Presented at IMA Conference on Vision, Video and Graphics, 10-11 July 2003, Bath, UK
Proceedings published by the Eurographics Association

Enhanced Texture Editing using Self-Similarity

S. Brooks, M. P. Cardle and N. A. Dodgson
University of Cambridge Computer Laboratory
15 J. J. Thomson Avenue, Cambridge, UK CB3 0FD
{sb329 | mpc33 | nad10 }@cl.cam.ac.uk

Abstract
Texture mapping is an indispensable tool for achieving realism in computer graphics. Significant progress has been made in recent years with regards to the synthesis and editing of 2D texture images. However, the exploration of user control for semi-automatic texture editing remains an open area of research. We present methods that partially address the semantic and technical limitations of Self-Similarity Based Editing. This is achieved by providing the user with more control over the similarity metric during editing and over spatial re-arrangement during cloning.

1. Introduction

Image editing software is often characterized by a seemingly endless array of toolbars, filters, transformations and layers. This is the most common approach to modeling the complexity of real world scenes. It provides the graphics designer with sophisticated tools that permit a high degree of control over geometric surfaces and their corresponding textures.

In general, this approach has been used extensively and has met with considerable success; however, the complexity of these editing tools requires that the user possess a correspondingly high level of expertise. In order to use these systems effectively, the user must typically be a capable artist as well as having substantial technical preparation. And so, it is understandable that these requirements are frequently beyond the casual user.

Recently, a counter trend has emerged in the field of image editing which aims to automate the process of constructing graphical objects of sufficient realism. Far from offering a massive array of image manipulation controls, these semi-automated prototype systems offer interaction at a higher semantic level, consequently minimizing the amount of user interaction. This alternate style of interaction has been made possible, in part, through advances in directed texture synthesis and computer vision.

It is in this context that we present enhancements to the self-similarity texture editing framework first presented in Brooks and Dodgson's short paper[7]. Self-similarity based editing allows complex operations to be performed on images with minimal user interaction. This is achieved by utilizing the inherent self-similarity of image textures to replicate intended manipulations globally.

The user is able to minimally specify alterations to a digital image, whilst relying on the system to perform repetitive, time-consuming tasks. As can be seen in Figure 1, self-similarity based editing is a visual analogue to 'text string search and replace' in that a single editing operation at a given location causes global changes: the same operation is performed on all similar areas of the image. Consequently, the style of interaction lies between automation and complete user manipulation.

The main contribution of this paper is the introduction of a method to control the spatial arrangement of textures for self-similarity based cloning. We also present the concept of Boolean similarity expressions and we show how improved self-similarity painting results can be produced using a mixture of neighborhood and wavelet similarity metrics.

2. Previous Work

Self-similarity based editing[7] shares many algorithmic features with multi-resolution approaches to texture synthesis, making it a good first point of discussion. Texture synthesis takes as input a texture of a fixed sized and produces an output texture of arbitrary dimensions which has the appearance of being made from the same source. Much of the current successful work in texture synthesis can be traced back to the pioneering work of Heeger and Bergen[15]. In their system they create a new instance of a texture through hierarchical histogram matching. De Bonet[8] later introduced a higher quality variant of this general approach, though perhaps not as compelling as the results of the simple neighborhood matching of Wei and Levoy[28].

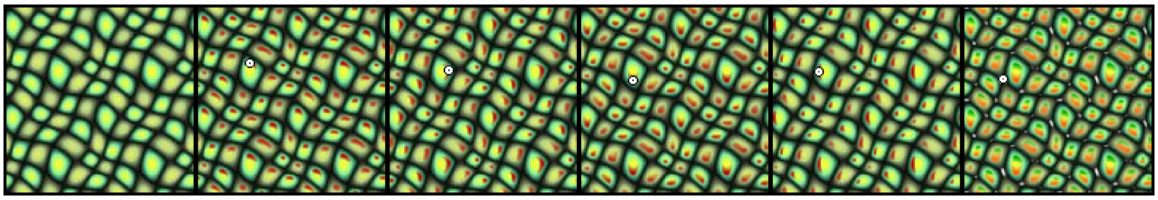
Figure 1: *Similarity Painting: (leftmost) original texture, (center four) single point painting, and (rightmost) multiple paintings applied.*

More recently, an entirely new class of image editing tool has emerged which employs texture synthesis to perform sophisticated image editing operations that require only a relatively small degree of user direction. One tool uses texture synthesis to remove entire objects from scenes[17]. Here the user selects a cropping area and a synthesis source area. The cropped patch is then replaced with a synthesized texture derived from the synthesis area. After the operation is complete no traces remain of the content that previously occupied the cropping area.

The distinction between texture synthesis and texture editing has become even more blurred with the development of user-directed texture transfer and synthesis methods[1, 12, 16, 20]. These user-directed synthesis methods permit the user to rearrange an entire image to conform to a paint-by-numbers sketch of the final image. Directed synthesis thereby becomes a rearrangement form of texture editing. And as we are introducing a modified texture cloning tool there are other forms of texture mixing[2] and image compositing[6] that deserve mention.

