

I. Introduction

Studio580 is an interactive, digital painting system. It belongs to a class of systems traditionally called non-photorealistic paint systems, which are in turn part of the broader endeavor of non-photorealistic computer graphics or “NPR” [10]. The goals of NPR differ from those of traditional computer graphics, so before diving headlong into our discussion of Studio580, a few words about NPR as a field are warranted.

In contrast to NPR, traditional computer graphics is centered on the visual simulation of reality. To illustrate this fact, take, for example, two of the most visible applications of computer graphics: the computer-generated, special effects imagery that enlivens Hollywood movies, and the visually-rich video games that run on the computers inside of console gaming systems like the Sony Playstation* or the Microsoft Xbox**. In both cases, what impresses us most is the realism of the images involved. The quest for visual realism is the defining enterprise of traditional computer graphics [9].

By contrast, non-photorealistic computer graphics does not seek to create images whose realism mimics that of a photograph. NPR eschews the photorealistic methods of traditional computer graphics and aims instead to formulate new methods for the creation of digital images. These methods produce images that sacrifice realism to attain other desirable qualities, such as interpretation, stylization, or communication [10].

In this sense, NPR is more analogous to one of the classical visual arts, like painting or drawing, than to traditional computer graphics. After all, as far as realism is concerned, a paintbrush is a poor substitute for a camera. But this doesn’t make painting a less valuable art than photography. Quite to the contrary, painters know that the strength of their medium lay not in its ability to faithfully reproduce reality, but in its capacity to empower the artist to alter, stylize, and re-interpret reality. Such expressive techniques as overloading brushes with pigment enriched in linseed oil, and placing strokes so as to trace the contours and crease edges of objects, enable painters to emphasize subtle qualities and features of their subjects that would be subsumed in a purely verbatim reproduction [13].

NPR aims to empower digital artists with the kinds of expressive tools that traditional painters and illustrators have long enjoyed. As such, a major area of research interest within NPR has been the digital simulation of traditional artistic media [10]. Indeed, arguably the first major scientific paper on an NPR-related subject ever published, Steve Strassman’s seminal “Hairy Brushes” of 1986, focused on the digital simulation of a Sumi-e, a traditional Japanese painting style [18]. In the two decades since “Hairy Brushes,” many further papers have appeared in the various ACM and IEEE journals, describing digital simulations of a wide range of traditional artistic media, such as watercolor [5], charcoal [4], pencil [17], oils [3], and pen-and-ink [22].

* Playstation is a registered trademark of Sony USA, Inc.

** Xbox is a registered trademark of Microsoft Corp.

Studio580 is a research demonstration application that continues this tradition. There are, however, a number of features that make Studio580 unique when compared to previous natural media simulation systems.

II. Factors Differentiating Studio580 from Similar Systems

a. Studio580 uses adaptive algorithms based on image statistics, not style-determined algorithms.

Most non-photorealistic paint and draw systems work by digitally simulating the behavior of some particular traditional artistic style or medium, or a small number of such styles or media [3, 4, 5, 17, 18, 22]. As an example, consider the pencil simulation of system Sousa & Buchanan [17]. Sousa & Buchanan built their system by painstakingly examining electron micrographs of pencil strokes, gradually formulating an empirical model of how pencil lead is deposited on the grain fibers of drawing paper. Once they had their model, Sousa & Buchanan developed computer algorithms that simulated the behavior their model described. The resulting system was extremely adept at producing digital pencil drawings, but the ideas embodied in it cannot easily be applied to other media or artistic styles, such as watercolor, oils, or pen-and-ink. In this sense, Sousa & Buchanan’s system is style-determined. This is to say, its creators explicitly determined at design time that their system would simulate only pencil drawing, and as a result, it was implemented solely for this purpose. The algorithms Sousa & Buchanan have developed are specific to pencil simulation [17].

By contrast, Studio580 uses ideas from the texture synthesis [2, 11, 12, 14, 19, 20, 21, 23] and image processing [1, 11] to implement more general algorithms that are not constrained to a single, pre-determined, artistic style or medium, or to a small, finite set of styles or media. In fact, given a suitable source or “exemplar” image, Studio580 is capable of simulating almost any artistic style imaginable. The system works as follows.

