MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Quilting for Texture Synthesis and Transfer

Alexei Efros and William T. Freeman

TR2001-17 December 2001

Abstract

We present a simple image-based method of generating novel visual appearance in which a new image is synthesized by stitching together small patches of existing images. We call this process ïmage quilting.Ḧirst, we use quilting as new, fast, yet very simple texture synthesis algorithm which produces surprisingly good results for a wide range of textures. Second, we extend the algorithm to perform texture transfer – rendering an object with a texture taken from a different object. More generally, we demonstrate how an entire image can be re-rendered in the style of a different image. The method works directly on the images and does not require 3D information.

Proceedings SIGGRAPH 2001, In Computer Graphics Proceedings, Annual Conference Series

MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY
http://www.merl.com

Quilting for Texture Synthesis and Transfer (DRAFT version)

Alexei Efros 545 Soda Hall Computer Science Division
EECS, UC Berkeley
Berkeley, CA 94720-1776
efros@cs.berkeley.edu

William T. Freeman
MERL, Mitsubishi Electric Research Labs.
201 Broadway
Cambridge, MA 02139
freeman@merl.com

TR-2001-17 April 2001

Abstract

We present a simple image-based method of generating novel visual appearance in which a new image is synthesized by stitching together small patches of existing images. We call this process *image quilting*. First, we use quilting as new, fast, yet very simple texture synthesis algorithm which produces surprisingly good results for a wide range of textures. Second, we extend the algorithm to perform texture transfer – rendering an object with a texture taken from a different object. More generally, we demonstrate how an entire image can be re-rendered in the style of a different image. The method works directly on the images and does not require 3D information.

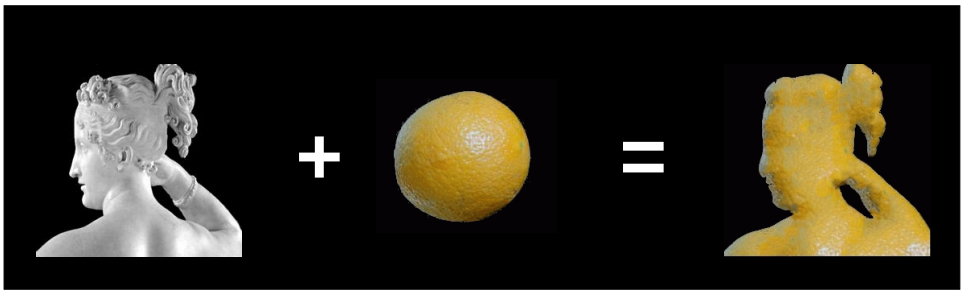
A revision of this draft will appear in: Proceedings SIGGRAPH 2001, In Computer Graphics, *Annual Conference Series.*

1. First printing, TR2001-17, April, 2001

* Alexei Efros was a student intern at MERL when this work was done.

Quilting for Texture Synthesis and Transfer

Alexei Efros and William T. Freeman. January, 2001. DRAFT--Please don't circulate.



Abstract

We present a simple image-based method of generating novel visual appearance in which a new image is synthesized by stitching together small patches of existing images. We call this process *image quilting*. First, we use quilting as new, fast, yet very simple texture synthesis algorithm which produces surprisingly good results for a wide range of textures. Second, we extend the algorithm to perform texture transfer – rendering an object with a texture taken from a different object. More generally, we demonstrate how an image can be re-rendered in the style of a different image. The method works directly on the images and does not require 3D information.

Key Words: Texture Synthesis, Texture Mapping, Image-based Rendering

1 Introduction

In the past decade computer graphics experienced a wave of research in the area of image-based rendering as people explored the idea of capturing some samples of the real world as images and using them to synthesize novel views rather than recreating the entire physical world from scratch. This, in turn, helped generate interest in image-based texture synthesis algorithms. Such an algorithm should be able to take a sample of texture and generate an unlimited amount of image data which, while not exactly like the original, will be perceived by humans to be *the same texture*. Furthermore, it would be useful to be able transfer the texture from one object to anther (e.g. the ability to cut and paste material properties on arbitrary objects).

In this paper we present an extremely simple algorithm to address the texture synthesis problem. The main idea is to synthesize new texture by taking patches of existing texture and stitching them together in a consistent way.

