Example-based Creation of Digital Imagery

Ing. Michal Lukáč

A dissertation submitted to
the Faculty of Electrical Engineering, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy.

Ph.D. programme: Electrical Engineering and Information Technology
Branch of study: Information Science and Computer Engineering

Co-Supervisor: doc. Ing. Daniel Sýkora, Ph.D.

November 2015
The research presented in this thesis was conducted in collaboration with, and supported by, Adobe Research and has further been supported by the Technology Agency of the Czech Republic under research program TE01020415 (V3C) and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS13/214/OHK3/3T/13 (Research of Progressive Computer Graphics Methods).
EXAMPLE-BASED CREATION OF DIGITAL IMAGERY
Ing. Michal Lukáč
michal.lukac@fel.cvut.cz
Department of Computer Graphics and Interaction
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlov nám. 13, 121 35 Prague 2, CZ

Abstract
In modern computer graphics, the collective creativity of digital artists far outstrips the capabilities of available tools. In the drive to catch up, more traditional simulation-based and procedural tools are being supplemented by example-based methods, which take advantage of the cornucopia of available data as guidance.

This thesis is an anthology of articles published in various journals in the period of 2013 to 2015, which present three such novel methods. The first is an example-based painting approach which focuses on proper preservation of feature-rich boundaries between textures. The second is a video stylization approach which allows fine control over the amount of temporal noise in the output, and the third is another texture painting approach which focuses on handling shape-specific features.

Keywords
Computer graphics, example-based, texture synthesis, texture painting, video stylization, interactive painting

Acknowledgements
I would like to take this opportunity to express my gratitude to the multitude of people who helped make this thesis a reality. I give thanks to my co-supervisor, Daniel Syýkora, for giving me the direction that set me down this path. To my colleagues, Jakub and Ondřej, for helping cover up the fact that I can’t illustrate my way out of a paper bag. To Eli Shechtman, for being a fount of wisdom and experience, and to Paul Asente for introducing me to how things are done in the real world and suffering me as an intern. To Cynthia Lu, for being a role model of sorts, and to Adam Sporka for providing much needed perspective over many a lunch break. And also ultimately to all my co-authors and collaborators, because everything worthwhile is a team effort.

Most of all however, I owe my thanks to my wife Lenka and my parents, for their faith in me and for unflinchingly supporting me throughout.
Contents

1 Introduction 1
   1.1 Example-based Methods ........................................... 1
   1.2 Applications in Image Creation ................................ 5
   1.3 Context and Contribution of the Thesis ........................... 8
   1.4 Thesis Structure .................................................. 9

2 Related Work 11
   2.1 Painting by Feature ................................................ 11
   2.2 Color Me Noisy ..................................................... 13
   2.3 Brushables .......................................................... 13

3 Painting by Feature: Texture Boundaries for Example-based Image Creation 15

4 Color Me Noisy: Example-based Rendering of Hand-colored Animations with Temporal Noise Control 23

5 Brushables: Example-based Edge-aware Directional Texture Painting 33

6 Conclusion 45
   6.1 Summary .......................................................... 45
   6.2 Concurrent and Future Work .................................... 45

A Author’s Publications 53

B Authorship Contribution Statement 55

C Supplementary material for the paper Painting by Feature: Texture Boundaries for Example-based Image Creation 57


E Supplementary material for the paper Brushables: Example-based Edge-aware Directional Texture Painting 81
1 Introduction

Digital image creation, or less formally digital art, is a computerized counterpart of the physical art of painting. It exhibits a significant overlap with other areas, such as digital image processing or signal analysis and from the research side is characterised by a continuous effort to develop new tools which let users create images in ever-higher fidelity while “intelligently” considering the image content.

With the increases of computational power and the volume of image data being processed over the past 30 years, the requirements on fidelity and quality of the result only grew. For instance, selecting objects for editing in images with a resolution of a megapixel or beyond using simple geometric selection was no longer feasible, and so automated selection techniques were developed [Rother et al., 2004]. Similarly, flat-colour, fixed-footprint brushes were no longer acceptable as tools for high-quality artwork, necessitating the development of more sophisticated tools.

Such tools generally fall under the heading of procedural tools; that is, brushes, filters, etc., which have been hand-crafted to achieve a certain artistic effect, be it an imitation of a physical art style or an entirely novel one (see Figure 1.1 for an example). Commercial programs such as Corel Painter take advantage of these tools for the purposes of digital painting. Similarly, algorithms have been devised to generate textures, usually by the expedient of applying post-processing to a noise function [Ebert et al., 2002]. Simultaneously, simulation-based approaches are being developed. In these, physical behaviour of various painting media is simulated to achieve maximum veracity, as shown in Figure 1.2.

Both the procedural and the simulation-based approaches do, however, suffer from an important drawback – each of the tools can only generate one particular style, and new ones have to be carefully designed by programmers who simultaneously have to have a degree of domain expertise. This limits the palette of available tools and styles, and thus the breadth of use of these approaches.

In response to this, example-based approaches (Figure 1.3) were conceived. These approaches aim to use a single algorithm to mimic an unlimited number of artistic styles by the expedient of analysing an example of an arbitrary style and then attempting to replicate it. It is exactly these methods which are the focus of this thesis.

1.1 Example-based Methods

The label “example-based” has been applied to a multitude of methods intended for various applications. What they have in common is that the parameters used for the underlying model for generation of visual content are neither provided by the user, nor determined by simulation, but instead inferred by the analysis of a user-provided exemplar. This makes them significantly more versatile with respect to the visual characteristics of the output, but also presents a challenging research problem; the methods need to be sufficiently flexible to account for various visual styles but not have so many free parameters that learning them from available input data is infeasible. At the same time, the computational expense of analysing the input exemplar needs to be considered. Nonetheless, the attractiveness of their versatility is sufficient to make them an active research topic in many areas of computer graphics.

There are methods for example-based super-resolution [Freeman et al., 2002], colourisation [Bonneel et al., 2013], image filtering [Liu et al., 2014], portrait stylization [Shih et al.,
CHAPTER 1. INTRODUCTION

Figure 1.1: An example of artwork generated by various brushes of a procedural system [Méch and Miller, 2012]

Figure 1.2: Artwork generated by a simulation-based painting system. In this system, behaviour of brush bristles is simulated in order to realistically determine pigment deposition patterns. [Chen et al., 2015]

Figure 1.3: Realbrush by Lu et al. [2013] is an example-based approach for digital painting. The brush stroke examples on the left, painted on paper and captured with a digital camera, are analysed and then used in stroke-wise synthesis to let artist create the image on the right.
Figure 1.4: An illustration of a region-growing synthesis process according to Wei and Levoy [2000]. (a) A pixel neighbourhood considered in filling the pixel \( p \) (b) Various such neighbourhoods in the source are considered according to their similarity with the neighbourhood of the output pixel being filled (c) The top left pixel is being filled. Note that the neighbourhood is considered under toroidal geometry, and the final two lines and columns of the output have been randomly initialized as a seed (d) Partially synthesized output. Note the shrinking of the black undefined region (e) Fully synthesized output.

2014], or paint compositing [Lu et al., 2014b]. Of particular interest for us are methods for Example-based Texture Synthesis, due to the wide variety of applications they are the basis for.

These methods were initially designed to generate infinite textures [Paget and Longstaff, 1998; Lefebvre and Hoppe, 2005]. Later, new methods sharing their theoretical basis have been developed to fill in holes in images [Efros and Leung, 1999; Criminisi et al., 2004; Wexler et al., 2007], create new structured images [Risser et al., 2010], or alter the artistic style of existing images [Hertzmann et al., 2001]. In the remainder of this section, we describe the two basic classes of example-based texture synthesis methods – Region-growing Approaches and Optimization-based Approaches – grouped according to their basic functional principles. The following section describes how the basic texture synthesis methods can be put to use in more complex applications.

### 1.1.1 Region-growing Approaches

Region-growing approaches operate by first dividing the image into a known region with assigned values and an undefined region. Gradually, new values are assigned to pixels in the undefined region (as shown by Figure 1.4), either pixel-by-pixel [Efros and Leung, 1999] or in larger coherent parts [Efros and Freeman, 2001]. When selecting an exemplar fragment to copy into the undefined region, values of adjacent pixels are compared to corresponding values in the assigned region adjacent to the area to be filled in order to make the output visually similar to the exemplar.

Approaches vary by the exact mechanics of the selection [Wei and Levoy, 2003], the order in which new fragments are added [Criminisi et al., 2004] as well as the size of the fragments and the mechanics used to splice them to existing content [Kwatra et al., 2003].

The advantage of these approaches is that they may be seamlessly adapted for de novo texture synthesis by starting with a random seed fragment, or used for hole-filling directly. They have also been adapted for example-based image stylization [Hertzmann et al., 2001], as well as interactive texture painting [Ritter et al., 2006].