Other semi-automated texture creation systems include Live Paint[23], which uses the concept of a multi-resolution painting system[5] to combine procedural textures[11]. Dischler et al. [9] describe a unique hybrid approach that combines texture analysis and geometric modeling. Lewis' [19] early paper presents an interactive procedure for generating textures in the frequency domain. Alternatively, the genetic algorithm methodology presents candidate textures to users who implicitly provide fitness functions based on subjective aesthetic judgments[26].

Another system which manipulates vector based images is the search and replace method of Kurlander and Bier [18]. Conceptually, this system is most similar to self-similarity based editing. However, their system differs algorithmically as it operates strictly on vector images that are composed of distinct geometrically defined objects unlike raster based self-similarity editing tools.

Yet another fruitful source of user assistance in image editing has come from advances in the computer vision community. Examples of which are intelligent image selection[21] and snapping[13] tools. But, the most characteristic example of an image editing method which employs both texture synthesis and computer vision techniques is Barrett and Cheney's object-based image editing system[3]. In their interactive system, image objects are user-selected by manually collecting sub-object regions detected by a watershed algorithm. Once selected, image objects can be scaled, stretched, bent, warped or even deleted with automatic hole filling.

Perhaps the most extreme form of automation that still permits some degree of user input is the image stylization system of DeCarlo et. al. wherein the user's task is simply to view the image in a natural way[10]. The focus of the user's eyes is tracked during the viewing. The duration that the user lingers over each portion of the image is used to assign priority to details for a non-photorealistic rendering of the same image.

The present paper extends the previous work of Brooks and Dodgson[7] in which a self-similarity based editing system was first introduced. In their short paper, a simple, yet novel, method of interactive texture editing was presented that utilizes self-similarity to replicate intended operations globally over an image. As the self-similarity editing framework is central to our discussion we will begin with an overview of their original system.

3. The Self-Similarity Texture Editing Approach

Like a number of current texture synthesis methods, the self-similarity editing framework also uses multi-scale neighborhoods to assess the similarity of pixels within a texture. However, neighborhood matching is not employed to generate new instances of a texture. Similar neighborhoods are instead located for the purpose of replicating editing operations on the original texture itself, thereby creating a fundamentally new texture. This general approach is applied to texture painting, cloning and warping. These global operations are performed interactively, most often directed with just a single mouse movement.

3.1. System Overview

In the self-similarity editing system, changes made to a particular pixel by the user are made to affect all pixels that exhibit similar local neighborhoods. This allows for the following concise texture editing operations:

1. *Replicated Painting:* altering the color or brightness of similar pixels.
2. *Replicated Cloning:* cloning of another texture onto the texture being altered.

3. *Replicated Warping:* locally contracting or expanding certain regions of the textures, based on similarity to the current selected pixel.

Painting and cloning are similar operations which overlay colors onto the image being edited. In Figure 1 we have a simple case of painting a solid red color onto each pixel whose neighborhood is sufficiently similar to the pixel selected by the user. The reader will note the directional control of the tool. By this we refer to the ability to affect a particular side of all of the texture elements at once. Figure 2 shows an example of the replicated painting of a solid green color over specific areas of a bark texture. This is followed by the replicated cloning of a moss texture over the same areas.

Replicated warping is distinct from the other tools in that it does not affect pixel color; it instead modifies the shape of image regions under the user's guidance. Those pixels whose local neighborhoods are within a certain threshold of similarity to the user selected point are expanded locally. Since the overall area remains the same, some regions are compressed while others are expanded. Figure 3 shows the application of replicated warping to an image of chrysanthemums. The original image (shown on the left) has been altered so the flower heads are enlarged (shown on the right).

3.2. The Original Algorithm

In order to determine which pixels in the image are sufficiently similar to the pixel selected by the user, the local circular neighbourhood of the chosen selection point is compared against that of every other pixel's neighbourhood in the same image. The editing operation is applied to the selected pixel but also to a subset of all pixels in the image: those that have local neighbourhoods whose difference from the selected pixel are within a certain threshold.

For small textures, where efficiency is not a serious constraint, the neighborhood is simply defined as those pixels bounded by an immediate circle of pixel diameter *d*. The distance metric is then the L_2 norm, i.e. the sum of square differences between each corresponding neighborhood pixel. For example, in Figure 1 the diameter $d = 9$, yielding a circular neighborhood of 69 pixels. This then requires summing 69 squared R, G and B differences, producing a number in the range of 0 to $256^2*3*69 \approx$ 13.5M.

The distance threshold is set by the user and defines the maximum distance value beyond which the opacity of the applied paint is zero. Between zero distance and the distance threshold the opacity is scaled linearly. The user is also provided with a global opacity multiplier, which reduces or increases the opacity for all affected pixels.

