IMPROVED TEXTURES FOR 3D VIRTUAL RECONSTRUCTION AND VISUALIZATION BY A MODIFIED MULTISCALE TEXTURE SYNTHESIS APPROACH

A. Hast [a, *], M. Ericsson [b], T. Reiner [c]

[a] Creative Media Lab, University of Gävle. Gävle, Sweden - aht@hig.se
[b] Uppsala University, Uppsala, Sweden - martin.ericsson@it.uu.se
[c] University of Stuttgart, Stuttgart, Germany - reinertm@vismail.informatik.uni-stuttgart.de

KEY WORDS: Virtual reconstruction, Multiscale texture synthesis, Aliasing, HSV-colour space

ABSTRACT:

When photos of walls are used in urban 3D visualizations they are often of limited quality due to the fact that it can be very hard, or even impossible, to take close up photos of the whole part of walls, especially for buildings with several floors. Thus walls will appear either pixelized or blurry when the viewer comes close to them. The latter if some kind of interpolation technique is being used to reduce the pixelization. In any case it his has a big impact on how the viewer perceives the 3D environment as it will look far from real. We present how a modified multiscale texture synthesis approach can be used to create highly detailed textures from photos with different levels of detail and scale. The novel idea is to switch colour space in order to improve both quality and speed. By using the HSV space it is possible to maintain colours, especially when the examplar image does not contain all colours present in the target image.

1. INTRODUCTION

1.1 Motivation

In the process of 3D virtual reconstruction and visualization of buildings it is necessary to acquire textures of walls, etc and often photographs are used in order to obtain the highest quality possible. Aliasing problems will occur when these textures are used for the 3D models and different interpolation techniques can be used to minimize the aliasing effect when one moves close to the walls. One drawback with antialiasing (Foley, 1997) is that it will make the texture look blurry, however this is preferred over having the pixels appear like big homogeneous square blocks, which makes the texture look pixelized.

One way to handle this problem is to take close up photos of all parts of the wall in order to obtain highly detailed textures that will look good on close distances. Nonetheless this is often not feasible or even possible for buildings with more than one floor, unless there is a ladder, mobile crane or similar available.

This paper propose a novel approach where one photo (the target texture) is taken of a large part or even the whole wall and will therefore have relatively low details. Next a so called exemplar image is taken on close range, which will contain high amount of details, however only for that particular part of the building. The main idea is to use the exemplar texture to insert details in the target texture using a modified multiscale texture synthesis technique that will be further explained in the paper.

The main contributions of this paper are, first to use multiscale texture synthesis for improving 3D virtual reconstructions of walls, second the novel idea to use the HSV colour space in order to maintain colours as they appear in the target texture. The latter will also speed up the time consuming process of texture synthesis considerably. Moreover an implementation using High Performance Computing (HPC) resources like the Graphics Processing Unit (GPU) will be discussed.

1.2 Related Work

Texture synthesis is the process of taking one smaller texture and then make it larger in size, not by tiling, but by synthesizing it. (Efros, 99; Wei, 2000) Several approaches exist and the hierarchical texture synthesis method (Heeger, 1995) builds a tree of the smaller texture with different sizes very much like in the mipmapping method. The smallest texture (on the lowest level) is used in the first step and texels are randomly taken from it and randomly inserted into the new synthesized texture of corresponding size. Then follows a process where a mask with a specific shape is scanning the synthesized texture in a scanline order fashion while copying texels from the original texture which has the best matching neighbourhood (Wei, 2002). In the same time as the texture is synthesized on this level, another texture is synthesized on a higher level by copying a 2x2 neighbourhood into that texture, which accordingly will be twice as large in both directions, i.e.4 times larger in total.