The disadvantages are their sensitivity to filling order (as discussed by Criminisi et al. [2004]), and a related issue with synthesizing large-scale structures. The limited size of the area considered before splicing and the fact that all decisions are made locally lead to artefacts where pieces of texture grown from different directions meet. Some approaches would
CHAPTER 1. INTRODUCTION

\[ E = (x; \{z_p\}) = \sum_{p \in X} |x_p - z_p|^2 \]

Figure 1.5: Energy optimization texture synthesis by Kwatra et al. [2005]. (a) shows the progress of the optimization process over iterations, with the stars marking points at which the optimization advanced to the next level of the image pyramid. (b) The pixel-wise energy being optimized. It is defined for a pixel \( x \) and its most similar counterpart in the source \( z_p \), and measures the sum of squared differences over their respective neighbourhoods. Total energy is expressed as the sum over all pixels \( x \). In alternating steps, values of \( x \) are altered and new nearest counterparts are found. (c) shows the exemplar next to the output of the algorithm; shown are the initialization, two intermediate results and the final result (not to scale).

run additional passes of synthesis after the initial one to correct for this (yielding the results shown in Figure 1.6). Later variants have attempted to use coarse-to-fine synthesis to alleviate this problem [Wei and Levoy, 2000], but doing so also introduces additional implementation issues as the information from the less-reliable previous synthesis steps needs to be considered differently from the comparatively more reliable information in the constraints.

1.1.2 Optimization-based Approaches

In optimization-based approaches, an optimization criterion is first defined, with the assumption that optimizing for this criterion yields an output more similar to the exemplar (as in Figure 1.5). Usually the entire output is initialized to random values. The optimization criterion is then evaluated on a single pixel and its neighbourhood, or on the entire image (depending on whether the optimization step is pixel-wise or global), and an optimization or sampling technique is applied to improve its value. The specifics in which these approaches vary are principally the formulation of the optimization criterion, and the optimization process used to improve it.
The original formulation was based on Markov Random Fields and represented the texture as such [Paget and Longstaff, 1998]. The fundamental assumption was that if the output is drawn from the same distribution as is determined by the exemplar, the textures will be similar. The texture was initialised by randomly sampling pixels in the input image and then pixel-wise Gibbs sampling was performed to get the final result. The local conditional probability distribution function was assumed to be homogeneous and was derived from an analysis of all neighbourhoods in the exemplar, effectively using pixel neighbourhood patches to retrieve similar pixels.

This yields a randomized approach that synthesizes textures satisfying this Markovian similarity criterion. As with the region-growing approaches, the size of the considered patch was shown to limit the size of the features that the method is able to consider, but in this case, a pyramid-based coarse-to-fine approach can be designed to be principled; starting with a coarse version of the output, the MRF is sampled until the burn-in period is considered to have finished, whereupon the result is upsampled and used as the initialisation of the next level.

Because the previously described method was computationally intensive, requiring a similarity search for each pixel update, and it was difficult to judge when the sampling converged, Lefebvre and Hoppe [2005] designed an alternate, coordinate-space method based on locally maximising the similarity in each pixel. The advantage of operating on coordinate space was that similarity could be pre-computed, and after each upsampling operation, a new output value would be selected from a set of precomputed similar alternatives to maximize local coherence. In this approach, random perturbation was used to introduce variation. This method, further developed to operate on appearance space [Lefebvre and Hoppe, 2006], remains the basis for the state of art in infinite stationary texture synthesis.

An explicit formulation based on global energy optimization was first developed in the context of video inpainting [Wexler et al., 2007] and later independently applied to infinite texture synthesis [Kwatra et al., 2005]. Both of these approaches define a global energy based on the sum of appearance-based dissimilarities over the set of all patches in the output region. This energy is then minimized using Expectation-Maximization, which has been shown to lead to a reasonable global minimum.

Like the previous group, these algorithms are generally run on a pyramid in a course-to-fine fashion, advancing to the next level on convergence. This requires no situational adaptations to the algorithm, as the objective function can be defined in exactly the same way for all levels.

Initially, these approaches were impractical because of the need to compute a dense nearest-neighbour field. However, after the development of PatchMatch [Barnes et al., 2009] global optimization at interactive rates became feasible. In addition, the generality of an energy-based approach allowed for adaptation to a variety of situations. Energy formulation has been extended to be bi-directional and used for image and video summarisation [Simakov et al., 2008], generalized to arbitrary dimension and used for synthesis of volume textures [Kopf et al., 2007] or adapted for multiple exemplars to facilitate a type of texture interpolation [Darabi et al., 2012]. An example of the latter application is shown in Figure 1.7.

### 1.2 Applications in Image Creation

Image creation is a more difficult problem than texture synthesis, as images, unlike textures, have semantically meaningful high-level structure that needs to be imposed or preserved. Automatically synthesising this high-level structure is out of reach of current technology, nor is it
CHAPTER 1. INTRODUCTION

Figure 1.6: Selected results from GraphCut Textures [Kwatra et al., 2003], a region-growing synthesis approach.

Figure 1.7: A result of Image Melding Darabi et al. [2012], an optimization-based approach, being used to interpolate between two different textures.
ultimately desirable, as the users typically want to be able to define the semantics of the image. Therefore, there are multiple possible approaches to creating a digital image while preserving visual style of an example. Generally, an example-based texture synthesis approach will be used to provide visual detail, with some sort of guidance integrated into it to ensure that the right visual elements are transferred to the appropriate portions of the image. According to how the guidance interacts with the synthesis, we may divide the available approaches into several categories:

**Structure-preserving Variation Synthesis** creates new images from a set of examples, with a general underlying assumption that the high-level structure is identical and images only vary in details (see Figure 1.8). The intended purpose is to expand a set of images with freshly synthesized ones in such a fashion that no detectable repetition occurs. The high-level structure may be inferred from the images being mutually spatially aligned [Risser et al., 2010], or a registration algorithm may be used to align input images in specialised cases [Assa and Cohen-Or, 2012].

**Example-based Painting** has the user define the high-level structure manually in its entirety. Guidance determined from user interaction (ie. brush strokes, vector primitives, etc.) is then imbued with a particular visual style similarly as if they had a fill colour assigned to them (see Figure 1.9). Such approaches may be based directly off of a texture synthesis approach by manipulating its initialization [Ashikhmin, 2001], they may use texture synthesis to interactively add texture to a user-created segmentation map [Ritter et al., 2006], or they can independently synthesize individual strokes based on a database of example strokes [Lu et al., 2013].
CHAPTER 1. INTRODUCTION

Figure 1.10: An example of video stylisation by Bénard et al. [2013]. From the rendered input sequence on the bottom, the artist manually stylises the leftmost a rightmost frames (shown above). The stylisation is then propagated to the intermediate frames.

Example-based Image Stylisation uses a target image beside the exemplar to provide structure and guide the synthesis. They generally behave as stylisation filters and may accept regular images for stylisation, or may work as analogies by defining a guidance map for both the target and the exemplar and synthesizing the output image on the basis of that [Hertzmann et al., 2001]. It is also applicable for video, as shown in Figure 1.10.

1.3 Context and Contribution of the Thesis

The core of this thesis consists of three distinct developments set in the context defined by previous sections. In this section, we provide a brief summarisation of each, along with a brief discussion of related work against which these are set. More detailed discussions are available in the following chapters, as detailed by the next section.

1.3.1 Painting by Feature: Texture Boundaries for Example-based Image Creation

Painting by Feature is an example-based painting approach which focuses on handling of edge features along with interior textures. Like previous approaches [Ashikhmin, 2001; Ritter et al., 2006], it follows an interactive workflow. Compared to Painting with Texture [Ritter et al., 2006], the key contribution is the explicit handling of linear features using a separate line synthesis approach.

This gives a better quality result than the use of segmentation masks in the previous approach, as it avoids overconstraining the texture synthesis. Free-form and rapid line and area feature selection also give the user a much greater degree of freedom in selecting the features they want to replicate and permits the use of interior lines as features, rather than merely boundaries between textures.
1.3.2 Color Me Noisy: Example-based Rendering of Hand-colored Animations with Temporal Noise Control

*Color Me Noisy* is a video stylisation approach based on example-based texture synthesis. The approach was borne out of discussions with artists who work with traditional animations and felt that the previous approaches [Bénard et al., 2013], with their focus on maximizing temporal coherence of the result, lost a significant artistic quality bestowed by temporal noise which is usually an artefact of hand-made animation. The approach is therefore based on stylising a video sequence to match the visual style of the example while ensuring that a controllable amount of temporal noise is present without becoming disturbing to the viewer.

This principally improves on the previous state of art by enriching the stylisation with an additional channel of artistic expression, as the amount of noise may be controlled both spatially and temporally to convey emotion or other artistic intent.