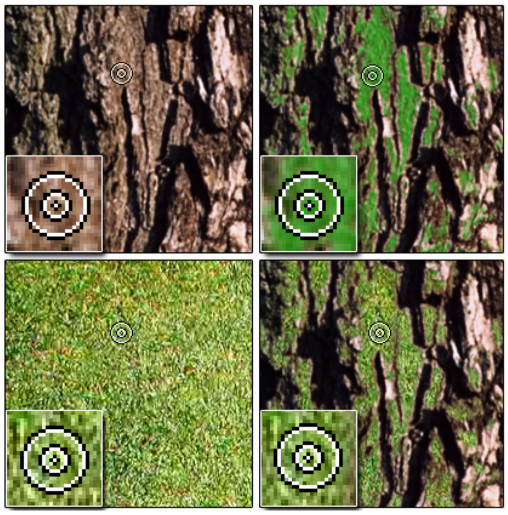
Figure 2: *Clock-wise from top left: original image, replicated painting, replicated cloning, cloning image.*

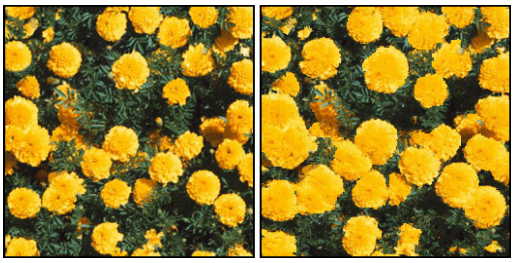
Figure 3: *Flowers expanded using self-similarity based warping.*

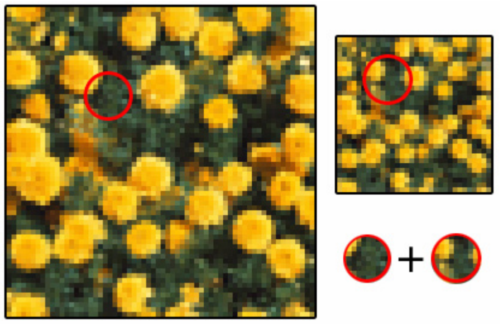
Figure 4: *Multiscale neighborhood.*

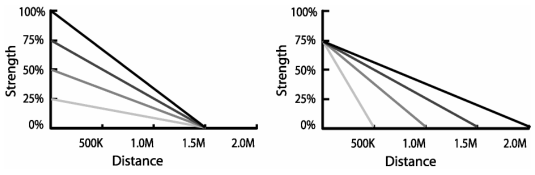

Figure 5: *Altering global strength or distance threshold.*

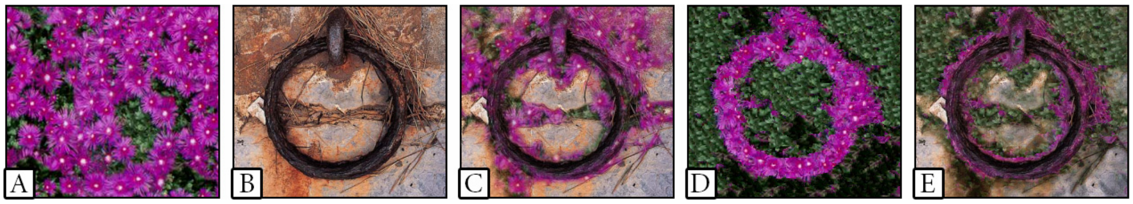

Figure 6: *Semantically meaningful texture cloning: A) Texture source for cloning. B) Image target for cloning. C) Cloning with the original source texture. D) Re-arranged version of A using self-similarity masks. E) Final cloning.*

The left-hand graph in Figure 5 depicts the altering of the global opacity multiplier (Strength) while maintaining a constant threshold (Distance) of 1.5 million. Conversely, the right-hand graph holds the Strength at 75% with the Distance ranging from 0.5 to 2.0 million. This indicates that the final opacity levels are a function of both the global opacity multiplier and the current threshold. Together the two controls can be used to control the number of pixels that are affected as well as applied opacity.

For improved efficiency, the neighbourhood of the selected pixel is augmented to include the corresponding neighbourhoods in higher levels of a constructed Gaussian image pyramid. In Figure 4, two levels of an image pyramid are shown. Each level has a circular neighbourhood of the same diameter, *d*, for a given pixel. But although it is the same diameter, the reader will note that the same sized circular area in the higher level in the pyramid includes a larger portion of the original image. By using the higher levels in an image pyramid, it permits the inclusion of a wider neighbourhood area at a lower computational cost, while the lower level neighborhoods retain priority for nearer pixels. The system performs painting and cloning operations on 512×512 textures at 7.5 fps running on an off-the-shelf Athlon 2200+ PC.

Moving from replicated painting to replicated cloning is plainly a matter of positioning the cloning texture and using the corresponding colour values from the cloned texture instead of a solid colour value over the whole image. But, as we will see later this is often not sufficient for convincing results.