In multiscale texture synthesis (Lee, 2008) there already exists a texture version available of the otherwise initially randomized and then synthesized texture, namely the target texture. Then a number of examplar textures are taken so that they will contain similar details like the target texture but on a higher level, and they can be used to build a more detailed version of the target texture. An examplar graph is built for this purpose and the target texture is placed in the root and textures with higher details are placed on the next levels depending on their resolution. One texture on one level can thus depend on several exemplar textures on the previous higher level. Since the

* Corresponding author

colours can differ on different levels it has been proposed to use a colour transfer function (Han, 2008)

2. METHODS

2.1 Considerations

It is proposed that a similar technique like multiscale texture synthesis is used with some important differences. First of all it is desirable to make the procedure work for a small number of exemplar images, often only one. The main reason for this is that we want to keep the initial process as well as the synthesis step as simple and fast as possible with a minimum amount of user guidance as possible. Therefore the time consuming process of building an examplar graph is avoided. As shown later in this paper, the synthesis step requires HPC resources and every simplification in the process, especially in the inner loops, will have a great impact of the total computation time needed.

While using these simplifications it can be supposed that the target image will contain colour information that is not found in the examplar images and it will be shown how this problem can be handled efficiently by switching to the HSV colour space.

Second it would be optimal if the resulting highly detailed texture would have the same properties as the target image has, i.e. when the upscaled texture is down sampled, it is desirable that the resulting texture will be equal to the original target texture. The proposed method comes very close to this requirement.

Furthermore the examplar texture is taken in such a way that when the multiresolution (Wei, 2002) tree is built, the lowest level will have approximately the same size in captured details, i.e., the same scale, as the target texture. Hence for a brick wall this means that the bricks will cover approximately the same area (number of texels) in both the target and the examplar texture on the lowest level.

Now it is possible to proceed in a similar manner as for multiresolution texture synthesis, however we do not need to change the synthesized texture on the same level as the algorithm proceeds, since we already have a structure in details on this level. Actually, it is undesirable to change the synthesized texture since the target texture should be unchanged, while details are added to the synthesized textures on higher levels. Hence, only the synthesized textures will be changed by adding new details. Moreover it is not necessary to have a mask that is adapted to scanline order traversal. It turns out that a simple 3x3 mask will yield a satisfactory result quality wise. It is preferable to have as small mask as possible when it comes to speed. Generally smaller masks will yield sharp undesirable noisy details while larger masks will smear out the details.

2.2 Changing to HSV Colour Space

The proposed method produces textures that look very realistic and often it is very hard to tell if it is a real photo or not. However, there are some problems when the examplar texture does not contain similar information as in the target texture, i.e. when the matching is bad. One example that often is apparent on brick walls is when a single brick has for an example a greenish tone in the otherwise red wall. It is proposed to handle this problem by converting the colours into the Hue, Saturation and Value (HSV) colour space (Sonka, 2008). The HSV colour model separates the colour into three channels, similar to the more common red, green and blue colour model (RGB) but instead it uses a measurement of hue, saturation and value also called brightness.

One of the benefits of this colour model is that it corresponds better to human perception where the brightness is decoupled from hue and saturation, in comparison to the RGB model where the brightness is encoded in all three colour channels. Actually, one advantage of using this colour model in the approach taken here is that the matching can be done using the V element only, instead of R, G and B. This will make the algorithm faster, which of course is a big advantage since texture synthesis is a heavily time consuming process, including four nested loops.

The synthesized textures on higher levels are constructed using the synthesized V elements and the H and S elements are taken from the target texture, which ensures that the original colours of the bricks are maintained. However it is important that the H and S elements are bi-linearly interpolated in the upscaling process, otherwise the colour will be visible as blocks. It should be said that there exist a number of interpolation techniques that could be used for this purpose (Gonzales, 1993)

2.3 Proof of Concept

A picture was taken of a wall and then it was down sampled so that the width and height was halved in each step, thus making it 4 times smaller (our target texture). This process was repeated four times and the final image was 256 times smaller in total. The result is shown in figure 1. Note that this image is heavily down sampled and some bricks are almost touching each other, hence it will really put the different synthesis approaches on the test.

Figure 1. A brick wall in low resolution: 204x153 pixels

The upper left corner of the original image (96x96 pixels) was chosen as the examplar image on the same level of details. In figure 2 we see the examplar on the previous level containing 4 times more details (192x192 pixels). In a real application we would probably choose to take images using the highest resolution possible with for an example a 10 Megapixel camera. However, for the initial tests it was necessary to limit the textures to be rather small for reasons that will be discussed in the next section.

