Self-Similarity Based Texture Editing

Stephen Brooks Neil A. Dodgson
University of Cambridge
{Stephen.Brooks, Neil.Dodgson}@cl.cam.ac.uk

Abstract

We present a simple method of interactive texture editing that utilizes self-similarity to replicate intended operations globally over an image. Inspired by the recent successes of hierarchical approaches to texture synthesis, this method 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. We instead locate similar neighborhoods 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.

Keywords: Texture editing, interactive, image cloning, texture warping.

1. Introduction

The most broadly applied approach to modeling the complexity of the natural world is to provide the scene designer with sophisticated tools that permit a high degree of control over geometric surfaces and their corresponding textures. This approach has enjoyed considerable success, yet the sophistication of the editing tools requires a comparable level of sophistication from the user. Often, the user must be a highly skilled artist as well as having considerable technical training and experience with computers. These prerequisites are beyond many users.

Recently, attempts have been made to automate the process of constructing graphical objects of sufficient realism. However, the integration of automation with user preference remains an open problem in the context of texture editing.

The primary contribution of this paper is a mechanism which allows the user to perform replicated texture editing operations with minimal input. A single editing operation at a given location causes global changes, affecting all similar areas of the texture (Figure 1). The style of interaction lies between automation and complete user control.

For the sake of brevity, the following section reviews only the principal related papers.

2. Previous Work

As multi-resolution approaches to texture synthesis provide the inspiration for the present work it is appropriate to begin with the pioneering work of Heeger and Bergen [1995]. In their system they create a new instance of a texture through hierarchical histogram matching. De Bonet [1997] 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 [2000].

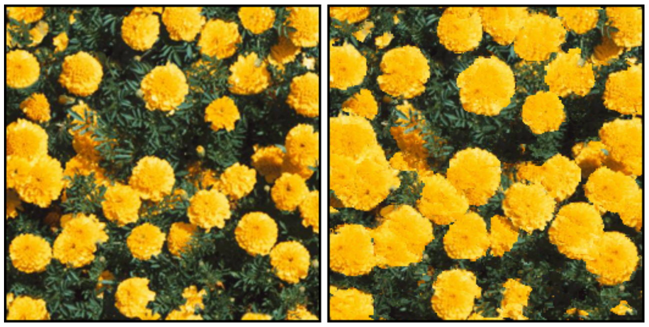
Figure 1 Flowers expanded using self-similarity based texture warping.

But the distinction between texture synthesis and texture editing has become blurred with the development of user-directed texture transfer and synthesis methods [Ashikhmin 2001; Efros and Freeman 2001; Hertzmann et al. 2001; Liang et al. 2001]. 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 [Bar-Joseph 2001] and image compositing [Burt and Adelson 1983; Porter and Duff 1984] that deserve mention.

Other semi-automated texture creation systems include Live Paint [Perlin and Velho 1995], which uses the concept of a multi-resolution painting system [Berman et al. 1994] to combine procedural textures [Ebert et al. 1994]. Dischler et al. [1999] describe a unique hybrid approach that combines texture analysis and geometric modeling. Lewis' [1984] 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 [Sims 1993].

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 [Mortensen and Barrett 1995] and snapping [Gleicher 1995] tools. Elder and Goldberg [1998] also offer a novel editing system that operates in an invertible contour domain.

Although neighborhood-based texture synthesis methods are algorithmically closest to the present work, conceptually, it is the vector-object search and replace method of Kurlander and Bier [1988] that bears closest resemblance to our own. But, their system differs significantly as it operates strictly on vector images that are composed of distinct geometrically defined objects unlike our raster based editing tools.

This is a preprint of the paper which appeared, in its final form, in ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002), 21(3), Jul 2002, pp. 653-656, ISSN 0730-0301, ISBN 1-58113-521-1

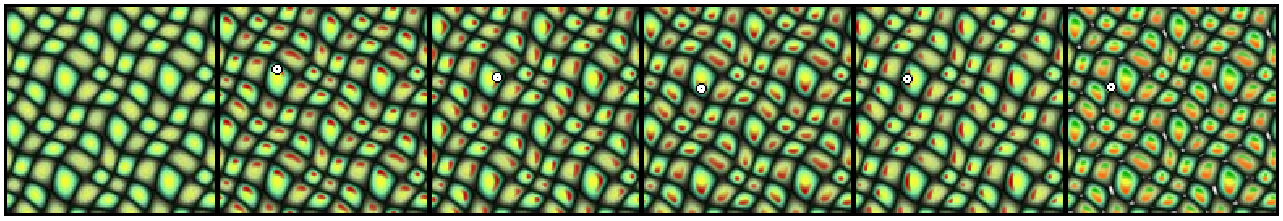
Figure 2 Similarity Painting: (leftmost) original texture, (center four) single point painting, and (rightmost) multiple paintings applied. Strength = 100%, Distance = 800K, d = 9, Level 0.