Similarity-based texture warping uses neighborhood similarity as a measure of local area deformation (Figure 3). The scalar similarity values derived from neighborhood distances are converted into 2D area expansions. But, depending on the texture and on the amount of expansion the warped texture can suffer a loss of high frequency detail. Brooks and Dodgson overcome this by re-synthesizing detail into expanded areas, using the newly warped texture as a constraining image for super-resolution synthesis as described in Hertzmann et al.[16]. A final result of which can be seen in Figure 3 (right).

4. Improved Texture Editing

In the original paper, Self-Similarity Based Texture Editing[7], a novel system of concise texture editing was presented which allows the user to make global changes to texture images with minimal user intervention by exploiting the inherent self-similarity of textures. However, as first proposed, the original method has limitations which fall into two categories:

Semantic: For cloning, the tool requires that the cloning image spatially 'matches' the image being cloned into. For example, when Figure 6A is cloned into 6B, the arrangement of the flowers is arbitrary with respect to the ring (Figure 6C).

Technical: For both cloning and painting, the tool does not work as well for textures that contain a high degree of randomness or sharp features. This is due to the smoothing tendency of Gaussian-pyramid neighborhood metrics.

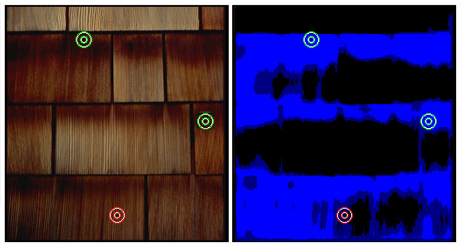

Figure 7: *Using a 3 point Boolean similarity expression to construct a synthesis mask.*

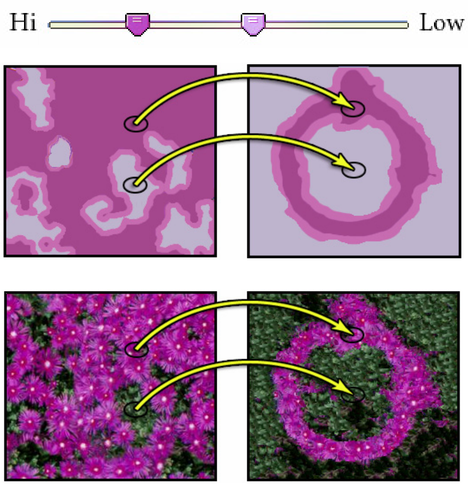

Figure 8: *Texture re-arrangement prior to cloning.*

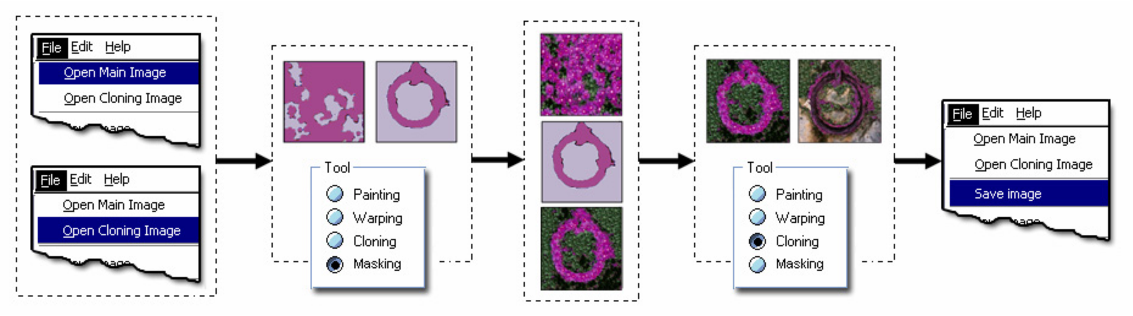

Figure 9: *System workflow. 1) Open target image and cloning texture. 2) Perform self-similarity masking. 3) Rearrange cloning texture using automatic Texture-by-Numbers synthesis. 4) Clone rearranged texture onto target image. 5) Save final image.*

4.1. Enhanced Cloning Control

Firstly, we address the semantic limitation by allowing the user to re-arrange the cloning texture (Figure 6A) so that it better matches the target image which it is being cloned onto (Figure 6B). This results in a more appropriate cloning texture (Figure 6D). We do this by semi-automatically constructing Texture-by-Numbers[15, 17] masks of both the cloning texture (Figure 6A) and the image being cloned onto (Figure 6B). These masks are then used for a Texture-by-Numbers[15, 17] guided re-synthesis prior to cloning. For example, we specify that the flowers in Figure 6A, which are labeled with dark purple in the top-left mask of Figure 8, are to be synthesized into the dark purple ring, shown in the top-right mask of Figure 8.

We could force the user to painstakingly construct the Texture-by-Numbers masks by hand. However, this would not be in keeping with the concise nature of Self-Similarity Editing. To automate time consuming Texture-by-Numbers mask constructions, we have developed an addition to the Self-Similarity toolset that separates an input texture into distinct regions. This tool can be seen as sophisticated "Magic Wand".