Figure 2. An examplar containing 4 times more details, 192x192 pixels

Initially the ordinary multiscale approach was examined. Since only one examplar is used it is likely that it will not contain all colours present in the target texture. It turned out that a brick in the upper right corner had a greenish tone that was not properly represented. In figure 3 the resulting texture in the right is compared to the target image on the same level of details to the left. It is obvious that the colours are not represented correctly.

Figure 3. The target image to the left and a synthesised version to the right using ordinary multiscale texture synthesis yielding colours, which are far from correct.

Using the proposed approach, by converting to HSV space and synthesizing the V part while interpolating H and S, will give a much more accurate result. In figure 4 it is shown how the problematic brick is synthesized while still keeping the greenish tone.

Figure 4. The proposed approach produces colours as expected.

It should be noted that the result in figure 4 cannot be exactly equal as the left image in figure 3 since a texture synthesis approach is used. However it is required that the colours in the target texture are kept as intact as possible in the upscaling process while details are added from the examplar image.

Another experiment was performed by inserting the text ”3D-ARCH”, changing just the H element to green in the brick wall as shown in figure 5.

Figure 5. “3D-ARCH” was written on the wall.

Then ordinary multiscale texture synthesis as well as the proposed method was used to produce the images in figure 6. It is obvious that the proposed method performs much better in reproducing the colours. Since previous methods work in RGB space, the matching will be poor as there are no green colours in the examplar image. Hence the texels will be taken from areas that are as similar as possible, i.e. giving the best match. In this case some grey colours are inserted and the text is barely visible.

Figure 6. Ordinary texture synthesis fails in representing the colour as shown in the upper image while the proposed approach in the lower image produces colours as expected.

2.4 Implementation

The images were produced by implementing the algorithm in MATLAB since it has excellent tools for image processing. However it is quite impractical to run texture synthesis on a single CPU because it is heavily time consuming due to its $O(n^4)$ complexity. For the proof of concept tests a target image of 204x153 texels and an examplar image of 192x192 texels were used. That adds up to more than 1150 million iterations. For each scale up the synthesized image becomes 4 times larger and an examplar image that is also 4 times larger will be used, thus the total number of iterations will increase with a factor of 16. That is why it is important to decrease the work done in the inner loop and the proposed approach that work on the V element instead of RGB decreases the work by a factor of 3.

Today when 3D architectures are captured using a digital camera a resolution of at least 8-10 Megapixels would be used and it can be expected that this size will continue to increase in the future, just as it has done in the past. This calls for efficient texture synthesis methods and fast hardware becomes more important.

Let us assume that the examplar is taken so that the details can be upscaled 4 times, then the target image will be 256 times

smaller than the final synthesized image. This will end up in almost 250 billion operations and each new upscale will yield 16 times more operations. Obviously some kind of HPC implementation is needed, either on a cluster of CPU's, on a multicore CPU or on a GPU. So far we have implemented the algorithm on a GPU and the performance increase is considerable. Of course the MATLAB implementation is not the fastest choice but very useful and the proof of concept test took a couple of minutes for the first upscale step. On the GPU the same computation took just a couple of seconds.

Method \ Step	1	2	3
A	1.2s	7.3s	106.3s
B	1.2s	2.8s	9.1s

Table 1. Timing of two different examplar approaches in three upscaling steps on the GPU.

Table 1 shows the timing in seconds of three upscaling steps. Method **A** used a 4 times larger examplar in each step, while method **B** used the same size for every step. Theoretically method **A** would require 16 times more time to finish while method **B** would require 4 times more time. Keeping in mind that we must subtract about 0.6-0.8s for the initializing, reconstruction, colour space conversion, etc, then we obtain numbers that are according to the theoretical values.

The implementation of our algorithm was made to run on the GPU of a modern graphics card. The OpenGL Shading Language provides data structures and functions that are very suitable for our texture-based task. A fundamental concept of general-purpose computing on GPU's is to set up a 1:1 mapping between the input pixels and framebuffer elements in order to render a full-sized quad that stretches out over the entire viewport. The fragment shader then processes this input stream and the results are eventually stored in the framebuffer. Nonetheless we are facing a difficulty here: our intention is to upscale a texture, so we lose a 1:1 mapping between input and output pixels. If both the texture's width and height are being doubled, one incoming texel generates four texels of the final output texture. In order to preserve a 1:1 mapping, we split our procedure into two separate tasks.