1.1 Previous Work

While the problem of texture analysis and synthesis from real images has had a long history of research in the computer vision and statistics communities, it was not until the seminal paper by Heeger and Bergen [5] that the quality of results reached a level acceptable for use in computer graphics. Their main insight was to model texture in terms of histograms of filter responses at multiple scales and orientations. It turned out that matching these histograms iteratively at different spatial scales was enough to produce impressive synthesis results for stochastic textures. However, since these histograms measure marginal, not joint, statistics they do not capture important relationships across scales and orientations, and thus the algorithm fails for more structured textures. Several attempts have been made to extend this idea to capture a wider range of textures, including De Bonet [1] who samples from conditional distribution over multiple scales, and later Portilla et.al. [8] who match both first and second order properties of wavelet coefficients. While important from a theoretical point of view, neither method is successful at capturing local detail of many structured textures.

A different approach is to directly model pixels given their spatial neighborhoods. Efros and Leung [3] propose a simple method of "growing" texture one pixel at a time. The conditional distribution of each pixel given all its neighbors synthesized so far is estimated by searching the sample image and finding all similar neighborhoods. This algorithm produces very good results for a wide range of textures, but is excruciatingly slow (a full search of the input image is required to synthesize every single pixel!). Wei and Levoy [11] show a way of significantly speeding it up (up to 2 orders of magnitude) by (1) using a multi-scale image pyramid and (2) clustering pixel neighborhoods (both inspired by earlier work of Popat and Picard [7]). However, using these optimizations means that often the best matching neighborhoods will not be found. Therefore, many textures, especially these with high frequency structure (such as images of text), are not synthesized as well as in [3].

Another recent approach, proposed by Xu et.al. [12], is inspired by the Clone Tool of PHOTOSHOP and GIMP and is so simple it's almost magical that it works. The idea is to take random square blocks from the input texture and place them randomly onto the synthesized texture (with alpha blending to avoid edge artifacts). While this technique will fail for highly structured patterns (e.g. a chessboard) due to boundary inconsistencies, for most semi-stochastic textures it works no worse than other more complicated algorithms (such as [1]). The technique was successfully used by Praun et.al. [9] for semi-automatic texturing of non-developable objects.

Methods have been developed in particular rendering domains which capture the spirit of our goals in texture transfer. Our goal is like that of work in non-photorealistic rendering (e.g. [2, 10, 6, 4]). A key distinction is that we seek to characterize the output rendering style by sampling from the real world. This allows for a richness of rendering styles, characterized by samples from photographs or drawings.

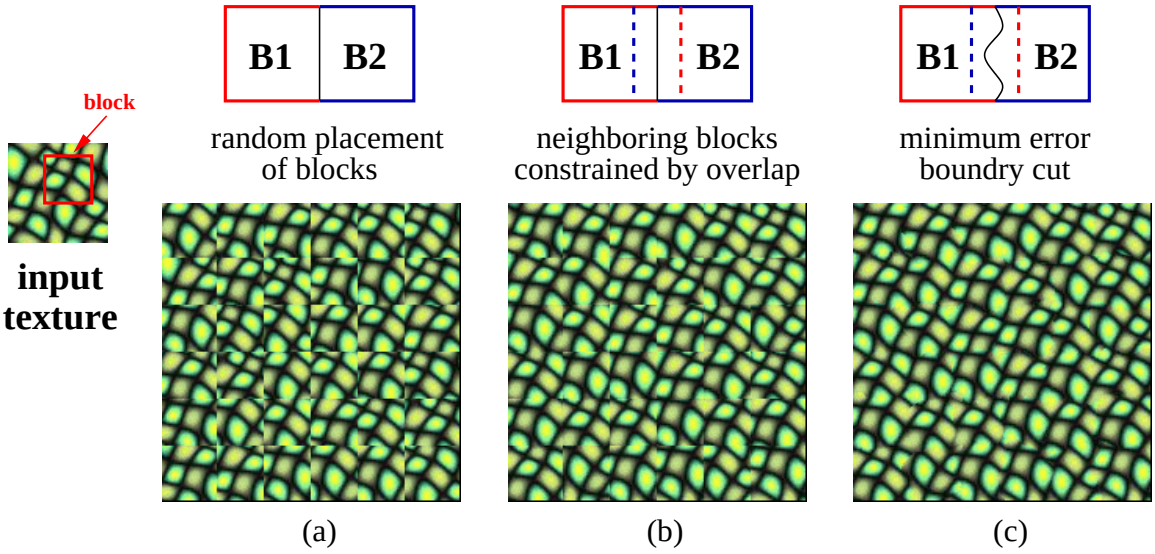


Figure 1 Quilting texture. Square blocks from the input texture are patched together to synthesize a new texture sample: (a) blocks are chosen randomly (similar to [12, 9]), (b) the blocks overlap and each new block is chosen so as to "agree" with its neighbors in the region of overlap, (c) to reduce blockiness the boundary between blocks is computed as a minimum cost path through the error surface at the overlap.