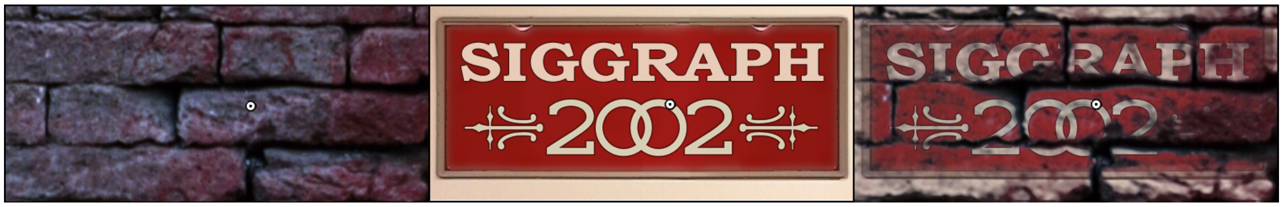

Figure 3 Similarity Cloning: (left) original texture, (center) cloning texture, (right) single point cloning. Strength = 75%, Distance = 500K, d = 5, Levels 0,1.

3. Similarity-Based Editing

Our editing system uses neighborhood self-similarity within a texture to edit the original texture itself. Changes made to a particular pixel by the user are made to affect all pixels that exhibit similar local neighborhoods at multiple scales.

Self-similarity based painting alters the color or brightness of similar pixels to that which the user selects (Figure 2). To perform a replicated painting operation, the user moves what we will refer to as the selection point onto a pixel within the original image. The local circular neighborhood of the chosen selection point is then compared against that of every other pixel's neighborhood in the image. The current painting color is then applied to the selected pixel but also to a subset of all pixels in the image: those that have local neighborhoods whose difference from the selected pixel are within a certain threshold.

The definitions of neighborhood, threshold and neighborhood similarity require clarification. 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 2 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 $256^2*3*69 \approx 13.5$M.

The selection point receives full paint opacity, as do all pixels with neighborhoods identical to it. 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.

The green texture in Figure 2 succinctly conveys the way in which self-similarity based editing operates from the user's perspective. With the original texture leftmost, the four center images show four consecutive positions that the user has selected within the texture. The reader will note two significant visual properties. The first is the soft gradient of each painted region. The second is the directional control of the tool. By directional control we refer to the ability of the user to successively select the 'top', 'right', 'bottom' then 'left' areas of the texture elements. These aspects would not be present if only the pixels themselves were compared without the neighborhood metric. And although this texture was chosen as a strong example of these behaviors for the purpose of illustration, both of these properties are exploitable to a greater or lesser extent in most textures.

The left graph in Figure 4 depicts the altering of the global opacity multiplier (Strength) while maintaining a constant threshold (Distance) of 1.5 million. Conversely, the right 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.

Moving from replicated painting to replicated cloning is plainly a matter of positioning the cloning texture and using the corresponding color values from the cloned texture instead of a solid RGB value. Figure 3 shows an example of cloning a

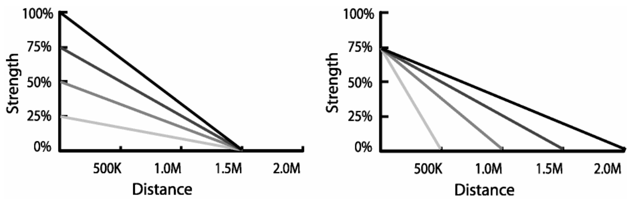

Figure 4 Altering global strength (left) or distance threshold (right).

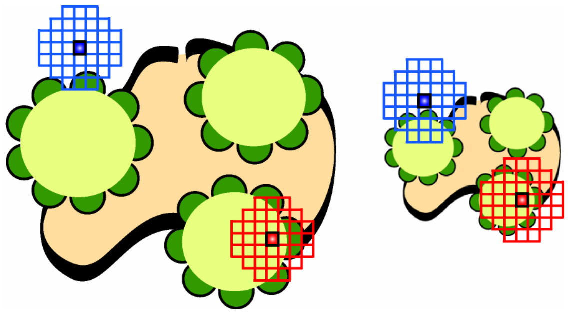
Figure 5 Multi-resolution neighborhood comparisons. Level n (left) and level n+1 (right). User selected pixel shown in red.