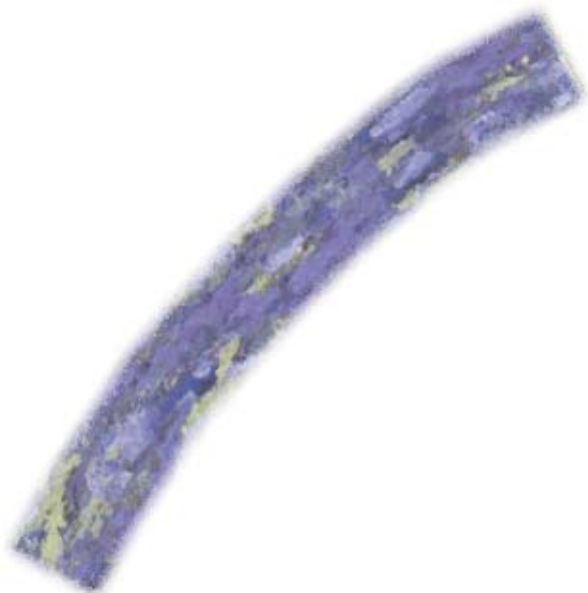
The user provides a small, square image called the *source image* or *exemplar*. The source image should be paradigmatic of—or capture the essence of—some artistic style or medium that the user wishes to simulate. Studio580 then uses an adaptive, multi-scale [1], Markov random-field algorithm [7, 14] to develop a probability model of the distribution of color and tone across the surface of the user-provided source image. Once this probability model is constructed, it can be sampled as many times as necessary. Repeated sampling allows Studio580 to “grow” arbitrary amounts of image data that have the same spatial distribution of color and tone as the source image. Furthermore, since similar spatial distributions of color and tone between images correlate well with human perceptions of “sameness” or “similarity” between images [14], the image data grown by Studio580 appears perceptually

similar to the source image furnished by the user. Insofar as a user's source image is able to capture the visual essence of a particular style or medium, Studio580 enables that user to paint or draw in the same style as the source image at interactive speeds. An example is shown in figure 1 below.

b. Studio580 makes full use of modern, programmable graphics hardware

Between 2000 and 2003, the emergence of inexpensive, programmable, graphics processing units (or GPUs) for PCs and workstations dramatically altered the interactive graphics landscape. Ultimately, most graphics computations are arithmetic operations over vectors

Figure 1. Two brush strokes generated from stylistically distinct source images. On the left, a patch of a “loaded” oil stroke, taken from Van Gogh’s *Starry Night*, is used as a source image. On the right, a patch of a “hairy” oil stroke from a Daumier painting is used as a source image. The brush strokes appearing beneath their respective source images were generated at interactive speeds by Studio580. At left, Studio580 has grown image data similar to the Van Gogh source image and texture-mapped it onto a stroke drawn by the user. At right, Studio580 has grown image data similar to the Daumier image and texture-mapped it onto another, shorter user stroke.



of floating-point numbers. Modern GPUs are specialized devices built around ASICs optimized for just these kinds of floating-point, SIMD computations. What’s more, they usually contain several such floating-point, SIMD pipelines operating in parallel. As a result, GPUs can perform the raw arithmetic calculations of computer graphics many times faster than can general-purpose CPUs.

But all of this performance comes at a price. Because their stream computational model differs radically from the one exposed by the Von Neumann architectures of traditional CPUs, GPUs cannot be programmed using conventional languages or techniques. Instead, GPUs are programmed via specialized *shading languages* [8, 16] better suited to their unique computational model. Here, the implementation of even moderately complex computational procedures usually requires the design of clever multi-pass algorithms.

The texture synthesis subsystem is by far the most complex and computationally demanding component of Studio580. It is responsible for dynamically “growing” brush texture information similar to the user-provided source image in real-time, and is implemented almost entirely on the GPU. Indeed, without the number-crunching acumen of modern GPUs, execution of the texture synthesis subsystem at interactive speeds would be impossible. At the heart of the subsystem is a progressive algorithm that grows the output brush texture in a *multi-scale image pyramid* [1], upsampling the output to a higher resolution and refining its appearance at each level. The refinement process compares the neighborhood around each pixel in the intermediate output pyramid against many different neighborhoods in the source image, searching for perceptual similarity. This search process, requiring the evaluation of a whole-neighborhood similarity metric at each search point, makes texture synthesis a very computationally-intensive endeavor. So much so, in fact, that as recently as 2000, real-time texture synthesis seemed a pie-in-the-sky objective that was well out-of-reach any time soon. In a notable SIGGRAPH paper published that year, Wei & Levoy reported synthesis timings on the order of minutes to grow a 128×128 patch of texture, on then state-of-the-art workstation-class CPUs. On a 2006-era GPU, Studio580 grows several times that amount of texture in less than a second.