1.3.3 Brushables: Example-based Edge-aware Directional Texture Painting

*Brushables* is an example-based painting approach which focuses on textures that could not be handled by previous approaches – namely textures with anisotropic directionality and fuzzy or gradual edge effects. Compared to *Painting by Feature*, Brushables handle texture edges and interiors simultaneously. This is seemingly a step back towards *Painting with Texture* but manually selecting the boundary becomes impractical for these more difficult textures.

The primary focus is therefore a significantly improved synthesis approach which handles fuzzy boundaries and incorporates an adapted shape descriptor into the synthesis, permitting the handling of edges in a soft, continuous way, as well as permitting the handling of significantly broader edges on which a simple mask-patch approach fails.

1.4 Thesis Structure

The rest of this thesis is structured around an anthology of publications co-authored by the author of this thesis, which were all published in high-impact peer-reviewed journals and contributed to the state of art in this area.

Chapter 2 summarises the previous research in areas related to the following chapters.

Chapter 3 presents the full text of *Painting by Feature*, published in the proceedings of ACM SIGGRAPH 2013 conference [Lukáč et al., 2013].

Chapter 4 contains the text of *Color Me Noisy*, as published in the proceedings of the Eurographics Symposium on Rendering 2014 [Fišer et al., 2014].

Chapter 5 presents *Brushables*, as published at the Pacific Graphics 2015 conference [Lukáč et al., 2015].
Chapter 6 summarises the contributions of previous chapters and discusses them in the context of recent developments in the state of art.

The appendices include the complete list of the author’s publications, statement of contribution and supplemental materials for the presented papers.
2 Related Work

In this chapter, we present the state of the art in the areas related to the papers included in this thesis. The overview is organized into three sections, each recounting the related work most pertinent to one of the papers included in this thesis.

2.1 Painting by Feature

As the groundwork in texture synthesis was laid down [Paget and Longstaff, 1998; Efros and Leung, 1999; Kwatra et al., 2005], follow-up research started to focus on novel applications. One of these was example-based stylization, pioneered by Image Analogis [Hertzmann et al., 2001], which introduced the idea of using external guidance channels to drive image synthesis. This idea has since been incorporated into other applications, such as image inpainting, usually also by the expedient of including “guidance distance” into the distance term used either to find compatible fragments, or to evaluate the objective function in optimization-based approaches.

One of the use-cases suggested in this paper was “texture-by-numbers”, wherein a segmentation map was used for the guidance channel. An input image was segmented and a novel segmentation was created describing the desired shape of the output image, and the analogies algorithm was then used to synthesise it. While this suggested the possibility of interactive painting scenarios, the computational expense of the guided synthesis made for a firmly offline algorithm and precluded interactivity.

The results of this algorithm also frequently suffer from local repetition artefacts where the guidance channels overconstrain the synthesis and the space of usable fragments becomes too sparse. Notably absent are any long-range curvilinear structures, as the algorithm does not have any explicit concept thereof. Since the synthesis is also pyramid-based, these thin features disappear from the guidance layers on the higher levels of the pyramid, further obscuring them. Because the overconstraining frequently occurs at texture segmentation boundaries, and because boundaries are known to be salient to the viewer, such an adaptation of a 2D texture synthesis algorithm thus tends to concentrate artefacts particularly in parts of the image that invite the viewer’s attention.

The first approach to directly focus on interactive texture painting was Synthesizing Natural Textures [Ashikhmin, 2001], which made interactivity possible by significantly accelerating the search step of the synthesis. Instead of performing a full search for each synthesized pixel, the algorithm would first attempt to copy a larger texture fragment based on the computed correspondences of the previously synthesized pixels. In addition, the user is permitted to guide the synthesis by effectively manipulating its initialization, which introduces a degree of interactivity.

Unfortunately, while this new synthesis approach is significantly faster than the previous methods [Wei and Levoy, 2000], by copying over larger coherent chunks of texture, artefacts may appear at the boundaries of such chunks. This is not a problem with stationary or “flat” textures, but becomes immediately apparent when the source texture contains even mild global shading, making it unsuitable for image synthesis. Furthermore, the interactive initialization effectively only “suggests” the low-frequency colour of the desired output, rather than constraining the texture to be used in the particular area of the output. This renders the approach usable for texture synthesis, where there no room for ambiguity in texture, but less useful for
CHAPTER 2. RELATED WORK

synthesis of images composed of multiple textures.

In *Painting With Texture* [Ritter et al., 2006], the authors attempted to improve the handling of boundary regions by introducing the concept of *neighbourhood shape*: a binary mask signifying which pixels within the patch belong to the same segment as the centre pixel. In synthesis, only pixels with identical neighbourhood shapes were considered as candidates. This significantly improves the synthesis of boundaries, since appropriate fragments are usually mapped to the salient boundary regions. Yet, artefacts still appear if the shape of the boundary does not match in a pixel-perfect fashion, as the neighbourhood shape constraint is hard and does not explicitly consider local continuity, which again causes overconstraining. There is also no concept of lengthwise continuity, so e.g. stitch patterns cannot be synthesised by this method. Finally, the method suffers from the discontinuity artefacts of the underlying synthesis algorithm [Ashikhmin, 2001].

Previous research suggests that explicit handling of salient boundaries can significantly improve result quality [Criminisi et al., 2004; DeCarlo et al., 2003; Fang and Hart, 2006], but the user constraints available in existing content-aware manipulation and synthesis methods [Kwatra et al., 2005; Lefebvre and Hoppe, 2005; Barnes et al., 2009; Shechtman et al., 2010; Darabi et al., 2012; Yücer et al., 2012] do not provide this form of control. Therefore, we aim to introduce an explicit boundary synthesis algorithm to be used as a separate tool alongside and in conjunction with a traditional texture synthesis tool.

McCann and Pollard [2008] introduced a set of gradient-painting tools, one of which was the *edge brush*: a tool that permits the user to pick an edge in an exemplar and then map it onto a path in an existing image using gradient-domain blending. However, this tool is intended for simple image editing rather than *de novo* painting and simply loops the input line as necessary. This causes discontinuity artefacts where the repetition happens, and the line, if it is any longer than the input line, is visibly periodic as there is no variety synthesis.

In *Video Textures* [Schödl et al., 2000], the authors use a synthesis process to extend a simple video, which is fundamentally a similar problem; the video is a 1D structure in the time domain and the aim is to minimise discontinuity in the cuts. To do this, they detect points at which a low-discontinuity transition is possible, but still do not synthesise variety, causing the output to be periodic.

Sun et al. [2005] demonstrated the benefit of explicitly synthesising boundaries, before synthesising interior texture, in an inpainting scenario. In their approach, this is achieved by imposing curvilinear constrains where only patches from the corresponding interface in the source area are eligible as source patches, and running dynamic programming to perform the actual assignment while minimising discontinuity cost. While this approach works well in an inpainting scenario with strict boundary conditions, in free-form painting it also produces periodicity. The underlying cause is the strict optimization for discontinuity only; it can be shown that for an output line significantly longer than the input line, the optimal solution consists simply of repeating the cheapest loop in the graph, which visually manifests as periodicity.

In our approach, we aim to preserve the benefits of using separate boundary synthesis, further enforcing them by having the synthesis of the less salient interior follow the synthesis of the more salient boundary. Furthermore, we introduce variety requirements which are to ensure that periodicity will be broken and all parts of the input will be used at some point, preserving the visual richness of the boundary structures of the input.

For area synthesis, we rely on optimization-based synthesis [Wexler et al., 2007] accelerated by fast approximate nearest-neighbour search [Barnes et al., 2009], which we expect to mitigate the artefacts exhibited by the synthesis core used by Ritter et al. [2006].
2.2 Color Me Noisy

Video stylisation is a difficult sub-task of non-photorealistic rendering, as adding the temporal dimension raises temporal coherence concerns which are non-trivial to address. In general, temporal incoherence caused by noise in the synthesising method is treated as an undesirable artefact and the various methods attempt to minimise it [Bousseau et al., 2007; Bénard et al., 2009; Bénard et al., 2011]. This treatment is a priori reasonable, as uncontrolled noise, like that introduced by frame-by-frame synthesis, can be unpleasant to the viewer and significantly affect viewing comfort.

On the other hand, in hand-drawn animations as well as other styles, temporal noise originally introduced as a result of imperfections in the animation process can be exploited by the artists as an additional medium of expression to either set a desired mood or allude to a particular style. Therefore it may in certain cases be desirable to deliberately introduce noise into the result in a controlled fashion.

Approaches for procedural noise generation [Bénard et al., 2010; Kass and Pesare, 2011], which generate temporal noise in a coherent manner by suppressing certain temporal components, could be used to introduce noise into the synthesised results. However, the noise thus introduced is not consistent with the noise created naturally as a result of physical animation technique.