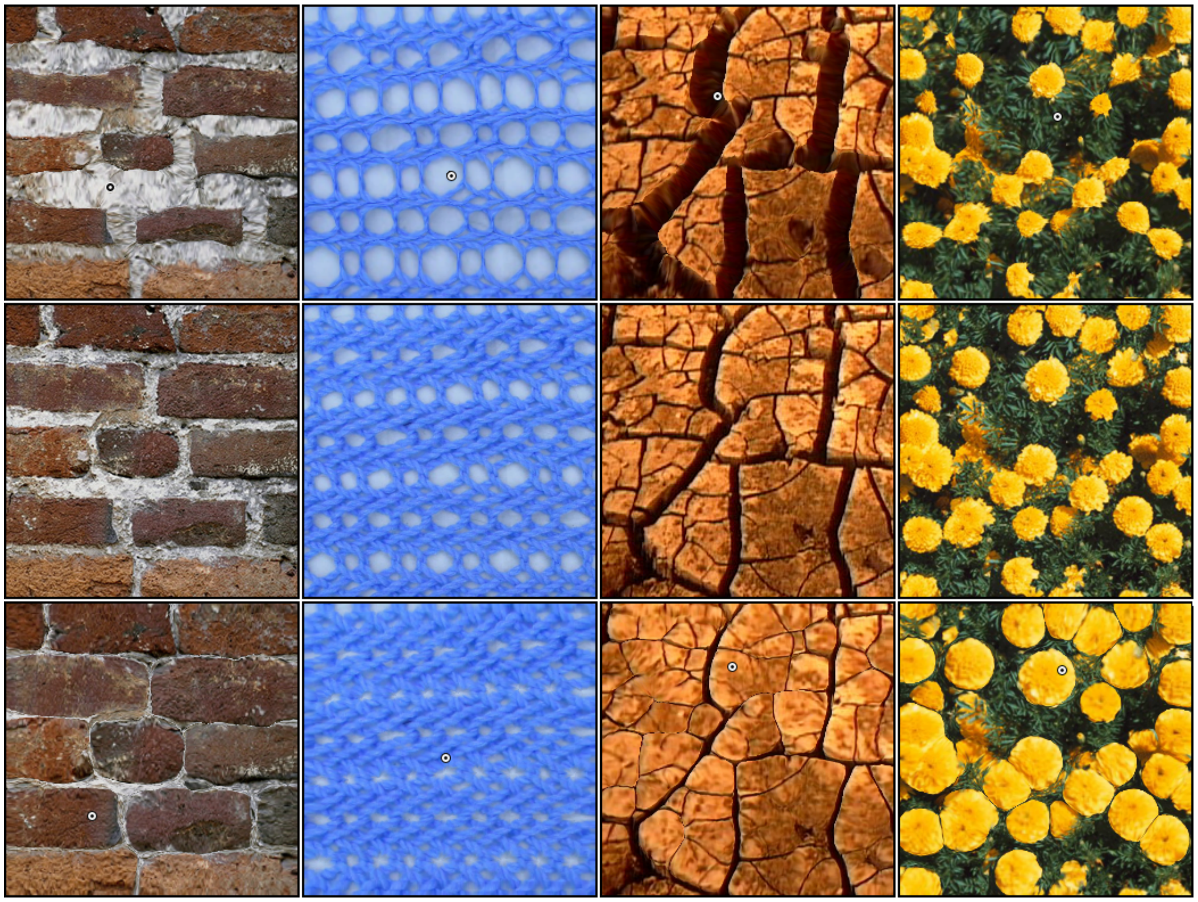

Figure 6 Similarity Based Warping: Middle row contains original textures. Rows above and below show warped textures. The selection point is indicated with a white circle. Those areas that are similar to the selected pixel are expanded. Strength = 85%, Distance = 1.2M(bricks); 800K(rest), *d* = 5, Levels 1,2.

structured sign onto a brick texture. But, since the brick image is a much larger texture, interactivity is jeopardized if we use the neighborhood metric as previously defined.

