ITEM: Inter-Texture Error Measurement for 3D Meshes

Rafael Pagés, David Fuentes and Francisco Morán*
Grupo de Tratamiento de Imágenes
Universidad Politécnica de Madrid

Abstract

We introduce a simple and innovative method to compare any two texture maps, regardless of their sizes, aspect ratios, or even masks, as long as they are both meant to be mapped onto the same 3D mesh. Our system is based on a zero-distortion 3D mesh unwrapping technique which compares two new adapted texture atlases with the same mask but different texel colors, and whose every texel covers the same area in 3D. Once these adapted atlases are created, we measure their difference with ITEM-RMSE, a slightly modified version of the standard RMSE defined for images. ITEM-RMSE is more meaningful and reliable than RMSE because it only takes into account the texels inside the mask, since they are the only ones that will actually be used during rendering. Our method is not only very useful to compare the space efficiency of different texture atlas generation algorithms, but also to quantify texture loss in compression schemes for multi-resolution textured 3D meshes.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: 3D mesh, texture, error, measure, RMSE, compression

1 Introduction

No one can deny that this is one of the sweetest moments for 3D graphics. Even if computers have been able to produce 3D content for decades, the growth of this engineering branch has been exponential in the last years. 3D graphics are now ubiquitous in many professional domains, such as the security or health industries, but even more in (arguably) "less serious" fields, like video games, cinema, or entertainment in general. In many of those domains, textured polygonal (typically, triangular) 3D meshes are used to model 3D objects, and then compressed seeking efficiency, since they have to be stored and/or transmitted, e.g., for an on-line computer game, or for a terrain/city fly-over application.

In the last two decades, a reasonable maturity has been achieved in what concerns 3D mesh *shape* compression, thanks to substantial research on two related topics. On one hand, many clever methods have been devised and published for the efficient coding of both the geometry and the topology of a static, "naked" 3D mesh (i.e., the positions and connectivity of its vertices) [Avilés and Morán 2008]. On the other hand, there is a good understanding and (up to a certain extent) a common agreement on how two 3D meshes should be compared [Aspert et al. 2002][Cernea et al. 2010].

However, research has been much less active with respect to the compression of *textured* 3D meshes. More precisely, there has been much less effort on understanding and measuring the error incurred

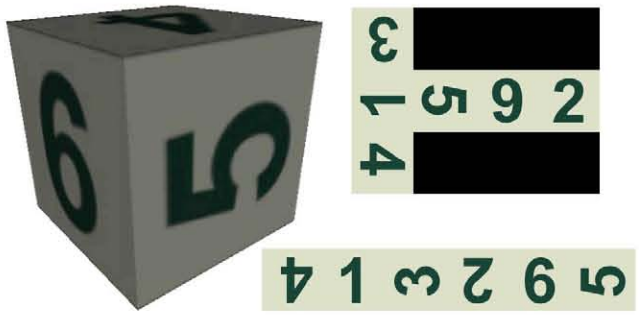

Figure 1: *Textured cube representing a dice, and two of the many possible texture atlases for it.*

when coding textured 3D meshes: for the vast majority of them, far many more bits must be devoted to coding their raw textures than their raw shapes, so textures are usually aggressively compressed by means of some lossy image coding scheme (e.g., according to the JPEG standard [JPEG 1994]) so they may be more compactly stored or transmitted through the Internet. But traditional metrics to objectively quantify the error associated to that lossy scheme, say, the RMSE (Root Mean Square Error) between the original and reconstructed images, are not enough to characterize the texturing error, since texture coordinates may introduce non-linearities in the process of 2D-3D mapping. For instance, in a texture atlas designed by an artist to be wrapped onto the 3D mesh of a virtual humanoid, the portion corresponding to its face may take many more texels than those to be mapped on its torso or legs, because it makes sense to represent with more details the body parts which are more critical for the subjective quality perceived by a human.

Besides, as suggested by Figure 1, even if texture coordinates are not used to privilege certain 3D triangles in the mesh with respect to others, texture atlases are frequently composed by a set of "islands" (which we will call "patches" below), containing real color information, within a "sea" of meaningless neutral texels, which are in fact not used at rendering time, given the interpolations induced by texture coordinates. Below, we will refer to that set of islands in a texture atlas as its *mask*. In the many cases where the number of "net" texels in the mask of a texture atlas is well below the total number of "gross" texels, the classic RMSE definition for rectangular images yields a misleading impression of having a difference between the texture atlases that is much smaller than it actually is.