III. Previous Work

Studio580 is a multifaceted system that uses texture synthesis to enable its user to interactively produce non-photorealistic digital paintings. As such, it is indebted to previous work across several different sub-fields of computer graphics, including:

a. Hardware-accelerated texture synthesis

Studio580 characterizes texture via an implicit Markov Random Field [7, 14] model. This model is sampled by matching pixel neighborhoods in the intermediate output to perceptually similar neighborhoods in the user-provided source image and directly copying pixels from source to output. Schemes implementing Markov Random Fields through neighborhood matching have a long history [2, 7, 11, 12, 14, 19, 20, 21, 23] and have become one of—if not the—dominant approach to texture synthesis.

Popat and Picard laid the foundations of the approach in 1993, and their paper [14] remains the best source for understanding the theory behind neighborhood matching. Moreover, some of the theoretical problems they explored became important in practice. For example, they discuss how matching and replacing pixels one-after-the-other introduces dependencies on the order in which these replacements are made. These dependencies complicate the implementation of neighborhood matching on parallel systems like GPUs, and are a barrier to exploiting the potential of programmable graphics hardware.

In 1999, Efros and Leung presented a system [7] based on Popat and Picard’s ideas that set new standards for synthesis quality and user control. Over the next few years, Ashikhmin [2], Wei and Levoy [20], Tong, et. al [19], Zelinka and Garland [23], and Hertzmann, et. al. [11] published descriptions of systems that improved on the state-of-the-art, introducing ideas like coherent synthesis [2], TSVQ acceleration [20], pre-computed maps of perceptually similar source regions [19, 23], k -coherence search [19], variety boosting [23] and jump penalization [11]. With the exception of TSVQ acceleration—which is unsuited to GPU implementation—all of these ideas are implemented in Studio580.

The major breakthrough enabling neighborhood matching schemes to be implemented on programmable graphics hardware came in 2003, when Wei and Levoy presented a system for what they called *order-independent texture synthesis* [21]. Using ideas from image processing, they devised a clever way to avoid the ordering dependencies that complicated parallelization. In 2005, Hoppe and Lefebvre unified Wei and Levoy’s order-independence with the state-of-the-art in synthesis quality and control, publishing a description of a neighborhood-based system that allowed texture to be grown in real-time by running almost entirely on the GPU [12]. It is this system to which Studio580 is most deeply indebted.

b. Texture synthesis applied to non-photorealistic rendering

In addition to the technical advance of jump penalization, as described above, the Image Analogies framework of Hertzmann, et. al., was also significant in that it applied neighborhood-based texture synthesis to transfer the look-and-feel of artistic styles from one image to another [11]. Alexi Efros, who worked with T. K. Leung on a significant neighborhood-based texture synthesis system, later presented a system developed with W. T. Freeman that

used patch optimization—a synthesis approach that competes with neighborhood matching—to produce a broad range of texture transfer effects, including the transfer of artistic styles [6]. Unlike Studio580 however, both of these are non-interactive, automatic systems, that take complete images as input and run a batch program to produce a complete output image with little or no user interaction.

c. Stroke capture and interactive techniques

Studio580 is first a painting system, then a texture synthesis system. As such, it relies on previous work in interactive paint techniques. Studio580’s stroke capture subsystem, for example, is adapted directly from Steve Strassman’s classic Hairy Brushes [18]. One of its key interactive techniques—the use of a simplified proxy preview stroke that is displayed while the user drags the mouse—was first demonstrated in a paint system developed by Pudet [15].

IV. Design & Implementation

One key design goal of Studio580 was that the system would not be “just a demo.” From the outset, Studio580 was designed to be a practical tool—albeit an intentionally limited one—that would allow its users to create real artwork within a double-clickable, desktop application. Furthermore, this practical goal wasn’t at odds with the system’s academic objectives. Indeed, the two are largely coincident. Since the system’s core objective was to evaluate the application of texture synthesis to interactive painting, it was natural to develop and refine its interactive aspects into a complete application. This meant that considerable design energies were directed at interactive concerns like visual feedback, user access to algorithm parameters, easy mistake correction, and intuitive stroke behavior.

a. System Overview

Studio580 logically comprises four subsystems: stroke capture, stroke assembly, texture synthesis, and user control. The responsibilities delegated to each of these subsystems and the relationships between them are summarized in figure 2 and discussed below.