Some example-based synthesis approaches have the ability to enforce variety [Lefebvre and Hoppe, 2005, 2006; Risser et al., 2010], which could in theory be used to introduce noise consistent with the example, but due to being formulated on the index domain, they lack the ability to decouple detail across scales. This in turn renders them unable to control the amount of noise in the result. While methods for multi-scale texture synthesis which could theoretically be combined with these approaches to control the scale exist [Han et al., 2008; Vanhoey et al., 2013], we would first have to decompose the input into constituent scale levels, which is non-trivial.

Like most contemporary example-based video stylisation approaches [Bénard et al., 2013], ours is an extension of the state of art image synthesis methods [Wexler et al., 2007; Simakov et al., 2008]. Although these can emulate the appearance of exemplar media satisfactorily, they cannot capture the temporal characteristics of hand-drawn animation. We can, however, take advantage of some of the principles used in example-based image morphing techniques [Darabi et al., 2012; Shechtman et al., 2010] in order to introduce randomness in a controlled way, the exact mechanism of which is described in the paper.

2.3 Brushables

As another example-based painting approach, Brushables [Lukáč et al., 2015], like Painting by Feature [Lukáč et al., 2013], are descended from the lineage of Image Analogies [Hertzmann et al., 2001], Synthesizing Natural Textures [Ashikhmin, 2001] and Painting with Texture [Ritter et al., 2006].

While combining a stroke synthesis approach with a texture synthesis approach helped improve the quality on the crucial salient regions on the boundary of the texture, the users were not satisfied with the workflow which necessitated explicitly annotating lines and boundaries and limited edge effects to a fixed width. The hybrid approach also causes discontinuities at the interfaces of areas created by stroke synthesis and those created by area synthesis. Further-
more, the users requested a tool that would be able to produce strokes of varying width, while also being usable for filling areas in a directed fashion.

Consequent research produced more flexible optimization-based stroke synthesis approaches, such as RealBrush [Lu et al., 2013], which formulated more involved optimization functions to replicate individual strokes. This and other similar approaches [Lu et al., 2014; Zhou et al., 2013] are capable of producing consistent strokes faithful in appearance to the exemplar, but cannot vary the stroke width and are strictly limited to 1D structure, making them unusable for area filling.

Thus, a texture synthesis approach that would account both for directionality inherent in stroke-based media, as well as boundary effects on the interfaces between textures, is called for. The former requirement implies that texture directionality has to be detected in the input, authored for the output, and considered during synthesis.

Some previous approaches would consider texture orientation in order to compensate for transformations in images [Lefebvre and Hoppe, 2006; Eisenacher et al., 2008] or to allow for user specification of direction of the output [Zhang et al., 2003; Diamanti et al., 2015], but this step in itself is relatively straightforward when the input and desired output directions are known.

Detecting directionality in textures and images has been explored in the context of various stylisation techniques [Hays and Essa, 2004; Kang et al., 2009; Kyprianidis, 2011], where direction was considered as a guide for how an artist would place brush strokes. In our scenario we are faced with a similar task, trying to determine the underlying direction of the texture in order to align it with output brush strokes.

To do this, we may employ some of the common techniques to detect orientation, such as structure tensors [Brox et al., 2006] or edge tangent flow [Kang et al., 2007]. Both of these, however, suffer from an important drawback; they detect orientation in textures modulo 180, or in other words, up to a sign. This means that the sign is either omitted or inconsistent in the final solution, which may break synthesis for textures which contain longitudinal structures asymmetric with respect to the direction sign. Because this is the case in many practical examples (e.g. grass, hair, some types of fabric), we must address it in our scenario. Although there are techniques that attempt to find consistent direction [Kang et al., 2009; Xu et al., 2009], they typically fail to do so over larger distances or when the input field contains singularities.

Authoring of vector fields has been extensively studied on 3D surfaces [Zhang et al., 2003; Fisher et al., 2007; Crane et al., 2010; Maharik et al., 2011], where the task was to compute a smoothly varying global field based on a small set of sparse user constraints. In an incremental painting scenario, however, this globality becomes undesirable, as artists expect any additional brush stroke to only have localised effect. We must therefore explore ways of creating such vector fields as well as incrementally modifying them in a localised fashion.

To be able to consider and synthesize edge effects, we rely on shape descriptors, commonly employed in computer vision [Berg and Malik, 2001; Belongie et al., 2002]. These are capable of establishing the position of any given point with respect to both important global and local shape features. As they must examine a large portion of the image to do so, and as texture synthesis requires numerous such evaluations in short order, they become a performance bottleneck which we must address.

Like many state-of-art example-based approaches [Darabi et al., 2012; Fišer et al., 2014; Kaspar et al., 2015], we take advantage of optimisation-based synthesis [Kwatra et al., 2005; Wexler et al., 2007]. Together with fast approximate nearest-neighbour field search algo-
rithms [Barnes et al., 2009], these can produce high-quality results at interactive rates. Simultaneously, they provide a flexibility of formulation into which we can integrate both shape- and edge-awareness in order to properly address the requirements of our scenario.
Abstract

In this paper we propose a reinterpretation of the brush and the fill tools for digital image painting. The core idea is to provide an intuitive approach that allows users to paint in the visual style of arbitrary example images. Rather than a static library of colors, brushes, or fill patterns, we offer users entire images as their palette, from which they can select arbitrary contours or textures as their brush or fill tool in their own creations. Compared to previous example-based techniques related to the painting-by-numbers paradigm we propose a new strategy where users can generate salient texture boundaries by our randomized graph-traversal algorithm and apply a content-aware fill to transfer textures into the delimited regions. This workflow allows users of our system to intuitively create visually appealing images that better preserve the visual richness and fluidity of arbitrary example images. We demonstrate the potential of our approach in various applications including interactive image creation, editing and vector image stylization.


Keywords: example-based painting, stroke synthesis, painting-by-numbers, vector image stylization, non-photorealistic rendering

Links: DL PDF WEB

1 Introduction

Strokes and lines are the most elementary primitives in painting, both digital and physical. The concept of drawing shapes by first sketching and developing object outlines seems to be so natural and intuitive that small children employ it just as artists and designers. Any existing image editor implements the basic pencil and/or brush tools, and various attempts have been made to enhance their expressive power, such as the calligraphic brush or the textured stroke. Similarly, vector-based image editors use paths as their most fundamental primitive for defining object boundaries.

Despite their importance for sketching the essential structures in an image, basic brush- or path-based tools are generally less suitable for creating a clean, richly textured image such as the ones shown in Figure 1. Researchers have long been aware of this gap between a sketch and production quality artwork, and proposed various ideas for converting simple sketches into richer and more expressive images [Ashikhmin 2001; Hertzmann et al. 2001; Ritter et al. 2006; Orzan et al. 2008].

Unfortunately, existing approaches often face difficulties when synthesizing images with significant structure, as the underlying algorithms generally focus on synthesizing 2D textured areas, without explicitly enforcing consistency to the boundaries of a shape. Due to the sensitivity of human vision to the contours of a shape [De-

Figure 1: Representative results generated by our proposed example-based painting framework. The user selects line features in a reference image (colored lines in the top left images, see also area features in the supplementary material) which are then immediately available as brushes for applications such as real-time painting or vector image stylization. The respective top right images depict the user's painted strokes in order to create the images in the bottom row. These demonstrate various use cases of our method: (a) complex paintings from a few input strokes, (b) painting detailed, structured boundaries, (c) watercolor, and (d) diffusion curve effects. Source credits: (a) Sarah G via flickr, fzap via OpenClipArt; (b) Pavla Sýkorová, clipartsy; (c) bitbox via flickr, papapishu via OpenClipArt; (d) Anifilm, Pavla Sýkorová
CHAPTER 3. PAINTING BY FEATURE

Inherently 2D, i.e., it does not preserve 1D structure of more complex boundaries. However, a key limitation is the lack of an additional energy term which takes similarity of source and target areas into account. Hence, the method produces convincing results only for textures which have a nearly constant cross-section profile along the boundary, producing artifacts otherwise (see Figures 2c and 9).

Our main technical contribution is a novel algorithm for interactive synthesis of line features (brush tool) which utilizes a randomized graph traversal mechanism with multi-level blending to seamlessly synthesize long, non-repetitive, textured strokes sampled from shorter exemplars located in the input image. For the transfer of area features ($f$ tool) we use a state-of-the-art texture synthesis algorithm [Wexler et al. 2007] which avoids visible discontinuities between painted line features and textured areas while preserving the richness of the original exemplar. Both tools provide immediate real-time feedback, making their use as intuitive and easy as an ordinary brush or fill tool. Creating complex, visually appealing drawings with our system requires similar effort as producing convincing results only for textures which have a nearly constant cross-section profile along the boundary, producing artifacts otherwise (see Figures 2c and 9).