Like the original self-similarity editing tools, the "Wand" compares the multi-scale neighborhoods of all pixels to the neighborhood of the pixel that the user has selected. Points that have 'Lower' similarity to the select point are given one color and those with 'Higher' similarity are assigned another. Although we provide default values for what is considered "Lower" versus "Higher" similarity, it is the user who is able to alter these designations by moving positions on a multi-point slider.

At the top of Figure 8 can be seen a two-point mask slider which is used to separate the image pixels into to those whose computed distance values are within the "High" level of similarity (shown in darkest purple) and those other pixels that are within the "Low" level of similarity (shown in medium purple), pixels whose value lies beyond the "Low" level are given the default value of light purple. In fact, the tool can separate the image into an arbitrary number of color sets with the addition of more similarity thresholds along the distance slider. In addition, for better texture synthesis, small regions are discarded. Once the similarity masks are constructed, they are then used for a Texture-by-Numbers[15, 17] guided re-synthesis prior to cloning.

The multi-stage cloning workflow is shown in Figure 9. The user begins by opening the target image and the cloning texture. The user then creates the masks for both the target and the cloning images using the self-similarity masking "Wand". These masks are then used to rearrange the cloning texture (Figure 6A) using automatic Texture-by-Numbers synthesis. The rearranged texture (Figure 6D) is then cloned onto the target image (Figure 6B), resulting in a more meaningful cloning operation (Figure 6E). Further example results of this enhanced cloning method are shown in Figures 13-16.

4.2. Boolean Similarity Expressions

This enhanced "Wand" also allows the user to select multiple similarity points (Figure 7) within the texture which together comprise a Boolean similarity expression. In this way, the user can specify that pixels must be like pixel A or pixel B but not pixel C.

It is also worth noting that Boolean similarity expressions also provide greater control for all self-similarity tools including masking, warping, painting and cloning (Figure 11). Using both positive and negative similarity points allows the 'similar' regions to be carved out with greater accuracy for each of these operations.

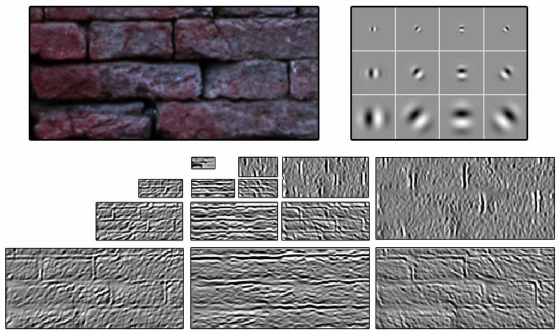

Figure 10: *The steerable wavelet filter responses of a brick texture.*

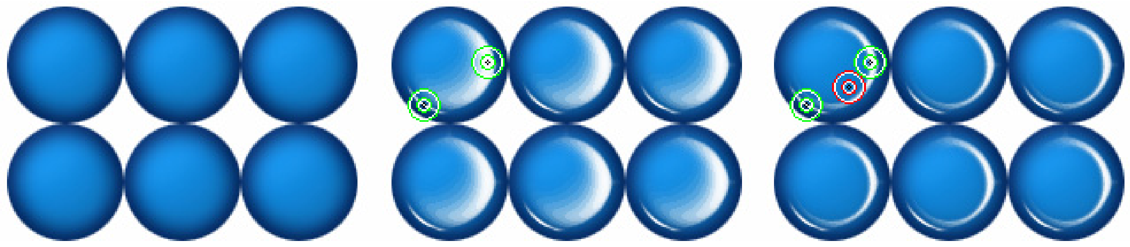

Figure 11: *Example use of a multi-point Boolean similarity expression. The green rings denote positively weighted similarity points, and the red is negative. Left: original repeating blue-dot texture. Middle: two positively weighted similarity points used to paint white onto similar pixels. Right: The expression is asking the system to "paint white those pixels that are similar to the green points but dissimilar to the red".*