incomplete draft only—content stops here—bibliography follows

*downloading this paper a week from now will likely yield a longer version, as
I will have had time to write more*

Works Cited

1. Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., and Ogden, J. M., "Pyramid Methods in Image Processing" in *RCA Engineer* 29(6). New York: RCA Press, 1984.
2. Ashikhmin, M. "Synthesizing Natural Textures" in *Proceedings of the 2001 ACM Symposium on Interactive 3D Graphics*. New York: ACM Press, 2001.
3. Baxter, W., Wendt, J., and Lin, M. C. "IMPASTo: a Realistic, Interactive Model for Paint" in *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering*. New York, ACM Press: 2004.
4. Blesser, T. W., Sibert, J. L., and McGee, J. P. "Charcoal Sketching: Returning Control to the Artist" in *ACM Transactions on Graphics* 7(1). New York: ACM Press, 1988.
5. Curtis, C. J., Anderson, S. E., Seims, J. E., Fleischer, K. W., and Salesin, D. H., "Computer-Generated Watercolor" in *Proceedings of SIGGRAPH '97*. New York: ACM Press, 1997.
6. Efros, A. A. and Freeman, W. T., "Image Quilting for Texture Synthesis and Transfer" in *Proceedings of SIGGRAPH 2001*. New York: ACM Press, 2001.
7. Efros, A. A. and Leung, T. K., "Texture Synthesis by Non-Parametric Sampling" in *Proceedings of the 1999 IEEE International Conference on Computer Vision*. New York: IEEE Press, 1999.
8. Fernando, R. and Kilgard, M. *The CG Tutorial: The Definitive Guide to Programmable Real-Time Graphics*. Reading, Massachusetts: Addison-Wesley, 2003.
9. Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F., *Computer Graphics: Principles and Practice*. Reading, Massachusetts: Addison-Wesley, 1982.
10. Gooch, B. and Gooch, A. A., *Non-Photorealistic Rendering*. Wellesley, Massachusetts: A. K. Peters, Ltd., 2001.
11. Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., and Salesin, D. H., "Image Analogies" in *Proceedings of SIGGRAPH 2001*. New York: ACM Press, 2001.
12. Hoppe, H. and Lefebvre, S., "Parallel Controllable Texture Synthesis" in *Proceedings of SIGGRAPH 2005*. New York: ACM Press, 2005.

13. Janson, H. W. and Janson, A. F., *History of Art*, 5/e. New York: H. N. Abrams, Inc., 1995.
14. Popat, K., and Picard, R. W., "Novel Cluster-based Probability Model for Texture Synthesis, Classification, and Compression" in *Visual Communications and Image Processing '93*. Bellingham, Washington: SPIE Press, 1993.
15. Pudet, T., "Real-Time Fitting of Hand-Sketched Pressure Brushstrokes" in *Computer Graphics Forum* 13(3). Oxford: Blackwell, Ltd., 2000.
16. Rost, R. J. *The OpenGL Shading Language*. Reading, Massachusetts: Addison-Wesley, 2004.
17. Sousa, M. C. and Buchanan, J. W., "Observational Models of Graphite Pencil Materials" in *Computer Graphics Forum* 19(1). Oxford: Blackwell, Ltd., 2000.
18. Strassman, S., "Hairy Brushes" in *Proceedings of SIGGRAPH '86*. New York: ACM Press, 1986.
19. Tong, X., Zhang, J., Liu, L., Wang, X., Guo, B., and Shum, H. "Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces" in *Proceedings of SIGGRAPH 2002*. New York: ACM Press, 2002.
20. Wei, L.-Y. and Levoy, M. "Fast Texture Synthesis Using Tree-Structured Vector Quantization" in *Proceedings of SIGGRAPH 2000*. New York: ACM Press, 2000.
21. Wei, L.-Y. and Levoy M. *Order Independent Texture Synthesis*. Electronic Publication of Stanford University (2003): http://graphics.stanford.edu/papers/texture-synthesis-sig03/texture_submitted.pdf, retrieved 11/16/2008.
22. Winkenbach, G. and Salesin, D. H., "Computer Generated Pen-and-Ink Illustration" in *Proceedings of SIGGRAPH '94*. New York: ACM Press, 1994.
23. Zelinka, S. and Garland, M. "Towards Real-Time Texture Synthesis with the Jump Map" in *Proceedings of the 13th Eurographics Workshop on Rendering*. Aire-La-Ville, Switzerland: Eurographics Association, 2002.