Similar considerations apply to other types of texture synthesis algorithms [Ashikhmin 2001; Efros and Freeman 2001; Kwatra et al. 2003] which partially also provide support for user constraints [Kwatra et al. 2005; Lefebvre and Hoppe 2005], or to matching based image manipulation and morphing techniques [Barnes et al. 2009; Shechtman et al. 2010; Darabi et al. 2012; Yücer et al. 2012]. All these methods provide very flexible and powerful tools for filling or transforming image areas with plausible and visually rich textures, but at the same time they are inherently 2D without support for user-controlled, real-time 1D structure transfer from a reference image. See Figure 2d for an exemplary result with the method of Ashikhmin et al. [2001]. The synthesis step of the above result took 130 seconds, whereas our approach provides instantaneous feedback.

Recently, other example-based content generation techniques have been proposed, which create new images from a user-provided set of examples [Risser et al. 2010; Assa and Cohen-Or 2012]. However, these techniques are non-interactive, global approaches which specialize in rapid generation of a large number of variations of the input image. Currently, the only way to influence this process is by providing a different choice of input images.

Sun et al. [2005] demonstrated the benefit of giving the user control over structural features in the context of image inpainting. They apply a constrained patch-based synthesis on the user-provided line features and then perform inpainting on the remaining areas that is consistent with the previously synthesized structures. A restriction of this approach is, however, that the employed energy minimization provides no guarantees that the global scale visual appearance of the synthesized line feature is consistent with its appearance in the respective source image. In texture synthesis, this problem is generally avoided by multi-scale synthesis, but this is not feasible for linear features, as they eventually disappear on lower resolutions. Using basic energy optimization without a sufficiently expressive model of a feature’s global scale, artifacts are often perceptible as a periodic repetition of a pattern along the output path. On a related note, another example for the benefits of contour-based editing is the work of Fang et al. [2007] for detail preserving shape deformation in images.

For vector graphics editing Orzan et al. [2008] presented a technique for creating smooth color transitions between spline paths using Poisson interpolation. Due to the purely vector-based representation this approach is not suitable for style and texture transfer

Figure 2: Comparison of results from different approaches given the input picture (a) as the reference image or feature palette. Result of (b) Image Analogies [Hertzmann et al. 2001], (c) Painting with Texture [Ritter et al. 2006], (d) Synthesizing Natural Textures [Ashikhmin 2001], (e) our approach. Source credit: Wednesday Elf – Mountainside Crochet via flickr

Carlo et al. [2003], such artifacts become immediately apparent (see comparison in Figure 2).

This paper addresses these issues by modeling an image as a set of two classes of features. The first class corresponds to 1D line features, such as important contours, boundaries of textured regions, or salient strokes which are used to define the basic structure of the image. The second class corresponds to 2D area features, which represent regions filled with a nearly-stationary texture. For defining the visual style of an image, we introduce the metaphor of a feature palette, which is simply one or more example images of a desired visual style, in which the user selects line and area features with which to paint.

Our main technical contribution is a novel algorithm for interactive synthesis of line features (brush tool) which utilizes a randomized graph traversal mechanism with multi-level blending to seamlessly synthesize long, non-repetitive, textured strokes sampled from shorter exemplars located in the input image. For the transfer of area features ($f$ tool) we use a state-of-the-art texture synthesis algorithm [Wexler et al. 2007] which avoids visible discontinuities between painted line features and textured areas while preserving the richness of the original exemplar. Both tools provide immediate real-time feedback, making their use as intuitive and easy as an ordinary brush or fill tool. Creating complex, visually appealing drawings with our system requires similar effort as creating a simple contour sketch in standard drawing systems.

2 Related Work

One of the first works for example-based visual style transfer between images is the Image Analogies approach by Hertzmann et al. [2001]. They discuss the possibility of example-based painting using the texture-by-numbers paradigm where an input image is first segmented into multiple regions denoted by color labels, and then these labels are painted to form a new segmentation from which an output image is generated using their texture synthesis algorithm. While this approach provides a high degree of freedom in defining the output result, it is not clear how to support the concept of 1D structure elements such as contours. Moreover, the algorithm complexity prohibits an interactive implementation. A representative result is shown in Figure 2b.

Ritter et al. [2006] further extended Hertzmann et al.’s framework and created a nearly interactive texture-by-numbers painting program where boundary pixels are refined automatically thanks to an additional energy term which takes similarity of source and target boundaries into account. However, a key limitation is the lack of user control in the boundary forming process and the technique is inherently 2D, i.e., it does not preserve 1D structure of more complex boundaries. Although pixels are transferred from locations with a similar boundary shape, there is no guarantee that they will produce a 1D, visually continuous strip since the source pixels can be located on different parts of the boundary.

Recent example-based content generation techniques have been proposed, which create new images from a user-provided set of examples [Risser et al. 2010; Assa and Cohen-Or 2012]. However, these techniques are non-interactive, global approaches which specialize in rapid generation of a large number of variations of the input image. Currently, the only way to influence this process is by providing a different choice of input images.
between images. McCann and Pollard [2008] broke new ground by introducing a set of gradient-painting tools, designed to be fully interactive and directly controlled by the user. Notably, they introduce an edge brush tool which allows the user to select a path in a source image and map it to a path in the result using gradient-domain blending. Their approach, however, targets image editing, and their simple copy-paste procedure offers no variation during feature synthesis, resulting in clearly visible periodicity when the output path is much longer than the source path of the respective feature. In our approach, we utilize a workflow similar to theirs for image creation. However, we introduce a generative line feature model to enable an indefinite extension of a source path without such artifacts.

Related to our algorithm for feature transfer is the work on video textures [Schödl et al. 2000]. They developed a feature model capable of extending a video in the temporal domain, where the frames are represented as graph nodes and the edge weights represent a measure of similarity between two frames. Thanks to this representation a permutation of video frames can be expressed as a low-cost traversal through the graph. Their approach served as an inspiration for our generative model for line feature synthesis. However, similar to the work of Sun et al. [2005] and McCann and Pollard [2008], a direct application of their loop-based synthesis algorithm would result in obvious periodic artifacts. Part of our contribution is a synthesis algorithm that resolves these issues.

3 Our Approach

As briefly outlined in the introduction, our proposed approach is based on three central concepts. The first two of them are the two different types of features and their corresponding tools:

- A **Line Feature** is a one-dimensional feature representing an arbitrary curvilinear structure, such as an edge or contour in an image. It typically represents a boundary between two textured regions, but can also represent other structures such as open curves. The corresponding tool for painting line features is the **Brush** tool.

- An **Area Feature** is a two-dimensional image region which has the semantics of a stationary texture rather than that of a one-dimensional structural element. It typically represents the interior of an area, but can also be a changing gradient or any other area sample. Its corresponding tool is the **Fill** tool.

Both tools consist of two parts, namely a **selection** component which allows the user to define a desired line or area feature, and a **synthesis** component which efficiently renders the corresponding output according to the user’s drawing.

The third central concept is the **Feature Palette** and it concerns the feature selection process. Rather than requiring the user to define features in a cumbersome manual way, the basic idea is to regard an arbitrary set of input images as a palette for painting. The user may simply pick one or more input images that reflect a desired visual style, and our algorithm provides the selection tools to intuitively and efficiently define line features as well as area features. Hence, any image can be used as a palette for defining features.

These concepts are fundamentally different from merely building a static database of strokes and fill textures, as commonly done in vector image editors. In our process, the reference image(s) used as the feature palette permit effortless definition of a dynamically changing library of brushes and textures on-the-fly. This facilitates the replication of the desired visual characteristics of the reference images in one’s own creation. The user directly benefits from the rich visual details that are typically present in paintings, drawings, or photographs. Just as a painter can efficiently mix colors on a physical color palette, our concept allows the user to intuitively and interactively modify and refine a feature with instant feedback while painting.

In the following section we describe how the respective selection and synthesis components of both the brush and the fill tools for line and area features are implemented.

3.1 Brush

Given a feature palette in the form of one or more input images, selecting a line feature such as an object contour requires the user simply draw a path (the width of which can be manually adjusted) approximately along the desired feature. Since precise drawing of such a path would be tedious, our algorithm supports an assisted selection that refines the user’s approximate path and aligns the selection closely to the actual line feature in the image. We found that a relatively simple gradient-based approach is reasonable in order to provide an active support for the user at a sufficient accuracy for our algorithm; hence we based our path selection on an Active Contours approach [Kass et al. 1988]. A considerable advantage of this approach is that it runs in real-time and gives an instant result, which is an important requirement for a responsive and intuitive user interface.