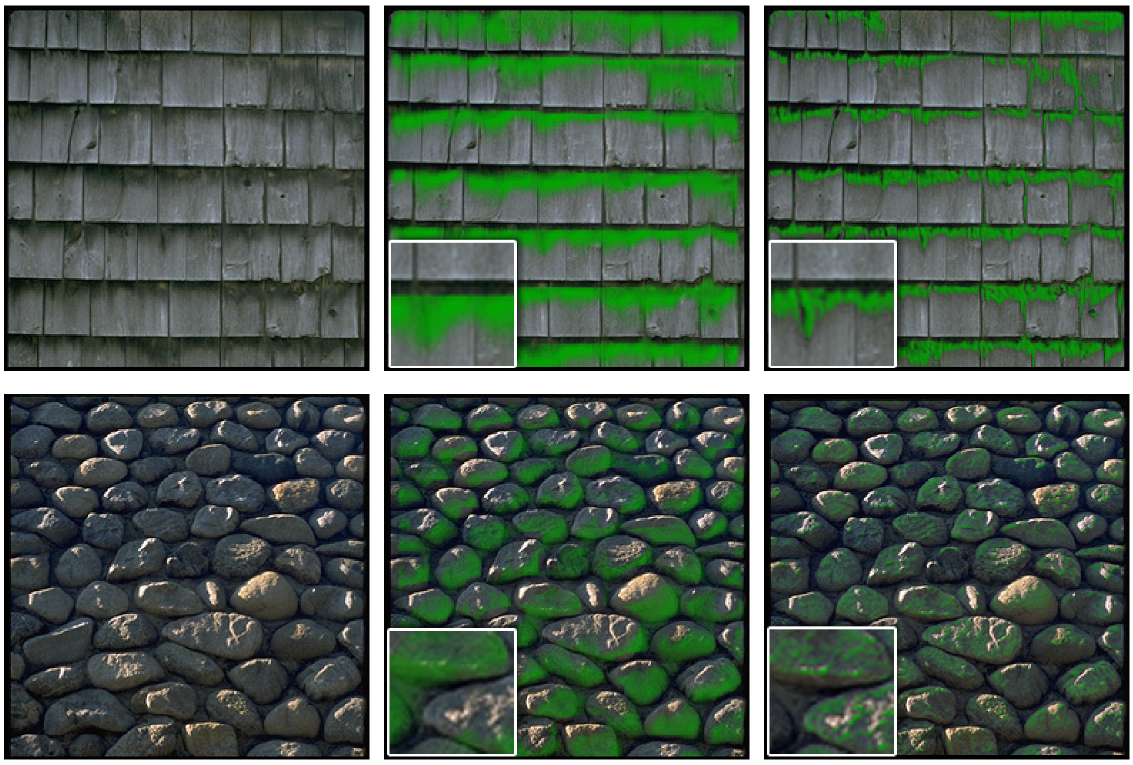

Figure 12: *Painting of solid green color onto textures. Left: original. Center: Painted using only Gaussian pyramid neighborhood responses. Right: Both neighborhood and wavelet responses used.*

4.3. Enhanced Similarity Metric

To address the limitations of the original similarity measure, we also explore the use of a wavelet based similarity metric. Moreover, we give the user even finer control by providing a slider that specifies what proportion of neighborhood versus wavelet responses are to be used in the similarity calculation.

To improve the similarity metric we include multi-scale responses from a steerable pyramid transform of the image being edited[25]. Like the Gaussian pyramid, this transform decomposes the image into several spatial frequency bands. It also further divides each frequency band into a set of orientation bands which respond to rotationally varying edges. Figure 10 shows an example of the wavelet transform being applied to a brick texture. The top left shows the original image with wavelet filters to the right (3 bands and 4 orientations). Along the bottom are the corresponding steerable pyramid sub-band images for the brick texture.

Since the wavelet transform responds strongly to edges at varying orientations, by placing more emphasis on wavelet responses the user can thereby cause the self-similarity tool to react more strongly to sharp features in the texture during editing and avoid the problem of excessive smoothing that can result from relying solely upon Gaussian pyramid neighborhoods.