Furthermore, as suggested as well by Figure 1, traditional methods for comparing rectangular images, RMSE-based or not, are useless for comparing texture atlases with completely different mask shapes. There are plug-ins for 3D modeling software packages which perform a rather good job of semi-automatically minimizing the ratio of net/gross texels, i.e., for generating colorful islands that may be packed as efficiently as possible, and with as little human intervention as possible, in a neutral sea. But they output different masks, and are all based on methods that introduce some distortion in the 2D-3D mapping, so there is no easy way of comparing the results of any two, since there is no easy way to determine the error we make when we transform a given texture atlas into another.

As a first step towards our final goal of comparing two textured 3D meshes, we wonder: *Is it really impossible to compare two different texture atlases to be mapped onto the same naked 3D mesh?* Answering this question is the aim of the system we describe below, which is based on first building two new texture atlases, corresponding to each of the two input ones, but with the same dimensions and masks, and then calculating a slightly modified version of the RMSE between them. To the best of our knowledge, our system is unique in making it sensible to use the (almost standard) RMSE to compare two texture atlases for the same 3D mesh, and we believe it may be specially useful to evaluate the objective quality results of textured 3D mesh compression techniques, where it is necessary to quantify texture losses.

2 Proposed technique: overview

As shown in Figure 2, our system consists of three main steps:

1. We unwrap the mesh with a zero-distortion method, which makes the image comparison to be performed in the last step reliable. With "zero-distortion" we mean that the angles of every 2D triangle are exactly the same as those of its 3D counterpart, and relative triangle sizes are preserved exactly – for the moment, all vertex coordinates, both in 3D and 2D, are expressed with floating point numbers.

2. Here is where we deal for the first time with texels and integer coordinates, and guarantee that every texel in each of the two original textures is duly represented (by one or more texels) in each of the corresponding new, adapted textures. To this end, for each (single) 3D triangle t_i, we calculate the ratios r_{i1} and r_{i2} of the real area of t_i vs. the (two, in principle different) numbers of texels it covers in each original texture. We then build one (single) new virtual texture atlas whose dimensions are determined to guarantee the surjectivity even in the case of the most restrictive ratio, r_{i1} or r_{i2} (step 2a in Figure 2). Finally, we fill in the texels of the new adapted texture atlases by emulating what any renderer would do (step 2b).

3. We now have two new texture atlases with identical dimensions and masks, but different (although typically very similar) color information. We only have to measure that difference, and we do so with a slightly modified version of the classic RMSE, that we have called "ITEM-RMSE".

3 Previous work

Although the problem we want to solve has apparently not been researched too thoroughly, there are several ways to evaluate the shape error between two naked 3D meshes, among which the most widely accepted as a reference is probably MESH [Aspert et al. 2002], a system to approximate the Hausdorff distance between the surfaces of both meshes. Even if our aim is just to estimate the difference between two textures applied to the same 3D mesh, and not to determine the combined shape and texture error between two 3D meshes, it could be interesting to address in the future the latter, more general problem.

Our method starts with the unwrapping of the 3D mesh. There are different algorithms to accomplish this task, but most of them are based on the use of a certain parametrization obtained by solving a linear equation system. One of the most widely accepted systems of this kind is Floater's [Floater 1997], which uses a shape-preserving parametrization to obtain visually smooth approximations. Khodakovsky and co-workers developed another system to cut the 3D mesh into portions which are then translated into 2D patches, and optimize these patches with respect to metric distortion