Once the user has defined a path over a line feature we require a real-time algorithm that synthesizes a corresponding line feature in the output image as the user paints. In the field of texture synthesis it has long been understood that synthesizing a larger texture simply by tiling a smaller example texture produces sub-optimal results. Thus, in order to avoid periodicity, some texture synthesis techniques [Lefebvre and Hoppe 2005] deliberately introduce a degree of randomness instead of tiling the texture. Likewise, our goal is to reproduce the local visual characteristics, i.e., the look and feel of a line feature, without introducing noticeable artifacts on a larger scale. Approaches such as [McCann and Pollard 2008] exhibit periodicity and cannot explicitly avoid visible discontinuities when stroke endpoints meet. We present a new algorithm for randomized line feature synthesis based on a graph model of the input feature to resolve such issues.

As line features such as object contours are one-dimensional and oriented, we found the graph formalism introduced by Schödl et al. [2000] for manipulating video over time to be an excellent basis for feature synthesis. We sample an input path at equidistant points and consider the direction of the feature to be equal to the direction of the user’s stroke. Treating these samples as graph nodes and using the direction of the feature for ordering, we define a complete oriented graph, where the weight w(i, j) of an oriented edge between nodes i and j is given by a dissimilarity measure. Specifically, we define w(i, j) = SSD(p(i), p(j)), where p(i) is a square image patch centered on the i-th sample and aligned with the path direction, and SSD denotes the sum of squared differences between patches (see Figure 4). We use SSD(p(i), p(j − 1)) rather than SSD(p(i), p(j)) because traversing to a consecutive sample on the original feature should be free, and thus w(i, i + 1) should be equal to zero. The size of the patch is a user-configurable parameter which is intuitively equivalent to brush width and can be adjusted interactively. A walk in such a graph represents a permutation (with repetition) of input samples, which, if transferred to equidistant samples on a different path and rendered, would yield a variation of the source feature. The total cost of this walk is then representative for the amount of discontinuities in the output.

Given this representation the main concern is how exactly to generate a walk through this graph to satisfy all of our requirements and
Figure 3: A comparison of different walk synthesis approaches. Top to bottom: looping, dynamic programming and our randomized graph traversal. Note that finding the cheapest walk of a given length by dynamic programming provides the optimal result with respect to discontinuity cost, but it does so by finding the cheapest loop in the graph and thus introduces periodicity. A randomized approach, though not optimal with respect to the global cost, provides a more natural, varied look without noticeable visual discontinuities.

Figure 4: A feature graph in matrix form, with color coded weights of the similarity between two patches $p(i)$ and $p(j)$. Blue represents a low matching error and hence a high similarity, while red represents a low patch similarity. Note that the matrix is not square, as there can be no edges into node 0, nor is it symmetric, as $w(i, j) = SSD(i, j) − 1$ rather than $SSD(i, j)$.

Figure 5: An illustration of blending on jumps. (a) A colorized indication of the jump strip and blending area. (b) Synthesized feature without blending. (c) Synthesized base layer without blending. (d) Synthesized detail layer. (e) Synthesized base layer with extrapolation-blending. (f) Synthesized base layer with extrapolation blending and added detail.

element present in the input will be rendered from time to time, without enforcing any particular ordering and keeping the visible discontinuities to a reasonable minimum. One could also conceivably bias the walk to a certain sub-portion of the feature by a more sophisticated selection of goal nodes, although we have not found this necessary for our application.

In order to render the selected feature onto a user-provided path, we sample the output path at equidistant intervals, generate a walk and assign to each of the output samples an input sample represented by the node at the given position in the walk. Having thus established correspondences between output and input samples, we use a simple piecewise-rigid mapping based on the Voronoi diagram of the output samples to determine the output pixel values for pixels within the stroke width of the sketched path. The process is illustrated in Figure 6.

Discontinuities in the synthesized path may occur when the edge with greater cost has to be traversed. To mask these without sacrificing fine details, we employ a decomposition-blending approach inspired by Burt and Adelson [1983]. Whenever consecutive output samples are created by a jump between non-consecutive input samples we perform local blending. To that end, we use a bilateral filter to decompose the source image into a base layer and a detail layer, as proposed by Durand and Dorsey [2002]. We then extrapolate the base layer values for each of the consecutive sub-sequences around the jump point and blend them, re-applying detail immediately thereafter, as illustrated in Figure 5.

3.2 Fill

The second tool, which we provide for efficient filling of image areas between line features, is essentially a paint bucket tool as present in all common image editors. However, analogous to the brush tool, our concept is to provide a fill tool that fills image areas with texture selected by the user from the image serving as the feature palette, maintaining consistency with the existing line features. Selection of area features is more straightforward than for line features as no specific structural properties have to be observed.
during the selection. Hence, in our implementation the user can simply specify any arbitrary region in an image and use it as an area feature.

Unlike a simple flood fill tool, we have to consider the boundary conditions of the region being filled to avoid inconsistencies with existing image content like line features. Thus, rather than formulating the task of the fill tool as a simple texture synthesis problem, we treat this step as a content-aware fill which respects boundaries of the filled area and implement the method of Wexler et al. [2007] in combination with PatchMatch [Barnes et al. 2009] for fast nearest-neighbor search.

A multi-scale optimization approach [Wexler et al. 2007] is critical for our purpose, since the areas to be filled span over the majority of the canvas and treating the fill synthesis locally would lead to undesired artifacts and would furthermore be prone to introducing unwanted repetitions in the generated texture. To improve the quality and visual appearance of the result, we also perform the nearest-neighbor search across a limited range of rotations (see Figure 7). However, rather than computing the transformed source patches on the fly, we found that the combination of pre-rotating the source selection and performing the nearest-neighbor search using only translations to be significantly faster, which is crucial for instant results and direct visual feedback to the user.

4 Applications and Results

An overview of our image creation workflow is illustrated in Figure 8. Due to its generality, our approach can be utilized in several applications. One of our primary applications is vector image stylization: the user selects line and area features in an example image and then assigns them to paths and fill shapes of a vector image, respectively. Figure 1 shows representative results created using our framework. Note that, unlike previous texture-by-numbers approaches, we can handle open paths and strokes. The result images are visually consistent on a local as well as a global scale and represent the visual style of the respective reference image (see comparison with previous texture-by-numbers approaches in Figures 2 and 9). Figure 1a illustrates that even a simple vector image composed of a very limited number of input strokes (three in this example) can lead to richly textured image. In Figure 1b please note the quality of the knitting stitches generated by our line feature synthesis at the boundaries of the different pieces of the penguin. Our approach can also be applied for watercolor painting, as shown in Figure 1c. This challenging task usually requires sophisticated techniques [Curtis et al. 1997; DiVerdi et al. 2013], whereas our approach can solve it without additional specific tools.

An interesting characteristic of our approach is that when paths incident to a region have different “inside colors”, the region inpainting algorithm attempts to diffuse the difference between their colors over the intermediate region, producing results similar to Diffusion Curves [Orzan et al. 2008], with no additional creative effort on the artist’s part. For example, in Figure 1d, note the diffusion effect along the cat’s whiskers and mustache (see also a more complex example of texture transitions in Figure 15). A comparison
with Diffusion Curves is available in Figure 10. Both approaches took a comparable amount of artistic effort to produce, however, our method enables the transfer of the visual style and richness in terms of texture from a reference image (see Figure 1d). Additional examples of different stylizations given a single user-drawn sketch are shown in Figure 11. In this stylization scenario, the user simply needs to select the line and area features they would like to incorporate in the result image.

Another exciting application is interactive example-based painting. We have developed a painting program which implements just the two tools we introduce in this paper, deliberately leaving out extra functionality of sophisticated image editors, in order to show that our painting-by-feature approach alone enables the creation of appealing results. In our paint program the user may select features from source images and transfer them to manually indicated positions, using the same mode of interaction as with the common brush tool and fill tool known from consumer image editors. A representative interaction with our application is shown in Figure 8 as well as in the supplementary video. It demonstrates that our application is simple to use, and that the user can create and edit paintings interactively with instantaneous feedback. Visually appealing results can be created in a short time, typical editing session for the results shown here were in the order of 1–3 minutes depending on the level of detail the user wishes to incorporate. Additional results are shown in Figure 12. Further potential applications of our method include image editing scenarios such as inpainting. We refer readers to the supplementary material for a representative result.

4.1 Limitations

While our approach has proven suitable for its intended applications and produces high quality results, some limitations do apply.

We do not explicitly handle possible intersections and junctions of line features which may produce visually disturbing transitions in the output image (see Figure 13). These artifacts can partially be alleviated by proper reordering of strokes or using some sort of blending, e.g., min/max-blending (GL_MIN or GL_MAX blending mode in OpenGL) or decomposition-blending described in Section 3.1. Nevertheless, in future work one may consider to incorporate support for intersections and junctions directly into the synthesis algorithm to automatically produce seamless output.