To avoid sluggish response times with large textures, we amend the neighborhood to be multi-scaled. As in [Ashikhmin 2001; Hertzmann et al. 2001; Wei and Levoy 2000], we first construct a Gaussian pyramid from the original texture where level 0 of the pyramid is the original texture itself and level n+1 is a filtered, down-sampled version of level n. An illustration of two successive levels of an image pyramid is shown in Figure 5. The neighborhood of the selection point is now a concatenation of two circular areas (two red neighborhoods in Figure 5) both of diameter d. But we note that the circular area of level n+1 represents a larger portion of the original texture. For example, in Figure 3 the diameter $d = 5$, yielding two neighborhoods of 21 pixels each. This requires summing 42 squared R, G and B differences, a number in the range of 0 to $256^2*3*42 \approx 8.2$M. By incorporating higher-level neighborhoods into the similarity metric we take into account a wider area at a lower cost, while the lower level neighborhoods retain priority for nearer pixels.

As currently implemented the system performs painting and cloning operations on 512×512 textures at 5 fps, using two neighborhood levels each with a diameter $d = 5$, running on an off-the-shelf 1.2 GHz Athlon PC. In our experience this frame rate is sufficient for the few, simple actions that the user must perform, though this is strictly a subjective claim requiring an appropriate HCI study for confirmation.

3.1 Similarity-based warping

In addition to published research [Glasbey and Mardia 1998] there are a number of commercially available image warping tools, many of which focus on morphing one image into another [Beier and Neely 1992] but others operate strictly as an image editing tool [Keahey et al. 1997]. Similarity-based texture warping belongs to the latter category and uses neighborhood similarity as a measure of local area expansion (Figure 6). The question becomes how to convert scalar similarity values derived from neighborhood distances into 2D area expansions (Figure 7).

To accomplish this we borrow the interactive image-warping scheme of Keahey et al. [1997]. In their notation, the grid of similarity values defines a magnification function, M, from which a 2D grid displacement function, T, must be derived. The magnification function, M, is essentially the derivative of the desired T, and a numerical algorithm is used to approximate the integration of M, yielding an estimate, T_C, at each iteration. The corresponding approximate magnification function, M_C, can be directly computed from T_C, allowing the resultant error, $M_E = M - M_C$, to be calculated. T_C is then further modified on a vertex-by-vertex basis. Effectively, the neighboring vertices are moved outwards in T_C where $M_E > 0$, and drawn inwards where $M_E < 0$, yielding a better approximation. From this, a 2D transformation is produced that is both symmetric and centered around magnification maxima. The algorithm benefits from various optimizations detailed by Keahey and our implementation converges in less than 0.1s.

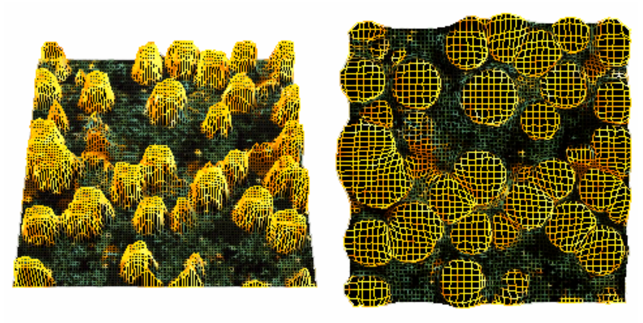
Figure 7 From scalar similarity values (left) to 2D texture warp (right).

Self-similarity scalar values are directly used to interactively drive area magnification. If in painting mode the pixel would have received 75% opacity, the local area instead increases by 75%. An example of its use can be seen in the fourth column of Figure 6, where yellow chrysanthemums have been contracted (top) and expanded (bottom).

But, depending on the texture and on the amount of expansion the warped texture can suffer a loss of high frequency detail. We 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. [2001]. A final result of which can be seen in Figure 1. The reader may, however, speculate that a yellow and green constraining image could be manually painted in any image editor and used directly for super-resolution synthesis. Although this is indeed the case, our technique again provides a more concise form of interaction for this task, as all flowers are altered interactively with a single mouse movement.

For warping, 256×256 textures are hardware textured over a warped 2D mesh (Figure 7, right). We find that the mesh itself need only be 128×128 for quality results and with these parameters a frame rate of 6 fps is achieved.

4. Conclusion and Future Work

We have presented an editing system that replicates painting, cloning and warping operations over a texture. The significance of this is that it allows the user to perform complex tasks in real-time with minimal effort. There remain however significant opportunities to extend the capabilities of this system.

Currently our approach works best for textures which are uniformly lit. Non-uniform lighting leads to poorer results. For texture cloning, it is also required that the two textures be roughly coplanar for convincing results. We believe that these restrictions might be addressed by integrating similarity based editing with a photo editing system such as that of Oh et al. [2001] which permits both distortion free cloning and texture illumination correction.