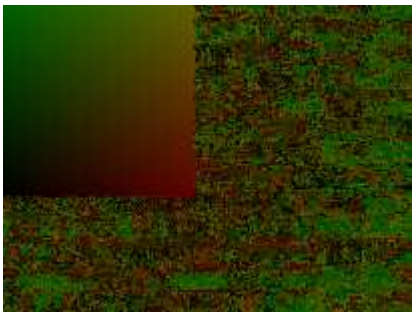

Figure 7. An atlas map where the *x* and *y* values are decoded in red and green colours.

First, a viewport-sized quad is rendered, textured with the input texture. The fragment shader tries to find a best match of the neighbourhood of every incoming texel from the examplar texture. As a result, the colour values will not be stored, instead the position of the best match in the framebuffer is stored and the resulting image can be used as an index map or an atlas as shown in figure 7, where the red channel encodes the *x* position and the green channel encodes the *y* position of the best match.

One can clearly see the smooth region in the upper-left part of the index map, which is the result of the fact that the reference texture was taken out of this part from the original image and therefore fits exactly back into its former position. With the blue and alpha channels included, positions up to (65536, 65536) can be encoded in an index map which is easily sufficient even for very large examplar textures. Using framebuffer objects, the index map can be rendered off-screen and directly into a texture.

Finally, we can access the high-resolution texture at the positions stored in the index map to synthesize and reconstruct a detailed upscaled version of the input texture. This task can be finished on the CPU using system memory as it is quickly done.

3. RESULTS

3.1 Comparison of Quality

When the target texture is zoomed in so that each texel is 64 times larger, the texture looks pixelized as shown in figure 8.

Figure 8. Zooming without filtering makes the texture look pixelized.

Note that since the original texture was heavily down sampled, it is difficult to see the mortar between the bricks in the upper left corner. This will yield artefacts in the synthesis step. Such shortcomings can be avoided by taking an image with better resolution.

Usually some filtering techniques are used in order to avoid the pixelization apparent in figure 8. This will make the texture look blurry and the details will appear as if they are out of focus as shown in figure 9. In this image a simple linear interpolation was used. Different interpolation kernels will give different results, however the main impression still remains: the texture appears to be out of focus.

Figure 9. Filtering makes the texture look blurry and ”out of focus”, but is the usual method to avoid pixelization.

Finally the proposed texture synthesis approach gives a much more appealing result as shown in figure 10.

Figure 10. The proposed approach gives a much more appealing and realistic result.

The level of details is increased by the multiscale texture synthesis and the colours are not drastically changed by using the proposed approach.

When 3D reconstruction of buildings is used by the proposed approach it is possible to use an image of the whole wall or large parts of a wall using a high resolution camera. The inserted details can be taken from an examplar image taken from the same wall. This image will be taken on a close range and will therefore cover a small part of the wall.

In order to test the algorithm further the same examplar was used for new photos of other walls. In figure 11 a part of an arch is shown that has been upscaled twice. One can see that the non horizontal bricks are a bit jagged and that probably depends on the fact that all bricks in the examplar are horizontal. Anyhow, the algorithm does not completely fail for these cases and it should also be noted that the bricks on the target picture was a bit larger than on the previous target picture and the algorithm still does a good job.

Figure 11. A part of an arch has been upscaled in two steps.

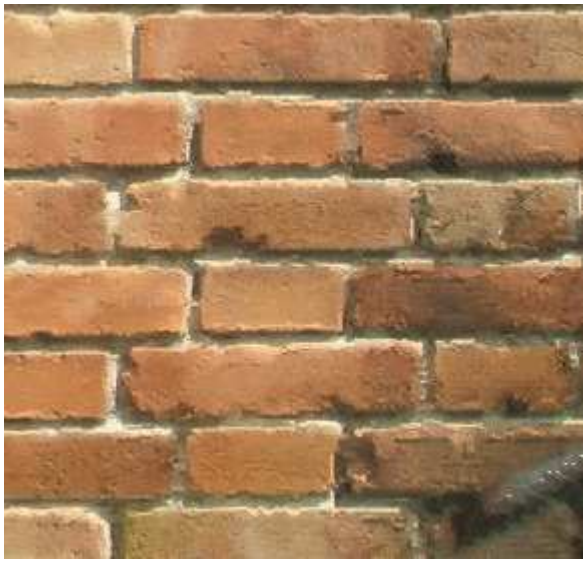

Figure 12. Artefacts are clearly visible.