1.2 Motivation

One curious fact about one-pixel-at-a-time synthesis algorithms such as Efros and Leung [3] is that for most complex textures very few pixels actually have a choice of values that can be assigned to them. That is, during the synthesis process *most pixels have their values totally determined by what has been synthesized so far*. As a simple example, let us take a pattern of circles on a plane. Once the algorithm has started synthesizing a particular circle, all the remaining pixels of that circle (plus some surrounding ones) are completely determined! In this extreme case, the circle would be called the texture element (*texel*), but this same effect persists to a lesser extent even when the texture is more stochastic and there are no obvious texels. This means that a lot of searching work is waisted on pixels that already "know their fate". It seems then, that the unit of synthesis should be something more than a single pixel, a "patch" perhaps. Then the process of texture synthesis would be akin to putting together a jigsaw puzzle, quilting together the patches, making sure they all fit together. Of course, determining precisely what are the patches for a given texture and how they are put together hits at the heart of texture analysis – an open problem in computer vision. Here we will present a poor man's version of stitching together patches of texture to form the output image. We call this method "image quilting".

2 Quilting

In this section we will develop our patch-based texture synthesis procedure. Let us define the unit of synthesis B_i to be a square block of user-specified size from the set S_B of all such overlapping blocks in the input texture image. To synthesize a new texture image, as a first step let us simply tile it with blocks taken randomly from S_B. The result shown on Figure 1(a) already looks somewhat reasonable and for some textures will perform no worse than many previous complicated algorithms as demonstrated by [12, 9]. Still, the result is not satisfying, for no matter how much smoothing is done across the edges, for most structured textures it will be quite obvious that the blocks do not match.

As the next step, let us introduce some overlap in the placement of blocks onto the new image. Now, instead of picking a random block, we will search S_B for such a block that by some measure agrees with its neighbors along the region of overlap. Figure 1(b) shows a clear improvement in the structure of the resulting texture, however the edges between the blocks are still quite noticeable. Once again, smoothing across the edges will lessen this problem but we will attempt to solve it in a more principled way.

Finally, we will let the blocks have ragged edges which will allow them to better approximate the features in the texture. Now, before placing a chosen block into the texture we will look at the error in the overlap region between it and the other blocks. We find a minimum cost path through that error surface and declare that to be the boundary of the new block. Figure 1(c) shows the results of this simple modification.

2.1 The Algorithm

The complete algorithm is as follows:

- Go through the image to be synthesized in raster scan order in steps of one block (minus the overlap).
- For every location, search the input texture for a set of blocks that satisfy the overlap constraints (above and left) within some error tolerance. Randomly pick one such block.
- Compute the error surface between the newly chosen block and the old blocks at the overlap region. Find the minimum cost path along this surface (using dynamic programming) and make that the boundary of the new block. Paste the block onto the texture (with optional alpha blending). Repeat.

The size of the block is the only parameter controlled by the user and it depends on the properties of a given texture; the block must be big enough to capture the relevant structures in the texture, but small enough so that the interaction between these structures is left up to the algorithm.

Details: in all of our experiments the width of the overlap edge (on one side) was 1/6 of the size of the block. The error was computed using the $\mathcal{L}2$ norm on pixel values. The error tolerance was set to be within 0.1 times the error of the best matching block.

2.2 Synthesis Results

The results of the synthesis process for a wide range of input textures are shown on Figures 2 and 3. Despite its simplicity, the algorithm does rather well when compared to the current state-of-the-art methods (Figure 4). The algorithm is not only trivial to implement but is also quite fast: the unoptimized MATLAB code used to generate these results ran for between 15 seconds and several minutes per image depending on the sizes of the input and output and the block size used. Because the constraint region is always the same it's very easy to optimize the search process (say, with a k-d tree) without compromising the quality of the results.

3 Texture Transfer

Because in image quilting there is a direct mapping to image positions in the texture source image, it is particularly well suited for *texture transfer*. We augment the synthesis algorithm by requiring that each patch satisfy a desired *correspondence map*, as well as satisfy the texture synthesis requirements. The correspondence map is a spatial map of some corresponding quantity over both the texture source image and a controlling target image. That quantity could be image intensity, low frequency image components, image orientation, or even 3-d surface slope, if that information were available for the texture source image as well.