We also deliberately do not check for consistency of the selected features to give the user full control and artistic freedom. As a consequence the user can select a line feature that is not aligned...
with an actual linear structure in the input image or one that is composed of incompatible structural elements. In these cases our algorithm might produce visually displeasing transitions (see Figure 14). Similarly, selection of an area feature which is incompatible with already drawn line features may also lead to an erroneous result (see Figure 15). An alternative scenario to investigate in future work is that the feature selection process could be assisted by interactive image segmentation tools [Li et al. 2004], or by identification and removal of inconsistent sub-elements, e.g., by texture analysis [Todorovic and Ahuja 2009].

To prevent periodicity, our approach runs a randomized graph walk (see Section 3.1). The disadvantage is that variations might exist between results for a same source image and input sketch. Additional results available in the supplementary video (elephant sequence) show that these variations are very limited, on a local level and visually consistent with the other results on a global level, which is sufficient for our target applications.

5 Conclusion and Future Work

We have presented a feature-based image creation model, useful for vector image stylization as well as manual image creation and image editing. Our flexible example-based stylization approach blurs the traditional border between the vector- and pixel-worlds, allowing us to create and manipulate images while preserving the visual richness of a chosen artistic style. We eagerly anticipate the new possibilities in artwork creation that this approach opens to artists, and are curious about the results which may be achieved by combining this simple, yet powerful basic approach with other existing creation and editing tools.

An interesting direction for future work is the automation of the entire process of vector image stylization. This could be achieved by automatically detecting features in a source image and assigning them to paths and regions of a vector image based, e.g., on similarity of fill and stroke colors to the colors in the feature.

We could also modify our algorithm to automatically synthesize the fill for areas between user-defined curves while they are being drawn, producing an example-based variant of the Diffusion Curves by [Orzan et al. 2008]. However, even though line feature synthesis is fast enough for interactive editing, this would require a real-time fill synthesis algorithm (even with PatchMatch, [Wexler et al. 2007] is too slow to permit this) and a similarly rapid image analysis tool, which would determine source areas for output regions based on the input strokes and other features present in the input in order to keep the output visually consistent.

Similarly, used in conjunction with an automated image decomposition algorithm such as [Guo et al. 2007], one could reduce an input image into a sketch representation and a representative subset of features in order to re-synthesize the original image at a later time. Thus one could facilitate image compression with a configurable loss of information (see supplementary material for an example of image decomposition based on our method).

For the brush tool, it might be possible to investigate whether the input line feature contains any underlying dimensionality (such as texture orientation), and modify our formulation so that the output is constrained by this underlying parameter, determined, e.g., by pen pressure. Similarly, the introduction of control maps for area features could play a role for synthesis.

Acknowledgements

We would like to thank Gioacchino Noris and all anonymous reviewers for their constructive comments. This research has been supported by the Technology Agency of the Czech Republic under the research program TE01020415 (V3C – Visual Computing Competence Center) and partially by the Grant Agency of the Czech Technical University in Prague, grant No. SGS13/214/OHK3/3T/13 (Research of Progressive Computer Graphics Methods).
References


Figure 15: An example output of our fill tool when synthesized strokes contain incompatible structures (a). When the whole source image (b) is taken as an example (red and blue rectangles) the fill tool produces pleasing transitions (c). However, when an incompatible portion (red and blue areas) of the source image is selected (d), the algorithm can produce erroneous results (e). Source credit: Carl Wycoff via flickr.
Color Me Noisy: Example-based Rendering of Hand-colored Animations with Temporal Noise Control


1CTU in Prague, FEE, 2Brno University of Technology, 3George Mason University, 4Adobe Research

Abstract

We present an example-based approach to rendering hand-colored animations which delivers visual richness comparable to real artwork while enabling control over the amount of perceived temporal noise. This is important both for artistic purposes and viewing comfort, but is tedious or even intractable to achieve manually. We analyse typical features of real hand-colored animations and propose an algorithm that tries to mimic them using only static examples of drawing media. We apply the algorithm to various animations using different drawing media and compare the quality of synthetic results with real artwork. To verify our method perceptually, we conducted experiments confirming that our method delivers distinguishable noise levels and reduces eye strain. Finally, we demonstrate the capabilities of our method to mask imperfections such as shower-door artifacts.


1 Introduction

Hand-colored animation is a type of traditional animation, where each frame is created independently, from scratch. It has a distinct visual style represented by a certain amount of temporal flickering which arises due to misalignment of details in consecutive frames. This characteristic feature lends it a look of liveliness and emotional expressivity, which is being successfully leveraged by critically acclaimed artists such as Bill Plympton and Frédéric Back.
While temporal noise is usually understood as an undesirable artifact in NPR techniques [BNTS07, BBT09, BBT11, OH12], used judiciously it may serve as an additional medium of artistic expression, either to evoke a hand-crafted look (such as sketchbook scenes in Disney’s Piglet’s Big Movie), or to set a certain mood (e.g., Shadow World sequences in The Lord of the Rings). In Bill Plympton’s more recent work (e.g., Cheatin’) noisy, hand-drawn sequences are combined with coherent sequences to convey different moods.

However, the nature of the medium makes it difficult to control the amount of noise, and high noise levels can cause visual fatigue in the viewer. This, in conjunction with the amount of labor involved in production, creates a demand for a more automated process that lets artists control the amount of noise, and high noise levels can cause visual fatigue in the viewer. This, in conjunction with the amount of labor involved in production, creates a demand for a more automated process that lets artists control the amount of noise without eliminating it entirely.

Noris et al. [NSC11] recently presented a system which affords control over the amount of temporal flickering in a sequence of digitally drawn sketches. By registering individual strokes in selected keyframes, they reduce temporal jitter using a weighted combination of original noisy motion and smoothed inbetweening. Although this approach produces impressive reduction of temporal noise level for sketchy vector drawings, it still requires a hand-drawn animation as an input.

Our aim is to reach a more practical workflow that takes a temporally coherent animation created using existing CG pipelines and enriches it with temporal noise synthesized de novo from examples of an arbitrary drawing medium. A similar workflow was recently used by Bénard et al. [BCK13] in their framework, which extends Image Analogies [HJO01] to render impressive stylized animations with a specific style or drawing medium given by example. They focus on enforcing temporal coherency using a sophisticated system of correspondence propagation; however, the underlying re-synthesis technique does not permit control over the amount of temporal noise.

In this paper, we propose a novel example-based technique that not only preserve temporal coherency but also introduces a controllable amount of temporal flickering that conveys lively dynamics and visual richness which can be used either to evoke an impression of hand-colored look or provide an additional dimension of expressivity.

2. Related Work

Synthesizing various drawing media is one of the key challenges of non-photorealistic rendering. A wide spectrum of techniques spanning from computational approaches [CAS’97, HLFRO7, LHXJ12] to realistic example-based methods [ZZXZ09, LBDF13, LBFT13] has been developed. A key issue arises when these techniques are applied to animations, where frame-independent synthesis leads to unpleasant temporal noise that affects viewing comfort.

Many techniques have been proposed to alleviate this issue by enforcing temporal coherency [BNTS07, BBT09, BBT11, OH12]. Although these methods produce visually pleasing results, their visual structure is inconsistent with the natural look of noise typical for hand-colored animation. A similar limitation holds also for procedural noise generation [BLV10, KP11] which allows for temporally coherent stylization by suppressing temporal components of the generated noise.

Related to synthesis with temporal coherence are methods that try to enforce variety during synthesis [LH05, LH06, RHGD10]. They introduce the ability to vary randomness between scales, but due to being formulated in index domain, they cannot de-couple visual information across scales, which would be necessary for temporal noise control. Related multi-scale texturing approaches [VSLD13] may use a separate source for each scale, but decomposing an example image in this way is problematic.

A different approach to variety synthesis proposed by Assa and Cohen-Or [ACO12] does not rely on texture synthesis but instead decomposes the exemplar into layers, which are then recombined and the result is randomly warped. In our scenario we would like to conform to the user-defined shape, and the remaining small number of discrete varying outputs is insufficient to simulate the variety typical for hand-colored animation.

Our approach is inspired by image morphing techniques [SRAIS10, DSBB12] that extend state-of-the-art image synthesis algorithms [WSI07, SCSD08]. Although these methods have the potential to simulate the look-and-feel of hand-colored animation they do not address the control over the amount of temporal noise.

Figure 2: Motivation—hand-colored animations (a) look temporally coherent when low-pass filtered (b). However, at higher frequencies they contain details that reflect physical properties of the drawing medium and introduce temporal noise (c).
3. Our Approach

Noise found in hand-colored animations has a specific nature. Artists tend to preserve coherency at a global level—when the sequence is viewed at a distance (see Fig. 2a) or when a low-pass filter is applied (Fig. 2b) the animation is perceived to be temporally coherent. However, at a local level, temporal variance in high-frequency details becomes visible (Fig. 2c). This creates the impression of visual richness, reflecting the real physical properties of the drawing medium used.