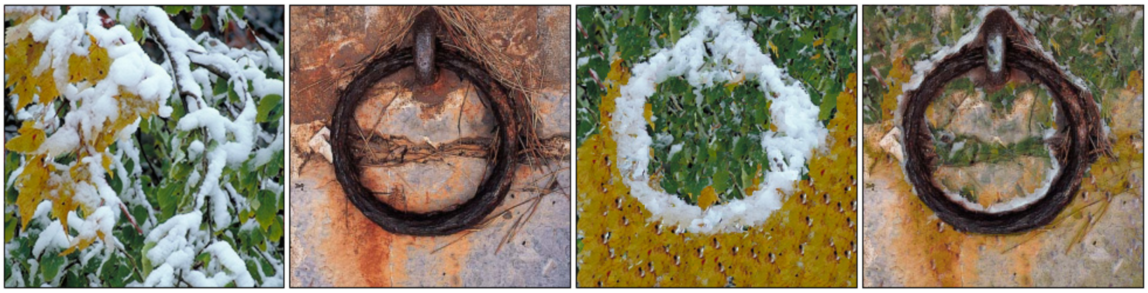

Figure 13: *Snowy leaf texture is re-ordered and cloned onto a rusting ring.*

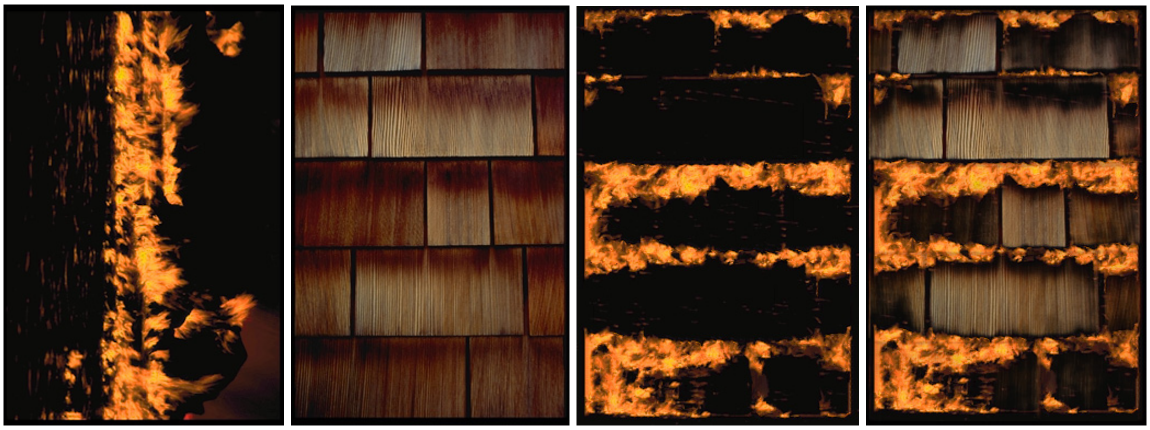

Figure 14: *Fire texture is re-ordered and cloned onto wood shingles.*

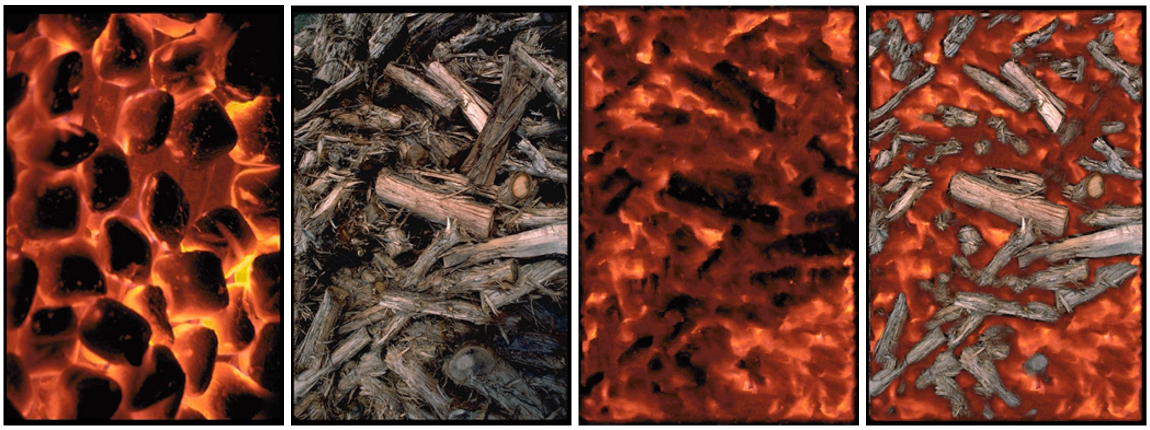

Figure 15: *Hot coals are re-ordered and cloned onto a wood pile.*

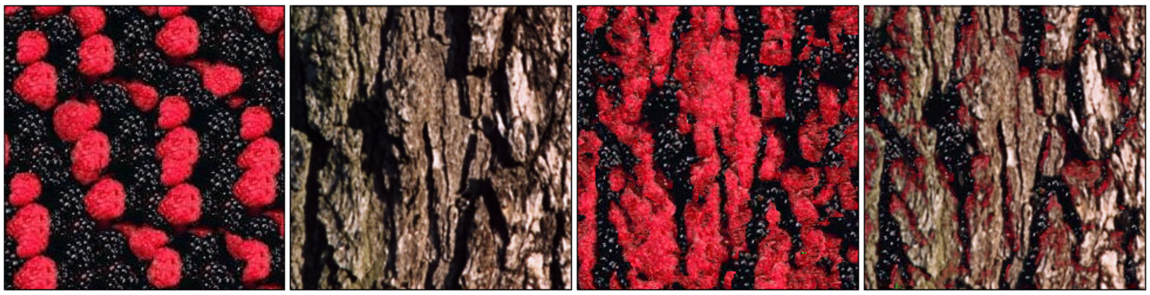

Figure 16: *Red and black berries are re-ordered and cloned onto bark.*

Figure 12 shows the painting of a solid green color onto two separate textures. The original images are shown to the left. The center images show results using only Gaussian pyramid neighborhood responses and the images to the right show results using both neighborhood and wavelet responses. As can be seen from the zoomed inset images, by incorporating the wavelet responses into the similarity distance metric, the self-similarity tool is able to respond to the finer edge details in the original images. And so, the user can dictate the extent to which these edge details influence the final outcome.

5. Conclusions

We have presented significant enhancements to the self-similarity editing framework which extend its range of use and improves the quality of results. This has been achieved by giving the user further control over the similarity metric and over the spatial arrangement of the cloning image. However, there exist further opportunities to extend the capabilities of this system.

Currently the approach works best for textures which are uniformly lit. Non-uniform lighting leads to poorer results. We believe that this restriction might be addressed by integrating similarity-based editing with a photo editing system such as that of Oh et al.[22] which permits both distortion free cloning and texture illumination correction.

Additional possibilities include the use of more sophisticated similarity measures. The technique might be extended to geometric and texture editing operations on a 3D object based on the similarity of local surface curvature instead of, or in concert with, texture similarity.