An example of texture transfer is shown in Figure 5. Here, the correspondence map are the image intensities of the man's face. That is, bright patches of face and bright patches of rice are defined to have a low correspondence error. The synthesized rice texture conforms to this second constraint, yielding a rendered image where the face image appears to be rendered in rice.

For texture transfer, the quilting algorithm must be modified to deal with the fact that the image being synthesized must now respect two independent constraints: (a) that local the output are legitimate, synthesized examples of the source texture, and (b) that the correspondence image mapping is respected. In order to do this, the procedure of picking the best block is modified in the following way. In addition to ranking all blocks by how well they satisfy the overlap constrains with the source texture synthesized so far, now we must also rank them by how well they match (according to the correspondence map) the destination texture patch at that location. The two ranking are merged with user-specified parameter α, which determines the tradeoff between those two constraints, and the best matching block is picked using the same procedure as before.

Because of the added constraint, sometimes one synthesis pass through the image is not enough to produce a visually pleasing result. In such cases, we iterate over the synthesized image several times, reducing the block size with each iteration. The only change from the non-iterative version is that in satisfying the local texture constraint the blocks are matched not just with their neighbor blocks on the overlap regions, but also with whatever was synthesized at this block in the previous iteration. This iterative scheme works surprisingly well: it starts out using large blocks to roughly assign where everything will go and then uses smaller blocks to make sure the different textures fit well together. In our tests, we used 3 to 5 iterations, reducing the block size by a third each time.

Our texture transfer method can be applied to render a photograph using the line drawing texture of a particular source drawing; or to transfer material surface texture onto a new image. Figures 6 and 7 show examples of texture transfer.

Figure 5 From the rice texture (upper left), the normal synthesis process produces a texture like on the upper right. However, if the synthesis is constrained by another image, the result looks quite different (lower right).

4 Conclusion

In this paper we have introduced *image quilting*, a simple method of synthesizing a new image by stitching together small patches of existing images. Despite its simplicity, this method works amazingly well when applied to texture synthesis, producing results that are equal or better than the state-of-the-art [3] but at a fraction of the computational cost. We have also extended our method to texture transfer in a general setting (something that, to our knowledge, has not been done before) with some very promising results.

Besides many practical applications, we think this work also provides new insights into the way texture is modeled and perceived. The fact that crude patches of texture stitched together with a few constraints work just as well as meticulously synthesizing each pixel by itself is a very non-trivial observation. It clearly shows the enormous importance of local structure for successful modeling of visual phenomena.

References

[1] J. S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *SIGGRAPH '97*, pages 361–368, 1997.

[2] C. J. Curtis, S. E. Anderson, J. E. Seims, Kurt W. Fleisher, and D. H. Salsin. Computer-generated watercolor. In *ACM SIGGRAPH*, pages 421–430, 1997. In *Computer Graphics* Proceedings, Annual Conference Series.

[3] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, pages 1033–1038, Corfu, Greece, September 1999.

[4] M. A. Kowalski et. al. Art-based rendering of fur, grass, and trees. In *ACM SIGGRAPH*, pages 433–438, 1999. In *Computer Graphics* Proceedings, Annual Conference Series.

[5] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95*, pages 229–238, 1995.

[6] V. Ostromoukhov and R. D. Hersch. Multi-color and artistic dithering. In *ACM SIGGRAPH*, pages 425–432, 1999. In *Computer Graphics* Proceedings, Annual Conference Series.

Figure 2 Synthesis results for a wide range of textures. The resulting texture is synthesized at twice the size of the original.