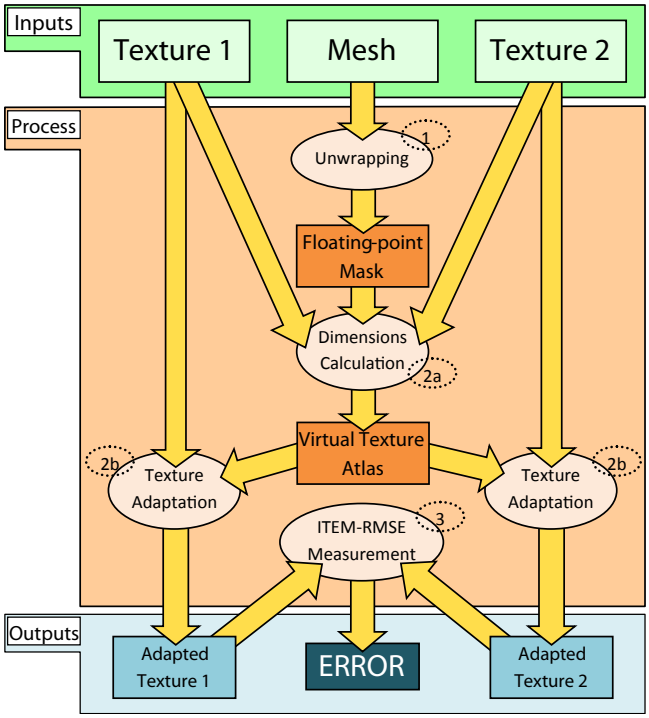

Figure 2: *Schematic overview of our system.*

or shape quality [Khodakovsky et al. 2003]. Also, Lévy et al. presented a method for generating texture atlases with a parametrization algorithm based on the least squares approximation of Cauchy-Riemann's equations [Lévy et al. 2002]. Finally, there exist other methods such as Lee's MAPS [Lee et al. 1998], which was in fact designed for 3D mesh simplification (as opposed to 3D mesh unwrapping), but whose parametrizations could also be used for our purposes. All these techniques have the same common problem: even though their results are quite efficient, they introduce some distortion in the texture atlas created for coloring the 2D patches. We, instead, use a simpler and faster method of ours originally devised for the creation of multi-texture atlases [Pagés et al. 2010]. The main advantage of our method is that it avoids distortion altogether, thus making the task of comparing two images very straightforward.

One could argue that the comparison between texture images should use some perceptual metric such as: *i)* Teo's psychophysics-based measure [Teo and Heeger 1994] which takes into account the characteristics of the HVS (Human Visual System), exploiting its sensitivity to contrast and tendency to pattern detection; *ii)* Beghdadi's non-redundant wavelet decomposition system [Beghdadi and Pesquet-Popescu 2003], where HVS's nonlinearities in terms of spatial frequency components are seized using Wigner-Ville's distribution [Iordache and Beghdadi 2001]; or *iii)* Wang's hybrid (perceptual plus analytic) quality index [Wang and Bovik 2002], which makes use of three different factors: loss of correlation, luminance distortion and contrast distortion. However, due to the nature of our adapted texture atlases, which are composed by a big number of 2D patches that have plenty of artificial borders, we opted to use a strictly objective metric, since we are only interested in the net data inside patches, and in checking, texel by texel, whether the colors in the two atlases match.

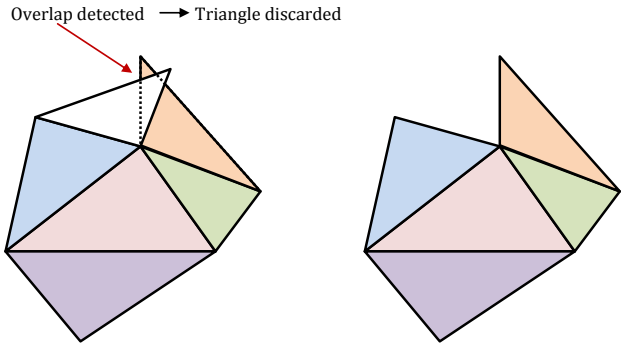

Figure 3: *Avoiding overlaps while unwrapping the 3D mesh.*

4 Proposed technique: detailed explanation

4.1 Unwrapping the 3D mesh

As summarized in Section 2, to obtain two texture atlases equivalent to the input ones, but with the same mask, we start by unwrapping the 3D mesh into 2D patches thanks to an algorithm which does not introduce any distortion, and which we developed originally for a previous system of ours [Pagés et al. 2010]. It performs a piecewise translation of the 3D mesh onto the plane, therefore preserving exactly the original angles and proportional sizes of the 3D triangles.

To create these 2D patches, a triangle adjacency list is initially built to simplify the navigation across the topology of the 3D mesh. For the first patch, we start by placing a random seed triangle on the plane, and recursively place its neighbors in the corresponding adjacent positions (still making all our calculations with floating point coordinates), always checking for overlaps among triangles. We can detect potential overlaps because we keep track of the edges forming the perimeter of the current patch, so candidate triangles are easily found to be conflicting if their edges would intersect any current perimeter edge. If such a conflicting triangle is found, it is discarded for that patch and left for a new one (see Figure 3). New patches are created as soon as all candidate triangles for the current patch are conflicting.

This unwrapping algorithm, that we have tested with hundreds of standard 3D meshes, does not require them to be regular, or closed, or even manifold. In fact, it may be fed as well with meshes with intersecting triangles, and only complains about flat triangles and repeated vertices.