Further replicated editing operations might be explored such as similarity based filtering or texture-transfer. Moreover, a variation of texture cloning is being considered that would synthesize the cloning texture directly into areas of similarity. This introduces the potential for further control of how the cloning texture is generated into areas that have greater and lesser degrees of similarity to the user-selected point.

Additional possibilities include the use of higher order similarity statistics, such as Bayesian 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. Lastly, an alternative to replicated operations over similar areas would be to duplicate editing operations along texture contours. An instance of which would be direct texture synthesis along a chosen contour.

5. Acknowledgements

We thank Marc Cardle and all those in our lab who provided valuable feedback. This work was supported by the Cambridge Commonwealth Trust and an Overseas Research Scholarship.

6. References

ASHIKHMIN, M. 2001. Synthesizing Natural Textures. *ACM Symposium on Interactive 3D Graphics*, 217–226.

BAR-JOSEPH, Z., EL-YANIV, R., LISCHINSKI, D., AND WERMAN, M. 2001. Texture Mixing and Texture Movie Synthesis Using Statistical Learning. *IEEE Transactions on Visualization and Computer Graphics*, 7, 2, 120-135.

BEIER, T., AND NEELY, S. 1992. Feature-Based Image Metamorphosis. In *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, 26(2), ACM, 35-42.

BERMAN, D., BARTELL, J., AND SALESIN, D. 1994. Multiresolution Painting and Compositing. *ACM SIGGRAPH 94*, 85-90.

BURT, P., AND ADELSON, E. 1983. A Multiresolution Spline with Application to Image Mosaics. *ACM Transactions on Graphics*, 2, 4, 217-236.

DE BONET, J. S. 1997. Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images. *ACM SIGGRAPH 97*, 361-368.

DISCHLER, J., AND GHAZANFARPOUR, D. 1999. Interactive Image-Based Modeling of Macrostructured Textures. *IEEE Computer Graphics and Applications*, 19, 1, 66-74.

EBERT, D., MUSGRAVE, F., PEACHEY, D., PERLIN, K., AND WORLEY, S. 1994. *Texturing and Modeling: A Procedural Approach*. AP Professional.

EFROS, A., AND FREEMAN, W. 2001. Image Quilting For Texture Synthesis and Transfer. *ACM SIGGRAPH 2001*, 341–346.

ELDER, J., AND GOLDBERG, R. 1998. Image Editing In the Contour Domain. *IEEE Computer Vision and Pattern Recognition*, 374-381.

GLASBEY, C., AND MARDIA, K. 1998. A Review of Image-Warping Methods. *Journal of Applied Statistics*, 25, 2, 155-172.

GLEICHER, M. 1995. Image Snapping. *ACM SIGGRAPH 95*, 183-190.

HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-Based Texture Analysis/ Synthesis. *ACM SIGGRAPH 95*, 229-238.

HERTZMANN, A., JACOBS, C., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image Analogies. *ACM SIGGRAPH 2001,* 327-340.

KEAHEY, A., AND ROBERTSON, E. 1997. Nonlinear Magnification Fields. *IEEE Symposium on Information Visualization*, 51-58.

KURLANDER, D., AND BIER, E. 1988. Graphical Search and Replace. In *Computer Graphics (Proceedings of ACM SIGGRAPH 88)*, 22(4), ACM, 113-120.

LEWIS, J. P. 1984. Texture Synthesis for Digital Painting. *Computer Graphics*, 18, 3, 245-252.

LIANG, L., LIU, C., XU, Y., GUO, B., AND SHUM, H. 2001. Real-Time Texture Synthesis by Patch-Based Sampling. *ACM Transactions on Graphics*. 20, 3, 127–150.

MORTENSEN, E., AND BARRETT, W. 1995. Intelligent Scissors for Image Composition. *ACM SIGGRAPH 95,* 191-198.

OH, B., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-Based Modeling and Photo Editing. *ACM SIGGRAPH 2001,* 433-442.

PERLIN, K., AND VELHO, L. 1995. Live Paint: Painting With Procedural Multiscale Textures. *ACM SIGGRAPH 95*, 153-160.

PORTER, T., AND DUFF, T. 1984. Compositing Digital Images. *Computer Graphics*, 18, 3, 253-259.

SIMS, K. 1993. Interactive Evolution of Equations for Procedural Models. *The Visual Computer*, 9, 8, 466-476.

WEI, L., AND LEVOY, M. 2000. Fast Texture Synthesis Using Tree-Structured Vector Quantization. *ACM SIGGRAPH 2000*, 479-488.