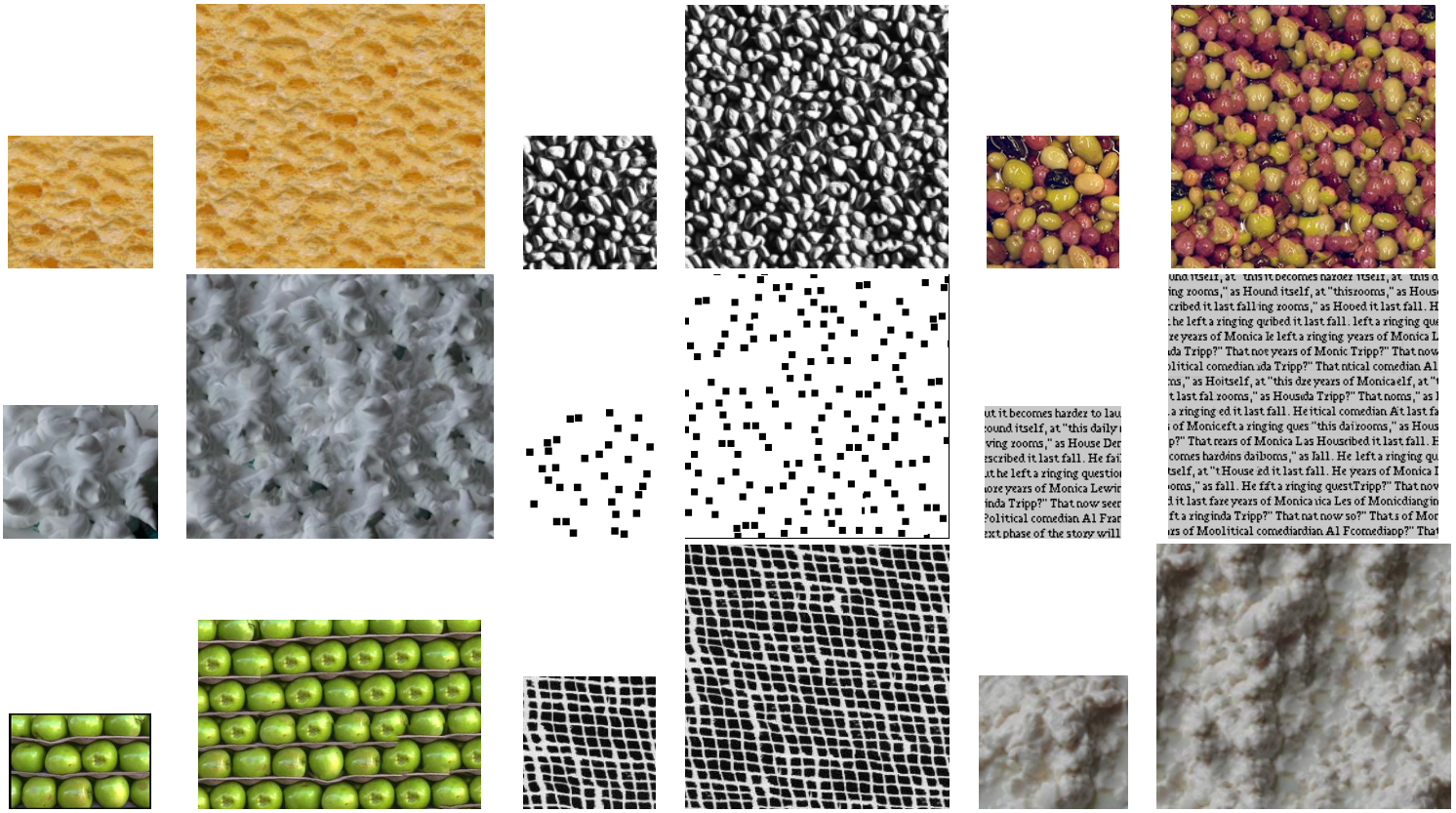


Figure 3 More synthesis results

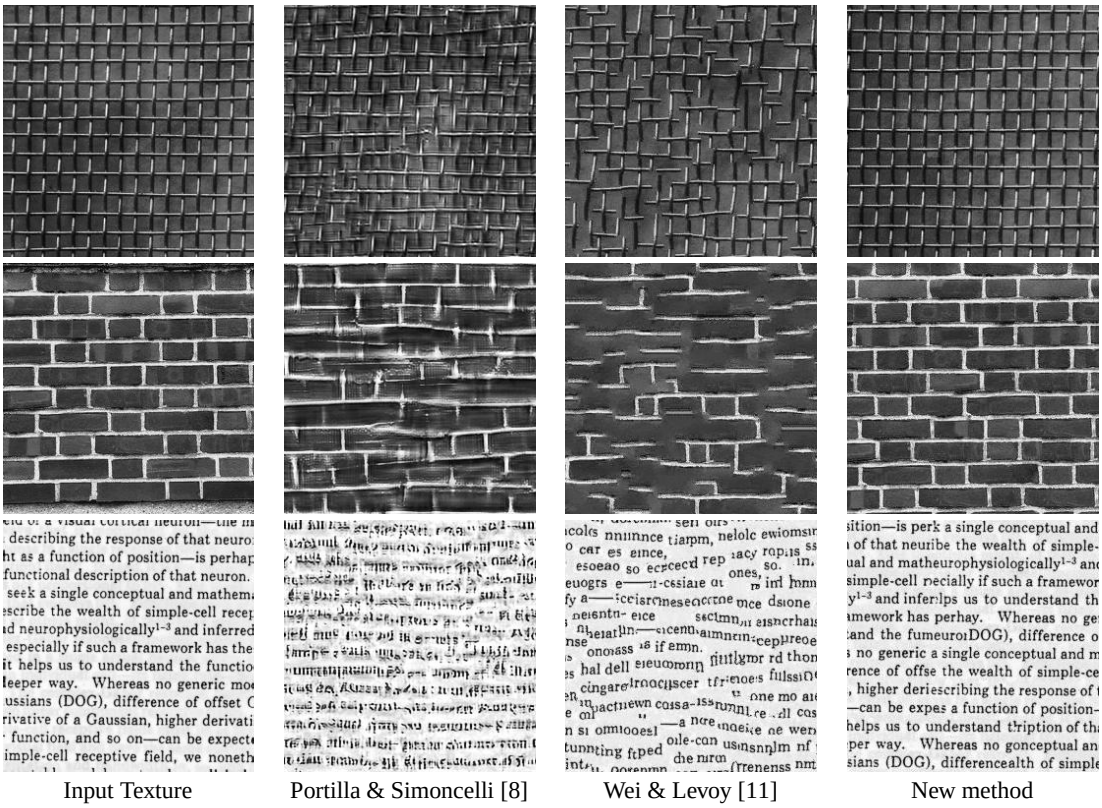


Figure 4 Our method compares favorably to the current state-of-the-art synthesis algorithms. Our results are virtually the same as Efros & Leung [3] (not shown) but at a much smaller computational cost.

Figure 6 Here a Picasso drawing is taken as a source texture (upper left) and applied to re-render a photograph of Richard Feynman (lower left) in a line-drawing style. A low-pass filtered version of the drawing is taken as the source correspondence map. The Feynman photo is used as its own correspondence map. The result (lower right) definitely shows the influence of Picasso – here Dr. Feynman is no longer smiling.

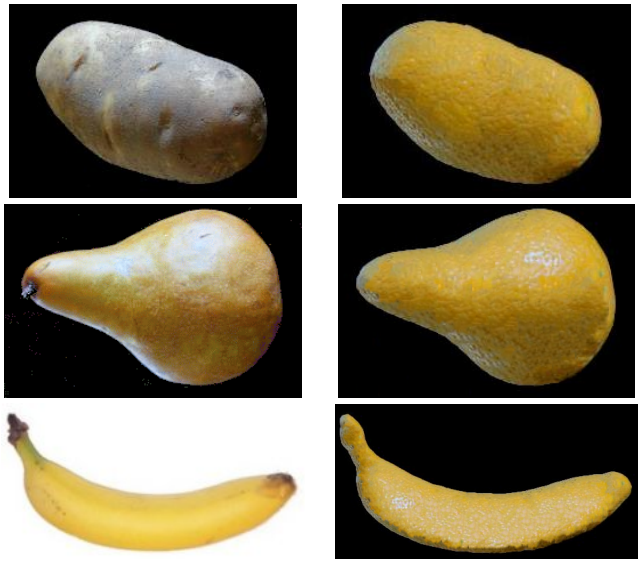


Figure 7 An orange has a very nice distinctive texture, quite suitable for texture transfer. Here, the photograph of an orange from the first page was used to give these fruits and vegetables a new look.

[7] Kris Popat and Rosalind W. Picard. Novel cluster-based probability model for texture synthesis, classification, and compression. In *Proc. SPIE Visual Comm. and Image Processing*, 1993.

[8] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–71, December 2000.

[9] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *SIGGRAPH 2000*, pages 465–470, 2000.

[10] M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin. Orientable textures for image-based pen-and-ink illustration. In *ACM SIGGRAPH*, pages 401–406, 1997. In *Computer Graphics* Proceedings, Annual Conference Series.

[11] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH 2000*, pages 479–488, 2000.

[12] Y. Xu, B. Guo, and H.-Y. Shum. Chaos mosaic: Fast and memory efficient texture synthesis. Technical Report MSR-TR-2000-32, Microsoft Research, April 2000.