This system tends to yield rather large texture atlases, but we do not care much about the output dimensions, since we do not need to store the two adapted texture atlases. Nevertheless, we want to be orderly and as efficient as possible, so it is important to pack the 2D patches on the texture canvas in an intelligent way. Packing the (rectangular) bounding boxes of the 2D patches instead of their irregular, star-like shapes reduces the complexity of the problem. This new, simpler (but still very complex) problem is commonly known as "block/Tetris packing" or, more technically, as "NP-hard pants packing", and has been very well studied, and even used for texture atlases creation [Sander et al. 2001]. We have implemented a simplified version of Murata's algorithm [Murata et al. 2002], where instead of optimizing both dimensions, we fix one and optimize the other to save some processing time (see Figure 4).

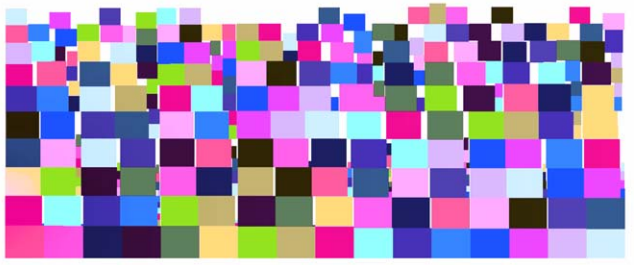

Figure 4: *Efficiently packing the bounding boxes of the 2D patches.*

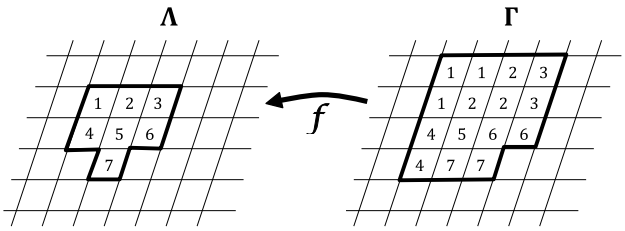

Figure 5: *Surjective function f mapping lattice Γ onto lattice Λ.*

4.2 Drawing the two adapted texture atlases

Since texture atlases provided by artists may have an irregular distribution of the texels, as mentioned in Section 1, some 3D mesh portions with smaller triangles may end up being covered, by means of texture coordinates, by larger areas in the texture atlas: see Figure 8. We must thus establish first a correct correspondence among the texels of the original texture maps and those of the virtual texture atlas, which will then enable us to build the two adapted, and easily comparable texture atlases. But before moving from the floating point coordinates of 3D or 2D vertices (of the triangles of the 3D mesh or of the 2D patches) to the integer coordinates of the texels in our virtual or adapted texture atlases, we need to ensure that every texel of the original texture atlas is represented by at least one texel in the texture atlas, so there is no loss in this process. This is why we first build a single texture atlas mask, still in floating point precision. Moreover, we must find which net texels will actually be used in the rendering process by projecting each 3D triangle onto each original texture map with the corresponding texture coordinates. Only then can we formulate mathematically our surjectivity requirement as follows.

If an original texture atlas has $m \times n$ net texels and our virtual texture atlas has $p \times q$ ($m \times n \leq p \times q$), we can define both textures as lattices Λ and Γ of dimensions $m \times n$ and $p \times q$ respectively:

$$\Lambda = \{(i,j) : i \in \{1,...,m\}, j \in \{1,...,n\}\}$$

$$\Gamma = \{(x,y) : x \in \{1,...,p\}, y \in \{1,...,q\}\}$$

We need to guarantee that, for every (i,j) in lattice Λ, there is at least one value (x,y) in lattice Γ which corresponds to (i,j). We do so by defining a function f to map the first lattice onto the second, $f : \Gamma \rightarrow \Lambda$, and ensuring that f is a surjective function (just injective in case the dimensions of both lattices match: see Figure 5), i.e.:

$$\forall (x,y) \in \Gamma \;\; \exists\, (i,j) \in \Lambda : f((x,y)) = (i,j)$$

We then calculate the ratios r_{i1} and r_{i2} of the real area of each 3D triangle t_i vs. the (two, probably different) numbers of texels it covers in each original texture. The largest ratio determines the size of the corresponding 2D triangle in the virtual texture atlas. In this way, we guarantee that all texels in both input texture maps are duly represented in the output texture map, even for the triangle with the highest texel resolution. This method works perfectly even for very inhomogeneous texture atlases, in which more texel resolution is assigned to certain 3D mesh portions, even if the corresponding 3D triangles are smaller than the rest.