References

1. M. Ashikhmin. "Synthesizing natural textures". *ACM Symposium on Interactive 3D Graphics*, pp. 217–226, March 2001.
2. Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. "Texture Mixing and Texture Movie Synthesis Using Statistical Learning", *IEEE Transactions on Visualization and Computer Graphics*, 7, 2, 120-135, 2001.
3. W. Barrett and A. Cheney. "Object-Based Image Editing". *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002)*, 21(3), pp. 777-784, August 2002.
4. T. Beier, and S. Neely. "Feature-Based Image Metamorphosis". *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, 26(2), ACM, pp. 35-42, 1992.
5. D. Berman, J. Bartell, and D. Salesin. "Multiresolution Painting and Compositing". *ACM SIGGRAPH 94*, 85-90, 1994.
6. P. Burt, and E. Adelson. "A Multiresolution Spline with Application to Image Mosaics". *ACM Transactions on Graphics*, 2, 4, 217-236, 1983.
7. Stephen Brooks and Neil A. Dodgson. "Self-Similarity Based Texture Editing". *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002)*, 21(3), pp. 653-656, July 2002.
8. J. S. De Bonet. "Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images". *ACM SIGGRAPH 97*, 361-368, 1997.
9. J. Dischler, and D. Ghazanfarpour. "Interactive Image-Based Modeling of Macrostructured Textures". *IEEE Computer Graphics and Applications*, 19, 1, 66-74, 1999.
10. Doug DeCarlo and Anthony Santella. "Stylization and Abstraction of Photographs". *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002)*, 21(3), pp. 769-776, August 2002.
11. D. Ebert, F. Musgrave, D. Peachey, K. Perlin and S. Worley. *Texturing and Modeling: A Procedural Approach.* AP Professional, Cambridge, MA, 1994.
12. A. Efros and W. Freeman. "Image quilting for texture synthesis and transfer". *Computer Graphics (SIGGRAPH '01 Proceedings)*, pp. 341–346, August 2001.
13. M. Gleicher. "Image snapping". *Computer Graphics (SIGGRAPH '95 Proceedings)*, pp. 183-190, August 1995.
14. P. Harrison. "A Non-Hierarchical Procedure for Re-Synthesis of Complex Textures". *WSCG'2001*. February 2001.
15. D. J. Heeger and J. R. Bergen. "Pyramid-Based Texture Analysis/Synthesis". *Computer Graphics (SIGGRAPH '95 Proceedings)*, pp. 229-238. August 1995.
16. A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless and D. H. Salesin. "Image analogies". *Computer Graphics (SIGGRAPH '01 Proceedings)*, pp. 327-340, August 2001.
17. H. Igehy and L. Pereira. "Image Replacement Through Texture Synthesis". *International Conference on Image Processing*, volume 3, pp. 186189, October 1997.
18. D. Kurlander and E. Bier. "Graphical search and replace". *Computer Graphics (SIGGRAPH '88 Proceedings)*, pp. 113-120, August 1988.
19. J. P. Lewis. "Texture Synthesis for Digital Painting". *Computer Graphics*, 18, 3, 245-252, 1984.
20. L. Liang, C. Liu, Y. Xu, B. Guo, and H. Shum. "Real-Time Texture Synthesis by Patch-Based Sampling". *ACM Transactions on Graphics*. 20, 3, 127–150, 2001.
21. E. Mortensen and W. Barrett. "Intelligent scissors for image composition". *Computer Graphics (SIGGRAPH '95 Proceedings)*, pp. 191-198, August 1995.
22. B. Oh, M. Chen, J. Dorsey, and F. Durand. "Image-Based Modeling and Photo Editing". *ACM SIGGRAPH 2001,* 433-442, 2001.
23. K. Perlin and L. Velho. "Live Paint: Painting With Procedural Multiscale Textures". *ACM SIGGRAPH 95*, 153-160, 1995.
24. Arno Schodl, Richard Szeliski, David H. Salesin and Irfan Essa. "Video Textures". *Computer Graphics (SIGGRAPH '00 Proceedings)*, pp. 489-498. August 2000.
25. E. P. Simoncelli, W. T. Freeman, E. H. Adelson and D. J. Heeger. Shiftable Multi-Scale Transforms. *IEEE Transactions on Information Theory, Issue on Wavelets 38* (1992), 587–607.
26. K. Sims. "Interactive evolution of equations for procedural models". *The Visual Computer*, 9(8), pp. 466-476, 1993.
27. I. Sutherland. *Sketchpad--a man-machine graphical communication system*. Technical Report 296, Lincoln Laboratory, Massachusetts Institute of Technology, 1963.
28. Li-Yi Wei and Marc Levoy. "Fast Texture Synthesis using Tree-Structured Vector Quantization". *Computer Graphics (SIGGRAPH '00 Proceedings)*, pp. 479-488. August 2000.
29. T. Welsh, M. Ashikhmin and K. Mueller. "Transferring Color to Greyscale Images". *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002)*, 21(3), pp. 277-280, August 2002.