In figure 12 yet another wall has been synthesized, still with the same examplars. Some artefacts are clearly visible. Not

surprisingly is the rusty iron bar in the bottom-right corner quite jagged. However, worse is that the bricks have repeated patterns that do not look good. In this case the original image is taken from a greater distance than for the first target image and thus makes the bricks smaller. This might be the reason for these artefacts and this underlines the need for having the target and examplar images in approximately the same scale of details. It can also be shown that these artefacts depend on the fact that the details are too small since they will disappear almost completely if the resolution is 4 times larger for the target image as shown in figure 13.

Figure 13. The artefacts disappear when a 4 times larger target image is used.

Both target images cover exactly the same part of the wall, however for figure 13 it has 4 times higher resolution (number of pixels). Both the bricks and the iron bar are clearly much better synthesized and the tendency for bricks to grow into each other is also diminished since there are simply more pixels covering the mortar between individual bricks.

4. CONCLUSIONS

4.1 Speed and Quality

It has been shown how a modified multiscale texture synthesis approach can be used to obtain highly detailed textures for 3D virtual reconstructions. Only one examplar image is necessary, which is the main difference between the proposed approach and previous algorithms. The problem of not finding the same colours is efficiently handled by switching to the HSV colour space. Since the matching operation is performed in the innermost loop, it will make the whole process faster when the matching is done using only one element (V) instead of three elements (RGB). The resulting texture with higher resolution will have the same main properties as the target texture. The synthesis process produces a visually plausible result with more details. In order to obtain good images it is important that the examplar is taken so that when it is down sampled it will cover the same area of details as the target image. Furthermore it is important that the target image has an adequate resolution to start with, in order to avoid that details start to grow into each other.

4.2 Future Work

There are still some things that can be both improved and further examined. When using the HSV space the first two colours where interpolated, while the V element was synthesized. There are many different kinds of interpolation techniques that could be examined. When a wall is built of different materials, like a stone base with bricks on top then it would be more efficient to use an examplar graph of at least two textures, one for each type of material. Similarly it would be preferable to mask out some details like windows and doors and treat them differently. It should also be examined if the mask should have weights or even increase in size for higher levels. Furthermore it has not yet been investigated what is the minimum size required for the examplar image in order to produce good results.

REFERENCES

Alexei A. Efros and Thomas K. Leung , 1999, *Texture Synthesis by Non-parametric Sampling*, In Proceedings of ICCV 99, pp 1033-1038.

Foley, J. D., van Dam, A., Feiner, S. K., Hughes, J. F., 1997, *Computer Graphics - Principles and Practice*, Addison-Wesley, pp. 617-646.

Gonzales, R. C., Woods, R. E., 1993, *Digital Image Processing*, Addison-Wesley, pp. 300-302.

Han, C., Risser, E., Ramamoorthi, R. and Grinspun, E., 2008, *Multiscale Texture Synthesis*. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008), 27(3) pp. 51.

Heeger, D. and Bergen, J., 1995, *Pyramid-Based Texture Analysis / Synthesis*. SIGGRAPH pp. 229-238.

Lee, S-H., Park, H-W., Lee, J. and Kim, C-H., 2008, *Multi-scale Texture Synthesis*. Journal of KCGS (The Korea Computer Graphics Society), 14(2).

Li-Yi Wei., 2002, *Texture synthesis by fixed neighborhood searching*. PhD Thesis, Stanford University, Palo Alto, CA, USA.

Li-Yi Wei, Marc Levoy. 2000, *Fast Texture Synthesis using Tree-structured Vector Quantization*, Proceedings of Siggraph, pp 479-488.

Sonka, M., Hlavac, V. and Boyle, R., 2008. *Image Processing, Analysis, and Machine Vision*. Thomson Learning, USA. pp.38.

ACKNOWLEDGMENTS

The authors wish to thank UPPMAX for using its HPC resources. All the images of walls were taken in the city of Perugia in Umbria, Italy.