To finalize the texture adaptation process, we determine the integer dimensions of the virtual texture atlas thanks to the ratios above, and we fill in the texels of each of the two adapted texture atlases, which will have (probably only slightly) different color information, but identical masks, and will hence be very easy to compare. To obtain this color information for, say, the first adapted texture, we use the first set of texture coordinates, which point to certain texels in the first original texture atlas. If we want to emulate what a sensible renderer should do, we must use bilinear interpolation between texels to avoid pixeling ("texeling"?) noise in the textured 3D mesh. But we may also want to emulate what a "quick'n'dirty" renderer would do, and use nearest neighbor replication, so that one texel in the original texture is simply copied several times in the adapted texture. We have implemented both choices, and report on the obtained results below.

4.3 Measuring the error (ITEM-RMSE)

Once we have both adapted texture atlases, we are ready to measure very easily the error between them. We have chosen to discard subjective/perceptual metrics for image differences, such as the ones referenced at the end of Section 3, and use a slightly modified version of the RMSE, which is universally accepted as the best objective measure of the pixel-to-pixel difference. RMSE is defined for instance in the classic book by Gonzalez and Woods [Gonzalez and Woods 2008]: for two images R and S of M columns and N rows,

$$e_{rms} = \sqrt{\frac{1}{M \cdot N} \sum_{k=1}^{M} \sum_{l=1}^{N} [r(k,l) - s(k,l)]^2}$$

where $r(k,l)$ is the value of the pixel at coordinates (k,l) in the first image, and $s(k,l)$ that of the corresponding pixel in the second image.

In our case, we have not computed the RMSE across the whole set of $M \cdot N$ texels, since this would artificially "dilute" the error by introducing many zeroes in the sum, corresponding to non-used texels of the adapted texture atlases. Instead, we have taken into account, for what we have called "ITEM-RMSE", only the net texels that have actually been transferred from both original images, so the ranges in the sums above are $k \in \{1, ..., p\}$ and $l \in \{1, ..., q\}$ respectively, $p \times q$ being the size of the lattice Γ described above.

5 Results

Analyzing the results we have obtained is not an easy task: since there is no other way to check the error between two textures for a same 3D mesh, we cannot guarantee that our results are correct. Nevertheless, we can set several hypotheses which any system needs to verify, and then test if ours does. To begin with, it is obvious that if we compare two identical texture images, the error must be zero. But what if we change the mask of the texture (and the texture coordinates accordingly), and preserve the color information, and compare the resulting texture with the original one?

We find an example of this situation in Figure 6. Image *a* is the original texture provided for the "Buddha" 3D model, whose mesh is shown in wireframe at its left, and *b* is another texture atlas manually created with the same net color information but resized and spatially relocated. Images *a* and *b* could never be compared with usual image subtraction techniques, but with our method we can check that the ITEM-measured error between them is, in fact, null (except for the intrinsic imprecision of floating point operations). On the other hand, we have image *c*, which is a reduced quality texture for the same 3D mesh. Again, this image could never be compared directly with neither *a* nor *b*, but thanks to ITEM we can measure the difference between them, which is 15.42 if we use bilinear interpolation and 14.84 with nearest neighbor replication.

Although being able to compare two texture atlases with totally different masks is one of ITEM's most important and innovative features, the results of such a comparison may not be as clarifying as the ones presented below. The following results illustrate the comparison of texture atlases that have been compressed for their lighter transmission, specially for online 3D applications. In this scenario, it could seem that the new atlas generation is not as useful as before but, thanks to it, triangles are presented proportionally to their real magnitude in the 3D mesh. This way, the distortion added by the compression scheme is evaluated considering its real impact in the textured 3D model.

In this new use case for ITEM, namely error measurement in coding frameworks for textured 3D meshes, JPEG would typically be used to compress the textures. By comparing pixel by pixel the original and reconstructed (i.e., compressed and de-compressed) images, we would not find out the real influence in the visual perception of the textured 3D mesh. Indeed, there are situations where even if the RMSE between the two images is not very high, big artifacts and considerable distortion are noticeable once the mesh is rendered. We therefore believe that ITEM-RMSE is more appropriate than classic RMSE to measure errors in the context of textured 3D meshes.

Besides, let us analyze and illustrate further the impact that texture coordinates (and the induced interpolation that takes place during the rendering process) may have in such a compression framework. Since there may be big areas of texture atlases that are not effectively used for texturing, the ITEM-RMSE differs, sometimes considerably, from the classic RMSE, as we prove with three examples:

1. For the "Buddha" 3D model shown in Figure 6, the plots in Figure 9-top give an idea of how misleading the classic RMSE may be in suggesting that the difference between the original and reconstructed texture maps is lower than it is. Although all the texels in the mask of image *a* are actually used (like all the ones of *b*, and unlike all the ones of *c*), since the atlas has a large number of black/useless texels, the vast majority of which will be equally black/useless after compressing and decompressing that image with JPEG, the classic RMSE value will be artificially low. As expected, the ITEM-RMSE value is larger, since only net texels are considered. This turned to be the case for all the "quality" JPEG compression levels[1] that we tested, and for the two options that we considered to emulate the rendering process: bilinear interpolation and nearest neighbor replication.

