generated for coarser meshes.

It is easy to think that finding the color for a pixel $p_c$ in the coarse texture is as simple as unprojecting it onto the corresponding 3D point $P_c$ on the coarse mesh, then finding the closest 3D point $P_f$ in the fine mesh (which is a pretty involved and time consuming problem in itself), and finally copying the color information of the corresponding fine texture pixel $p_f$ to $p_c$. This is partly true but, as our textures are calculated to work with true size triangles, with zero distortion, so that the texture is later projected onto them in a perfectly orthogonal way, it is important to redefine closest to mean, instead of “the one with the smallest Euclidean distance”, “the intersection of a ray, cast from $P_c$ in the local normal direction to the coarse mesh, and the fine mesh”.

So what we actually do in this third substep of the skin transplant process is the following: for every pixel $p_c$ in the coarse texture, we find the corresponding $P_c$, we cast a ray in the normal direction to the coarse mesh triangle on which $P_c$ lives, and we test every fine mesh triangle for intersection thanks to Badouel’s efficient algorithm [3].

V. WORKING MODES AND RESULTS

There are two obvious ways to use our system in a MR textured 3D mesh transmission context, depending on how the coarser and coarser texture LODs are extracted from the finest, original one. We have named them “Palm mode” and “Michelin mode” for reasons that should be easy to understand by looking at Figure 2.

A. Palm Mode

We designed this operation mode for our system so that all texture LODs are generated from the original, finest one. Despite having been conceived for MR environments, the Palm mode turns to be, according to our tests, very convenient for the simple progressive coding of the finest LOD. Indeed, as we decrease the quality of the lower LODs, the sizes of the files containing their meshes and textures drop quickly, but the corresponding texture refinements do not increase their size almost at all, and their encoding and decoding times do decrease drastically, since both operations take 20-50 ns per texel and receiving triangle mesh, which amounts to a total of 7 s for encoding and 56 s for decoding for the 29184 triangle GyeongJu Drum model shown in Figure 3 (CPU: Intel Core 2 Duo @ 2.4 GHz; RAM: 4 GB). This is why we do not care much about the lower LOD qualities: as long as the coarser texture LODs have some detail, the size of the refinement textures is never close to that of the original texture.

B. Michelin Mode

In this second operation mode, every texture LOD is generated from the next finer one. The Michelin mode is ideal for MR environments where the lowest LOD must be quickly transmitted in the first place, and then refined upon request until the finest LOD is received or until the decoder decides that some intermediate LOD is enough given the network bandwidth or the terminal display resolution or computational/rendering power. In Michelin mode, encoding and decoding times for each LOD step are considerably lower, because only a fraction of the number of triangles involved in the loop of the third substep in the skin transplant process is considered at a time. Besides, as the refinement texture size for each LOD step is in relation to that of the previous one, we avoid transmitting big refinement textures every time. However, if the final goal is to recover the finest LOD, since we need to go through all intermediate LODs, the bit savings are logically lower than with the Palm mode, and the decoding times larger.

C. Bit Savings

We show subjective quality results in Figure 3, and summarize below the objective bit savings yielded by our system. To approximate the raw (i.e., uncompressed) size of a dressed mesh with $V$ vertexes, $T$ triangles, $C$ pairs of texture coordinates and $R$ references to them, we have reasoned like Avilés [2], who established the raw size of a naked triangular mesh with $V$ mostly regular vertexes as $S_n \approx 96 + 6 \log_2 V$ bvp (bits per vertex). We have assumed that, in typical dressed meshes like the ones we have used to test our system, $C \approx 3V$ and $R = T \approx 2V$, and concluded that the raw size of a large and mostly regular triangular textured mesh with $V$ vertexes is $S_d \approx 288 + 6(\log_2 V) + \log_3 2V$ bvp.

With respect to this raw size, the average compression rate we have achieved in Palm mode is 60% for a final texture loss of 5 (out of 255), in terms of ITEM-RMSE [15], which is a more reliable, objective error metrics than the plain RMSE for textured 3D meshes. For this objective error value,
the reconstructed dressed mesh is indistinguishable from the original, unless drastically zoomed in. Nevertheless, if we want an almost lossless reconstruction for which even small errors are not easily detectable for the human eye, it is possible to bring the error down to 2.5 and still obtain some bit savings (around 20%).

VI. CONCLUSIONS
The MR codec for textured 3D meshes presented in this paper is an innovative system to transmit the texture LOD set associated to a mesh LOD set in a progressive way. Our system is being considered by MPEG within the core experiments on MR 3D mesh compression. It could be especially useful in a framework including portable devices, such as cell phones, which are much less powerful than desktop and laptop computers, and make it thus pointless to try and download always high quality textured 3D meshes.

Our system is based on an unwrapping algorithm which makes it completely unnecessary to code or transmit texture coordinates, thus yielding significant bit savings. So far, we have designed and analyzed two operation modes for our system, named Palm and Michelin, depending on how the coarser texture LODs are extracted from the finest, original one. By choosing one or the other, it is possible to achieve a trade-off between bitstream size efficiency, which is best in Palm mode, and LOD granularity, which is best in Michelin mode, since it allows to adjust more gradually the texture LOD/quality according to the rendering terminal power and network capacity. It could be worthwhile to design and analyze other hybrid working modes.

ACKNOWLEDGMENTS
The authors are very thankful to Sergio Arnaldo Duart for his contributions to their work, which has been partially supported by the Ministerio de Ciencia e Innovación of the Spanish Government under project TEC2010-20412 (Enhanced 3DTV).

REFERENCES