2. The opposite situation is presented in Figure 9-bottom, where the same curves are presented for the "Pagoda" 3D model (see Figure 7). In this case, the original texture atlas shown in image *a* is also inefficient, but not due to the existence of many black and obviously useless texels: image *b* highlights the texels actually used once texture coordinates are taken into

[1] In fact, quantization levels for the discrete cosine transform coefficients.

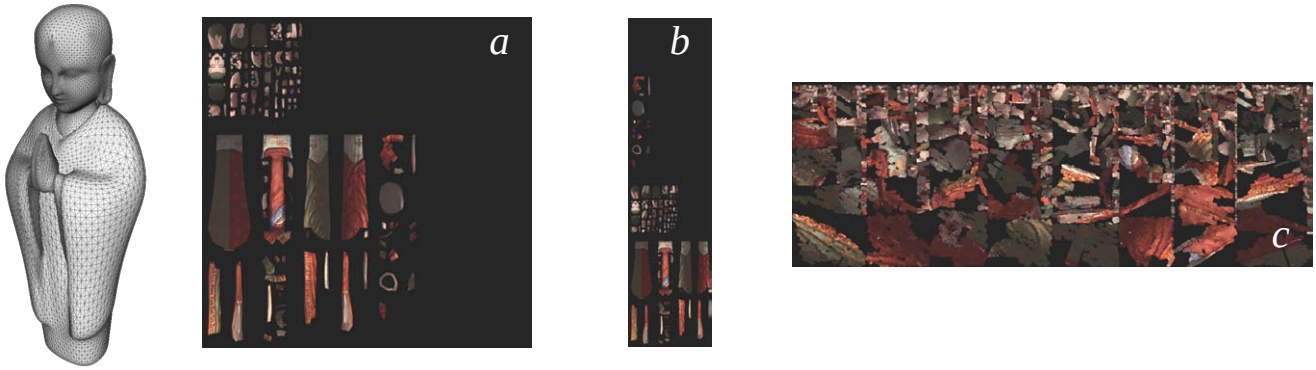

Figure 6: *"Buddha" 3D model: mesh, original texture atlas (a), and another two potential atlases (b, c).*

Figure 7: *"Pagoda" 3D model: mesh and original texture atlas (a), of which for some reason only the highlighted portions (b) are really used.*

Figure 8: *"Soldier" 3D model: mesh, original texture atlas (a) (notice how many more pixels are devoted to the face than to the clothes, proportionally to their true 3D sizes), and a section of the textured 3D mesh where, due to the bilinear interpolation (b), the noise due to JPEG compression (c) is hardly noticeable.*

account – artists have their own logic... When JPEG is used to compress this texture atlas, quantization introduces distortion everywhere, including areas not really used for texturing, so the classic RMSE is in fact higher than our ITEM-RMSE, and also artificially so.

3. Another interesting case is illustrated thanks to the 3D model of a soldier in a camouflage uniform, depicted in Figure 8, whose texture atlas has many borders, i.e., high frequency components. One could again think that the ITEM-RMSE would be higher than the classic RMSE, but Figure 9-bottom proves it is clearly lower *when bilinear interpolation is used*. This has a logical explanation which again shows how ITEM is able to calculate a more reliable error. JPEG-based compression tends to add noise and distortion along any strong edges causing the image to have high frequency components: see Figure 8-*c*. When the images are compared using the classic RMSE, noisy pixels increase the error. However, when we apply ITEM-RMSE and use a bilinear interpolation (to simulate the behavior of a good 3D renderer), we are in fact including in the process a low pass filtering stage which reduces the noise in the texture (see image *b*) and, as a result, the difference between both images decreases. On the other hand, since nearest neighbor replication does not include any kind of filtering, the aliasing effect is even worse, and therefore the error increases considerably.

6 Conclusions and future work

As the industry requires more and more 3D content, there is a need of objective tools for measuring the quality of synthetic 3D models. There exist methods which measure the shape error between two naked 3D meshes, of which one may be a simplified version of the other, with less vertices, edges and facets. There exist as well traditional methods to analyze the error between different images based on visual perception features or just direct, objective analysis of the pixel-by-pixel color difference. But none of these methods addresses the problem of comparing two different texture maps meant to be wrapped around the same naked 3D mesh. Indeed, the error, be it objective or subjective, measured between two texture maps considered as plain, rectangular images, may not represent correctly at all the resulting error in the textured 3D mesh once it is rendered. In fact, since the two texture maps need not even have the same number of texels or aspect ratio, a direct comparison using classic image processing techniques is not at all straightforward in the general case.

The ITEM (Inter-Texture Error Measurement) method and ITEM-RMSE metric we have presented in this paper are specifically designed for comparing two textures to be mapped onto the same 3D mesh, regardless of their aspect ratios or dimensions. This is a first step towards our final goal of comparing the final textured 3D models, a context in which it is meaningless to use a traditional image error measuring system, or even a perceptual image comparison algorithm, to examine the differences between two texture atlases whose masks, to begin with, might be completely diverse. ITEM takes into account the size of the 3D mesh triangles in each atlas to calculate the visual influence of every texel, and ITEM-RMSE gives a much less biased value than the classic RMSE, since only net texels are taken into account. Moreover, we have presented and analyzed different examples showing that ITEM-RMSE is more meaningful when texture resolution is not homogeneous, or when JPEG-like compression introduces noise by mistreating high frequencies. As we have seen in these examples, the ITEM-RMSE value is not always higher or lower than the classic RMSE, but more adjusted to reality and, therefore, more reliable. ITEM can be used for many applications, such as testing the space efficiency of dif-

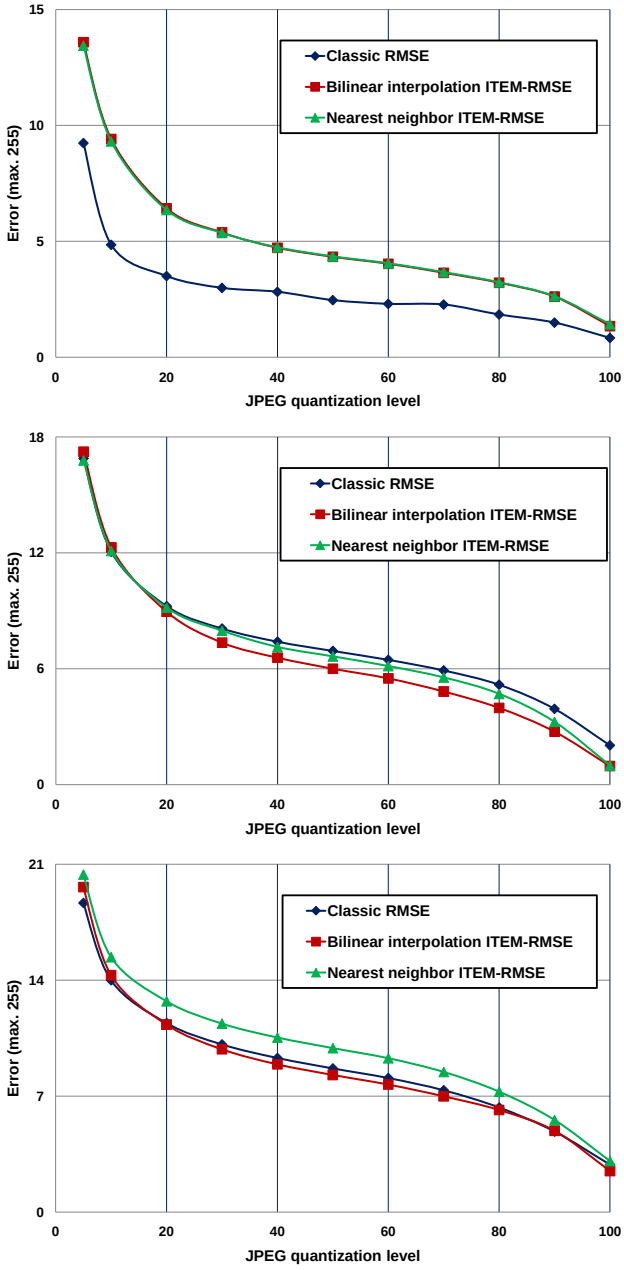

Figure 9: *Comparison of RMSE (blue) vs. ITEM-RMSE with bilinear interpolation (red) and ITEM-RMSE with nearest neighbor replication (green), for three 3D models: from top to bottom, "Buddha", "Pagoda" and "Soldier" (see Figures 6, 7 and 8, resp.).*

ferent 3D modeling software plug-ins for compiling several images into a single texture atlas, or for measuring the objective quality of a multi-resolution texturing scheme, etc.

We foresee several avenues for future work. From a practical viewpoint, implementing a more efficient and intuitive program or plug-in that researchers or artists could use to compare texture maps is very appealing. From a more scientific viewpoint, we consider trying to measure the combined shape and texture error to compare two textured 3D meshes in a more complete way.

7 Acknowledgements

This work has been partially supported by the *Ministerio de Ciencia e Innovación* of the Spanish Government under project TEC2010-20412 (Enhanced 3DTV).

We thank Samsung for the "Buddha" and "Pagoda" textured 3D models, donated among others to MPEG[2] for their use within the core experiments on multi-resolution 3D mesh compression. We also thank Khaled Mamou for having shared with us initial ideas, and our colleagues Daniel Berjón and Raúl Mohedano for their valuable suggestions to improve this paper.

References

ASPERT, N., SANTA-CRUZ, D., AND EBRAHIMI, T. 2002. MESH: Measuring Errors between Surfaces using the Hausdorff distance. In *Proc. ICME'02 (Intl. Conf. on Multimedia and Expo)*, IEEE, 705–708.

AVILÉS, M., AND MORÁN, F. 2008. Static 3D triangle mesh compression overview. In *Proc. ICIP'08 (Intl. Conf. on Image Processing)*, IEEE, 2684–2687.

BEGHDADI, A., AND PESQUET-POPESCU, B. 2003. A new image distortion measure based on wavelet decomposition. In *Proc. ISSPA'03 (Intl. Symp. on Signal Processing and its Applications), vol. 1*, IEEE, 485–488.

CERNEA, D. C., MUNTEANU, A., ALECU, A., CORNELIS, J., SCHELKENS, P., AND MORÁN, F. 2010. Efficient error control in 3D mesh coding. In *Proc. Intl. Workshop on MMSP'10 (MultiMedia Signal Processing)*, IEEE, 292–297.

FLOATER, M. S. 1997. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14*, 3, 231–250.

GONZALEZ, R. C., AND WOODS, R. E. 2008. *Digital image processing*. Pearson/Prentice Hall.

IORDACHE, R., AND BEGHDADI, A. 2001. A Wigner-Ville distribution-based image dissimilarity measure. In *Proc. ISSPA'01 (Intl. Symp. on Signal Processing and its Applications), vol. 2*, IEEE, 430–433.

JPEG. 1994. *ISO/IEC Intl. Standard 10918-1:1994, Information Technology - Digital compression and coding of continuous-tone still images: Requirements and guidelines*. ISO/IEC.

KHODAKOVSKY, A., LITKE, N., AND SCHRÖDER, P. 2003. Globally smooth parameterizations with low distortion. *ACM Trans. on Graphics 22*, 3 (Proc. SIGGRAPH'03 Conf.), 350–357.

[2]MPEG (http://mpeg.chiariglione.org/) stands for "Moving Picture Experts Group" and is formally ISO/IEC JTC1/SC29/WG11, whose sister committee ISO/IEC JTC1/SC29/WG1 is commonly known as JPEG (Joint Photographic Experts Group).

LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. MAPS: Multiresolution Adaptive Parameterization of Surfaces. In *Proc. SIGGRAPH'98 Conf.*, ACM, 95–104.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. on Graphics 21*, 3 (Proc. SIGGRAPH'02 Conf.), 362–371.

MURATA, H., FUJIYOSHI, K., NAKATAKE, S., AND KAJITANI, Y. 2002. Rectangle-packing-based module placement. In *Proc. ICCAD'95 (Intl. Conf. on Computer-Aided Design)*, IEEE, 472–479.

PAGÉS, R., ARNALDO, S., MORÁN, F., AND BERJÓN, D. 2010. Composition of texture atlases for 3D mesh multi-texturing. In *Proc. EG-IT'10 (EuroGraphics Italian Chapter) Conf.*, EG, 123–128.

SANDER, P. V., SNYDER, J., GORTLER, S. J., AND HOPPE, H. 2001. Texture mapping progressive meshes. In *Proc. SIGGRAPH'01*, ACM, 409–416.

TEO, P. C., AND HEEGER, D. J. 1994. Perceptual image distortion. In *Proc. ICIP'94 (Intl. Conf. on Image Processing), vol. 2*, IEEE, 982–986.

WANG, Z., AND BOVIK, A. C. 2002. A universal image quality index. *IEEE Signal Processing Letters 9*, 3, 81–84.