A characteristic feature of hand-colored animations is that physical properties of the drawing medium are hard to control, so maintaining temporal coherency becomes tedious. The difficulty increases with the scale of details an artist wishes to preserve as coherent. Due to this, hand-colored animations contain specific spatial changes between individual frames that are perceived as high-frequency temporal noise when shown successively. The noise has flat power spectrum uniformly distributed energies over all frequency bands. Vision science offers an explanation of this perception with multi-channel models of human vision [Win05]. When the human visual system processes the temporal signal, two visual mechanisms, the transient and the sustained channels, come into play [KT73, Wat86]. The sustained channel performs a detailed analysis of stationary, or slowly moving, objects (low temporal frequencies) while the transient is involved in signalling the spatial location or change in spatial location (high temporal frequencies). The content of transient channel is therefore perceived as noise, stimulus flickering, or apparent movement [MRW94]. We hypothesize and experimentally measure (see Section 4.1) that the larger the spatial changes in frames, the higher the power spectrum of temporal frequencies, the higher the energy in transient channel, and accordingly the higher the level of perceived noise in animation.

This mechanism motivated us to design a new algorithm that enables control over the amount of perceived temporal noise (see Fig. 3). We render a sequence of images that have similar low-frequency content as the reference animation while high-frequency details are reintroduced by example in a random fashion. The user can then change the frequency threshold to increase/decrease spatial extent of synthesized details and thus control the level of perceived temporal noise.

In the rest of this section we formulate the problem more precisely and propose an algorithm to solve it. We also briefly mention simple extensions that can further improve the quality of the resulting image sequences.

3.1. Problem formulation

The input to our algorithm is a sample of a real drawing medium S and a sequence of N reference images R that represent a coherent, noise-free animation with a similar appearance to S (see Fig. 3). The task is to synthesize a target animation T that satisfies the following three criteria (see Fig. 4):

1. **Fine consistency.** Visual dissimilarity between source S and target Ti should remain small (i is the frame number). This can be accomplished by minimizing established patch-based energy [WS107]:

   \[
   \sum_{q \in I} \min_{p \in \mathbb{P}} ||P - Q||_2^2
   \]

   where Q denotes a patch of size w × w centered at the target pixel qi, and P is a patch of the same size taken from source pixel qi + j, possibly undergoing additional geometric transformations (we consider rotations and reflections).

2. **Coarse consistency.** Low-frequency content of Ti should be close to the low-frequency content of Ri. Formally we need to minimize the L2-norm over all pixels of the low-pass filtered signals:

   \[
   ||h_f * R_i - h_f * T_i||_2^2
   \]

   where h_f is the low-pass filter with tunable strength f and * is the convolution operator.

3. **Temporal noise.** Suppose R is a sequence showing a static image over several frames and qi is a 1D function yielding the value of a target pixel q ∈ Ti at the frame i. We would like qi to contain a signal with white properties, i.e., its power spectrum \( Q(\omega) = |\mathcal{F}(q)|^2 \) should have uniformly distributed energies over all frequency bands. Formally we can express this by minimizing the standard
CHAPTER 4. COLOR ME NOISY

Fine consistency. The algorithm of Wexler et al. [WSI07] utilizes image pyramids $\Delta S$ and $\Delta T$ to represent the source and target images at multiple scales. It starts with the coarsest level $\ell = 1$ and gradually upsamples the solution until the finest level $\ell = M$ is reached. At each level of the pyramid $\ell$ the following steps are performed iteratively:

- find nearest neighbor patches $P \subset \triangle^\ell$ for all target patches $Q \subset \triangle^\ell$ so that $|P - Q|_2^2$ is minimal.
- for each pixel $q \in \triangle^\ell$ compute the mode of colors at collocated pixels $p \in \triangle^\ell$ that belong to retrieved nearest neighbor patches $P$.

Coarse consistency. To integrate (2) into the joint optimization process we can exploit the fact that the original Wexler algorithm uses a multi-scale approach to optimize (1). In our setting the synthesis at lower levels of the target pyramid $\Delta_T$ is redundant since from a certain level $k$ a good solution $\Delta_T^k$ is already known: $\Delta_T^k = h_f * R_i \downarrow^k$, where $\downarrow^k$ denotes the downsampling operator that sets an appropriate sampling rate according to the strength $f$ of the low-pass filter $h_f$. This leads us to propose the following modified version of the original algorithm.

Given the source drawing medium $S$ and the user-specified strength $f$ of the low-pass filter $h_f$, we initialize source pyramid $\Delta S$ by low-pass filtering and subsampling $S$ at multiple levels $\ell = 1...M$:

$$\Delta_S^\ell = h_f(\ell) * S \downarrow^\ell$$

where $f(\ell)$ is a function which interpolates strength of the low-pass filter $h_f$ according to the level $\ell$. For a box filter where $f$ is the width of the box, $f(1) = f$ and $f(1) = 1$. Inbetween values are set so that the sampling rate of two consecutive levels decreases with a subite ratio of 0.85, as in the work of Simakov et al. and Shechtman et al. [SCSI08, SRAIS10], to reach finer granularity during the synthesis and help avoid visual artifacts.

Once the source pyramid is built we create a target pyramid $\Delta_T$ with the same resolution as levels of $\Delta_S$ and enforce (2) by feeding downsampld low-frequency content of the reference animation frame $R_i$ into the coarsest level of $\Delta_T$, i.e., $\Delta_T^1 = h_f * R_i \downarrow^1$. After this initialization the algorithm continues as usual.

Note that successive downsampling of reference animation leads to removal of high-frequency details and introduces fuzziness into the shape of region boundaries. This is a desirable effect which is characteristic for drawing media such as watercolor (see Fig. 2). Nevertheless, there can be situations when these irregularities are unintended. In such cases we provide mechanisms that allows to improve the quality of border synthesis using local noise control and source selection. These extensions are further discussed in Section 3.3 and supplementary material.

Temporal noise. Suppose we have the same simplified set-
so far would lead to a sequence of static images. Temporal noise as described in the formulation of temporal noise criteria, i.e., a reference animation $R$ that consists of a static image played over several frames. The algorithm proposed so far would lead to a sequence of static images $T$, where each pixel $q \in T$ would be constant over time. This is the situation we need to avoid as our aim is to produce a noisy sequence.

Direct minimization of (3) would be problematic as it requires computation in the frequency domain, operates over a large amount of data, and for moving objects complex optical flow estimation is necessary to compensate for the global motion. Rather than trying to minimize (3) explicitly we instead synthesize each frame independently and introduce randomness into the original deterministic algorithm by randomly voting over possible patch candidates and pre-deforming the source $S$. Later (in Section 4.1.1) we demonstrate that such a simplified solution is sufficient to obtain noisy sequences with equally distributed energies over all frequency bands as required by (3).

Recently, PatchMatch—a fast approximate nearest neighbor search algorithm [BSFG09, BSGF10] has become popular. Besides significant performance gains, it offers a kind of non-determinism that we can exploit in our scenario. The algorithm uses a random number generator to perform sampling over possible candidates in the space of source patches. Changing the seed of this generator causes the optimization to converge to a different local minimum, changing the appearance of the resulting image.

For low values of $f$ when the synthesis comprises only a few pyramid levels, the likelihood of changes caused by randomized PatchMatch reduces significantly. Accordingly, the temporal variance of the resulting sequence $T$ is insufficient to evoke perception of noise in the observer. We attribute this to two known perceptual principles: visual grouping [BL05] and feature fusion [SHK’07]. It was hypothesized that if two visual features have a “common fate” (e.g. they move slowly together in the same direction) and/or “close enough” in the successive frames, the observer is able to align and fuse them. They are thus perceived as a single object in an apparent motion. An effect of synthetic, unpleasant “floating texture” is then perceived instead the desired noise (see supplemental video for visual inspection).

We address this by randomly pre-deforming the source texture for each synthesized frame, constructing a control lattice with the control points randomly moved in a small radius. The result is deformed using an as-rigid-as-possible moving least squares approach [SMW06]. For our examples we set the grid size to 50 pixels and shift each point 15 to 25 pixels in a random direction. The average offset between synthesized features in two successive frames is above 20 pixels, which corresponds approximately to 20’ (visual arcminutes). This value is much higher than the theoretical minimal offset [SHK’07] needed for spatial superposition ($2’ \approx 2$ pixels). This ensures that generated random features are sufficiently far apart to avoid visual fusion.

Note again that the control over the amount of perceived temporal noise is not addressed in this step since it is already encoded in the previous coarse consistency phase by setting the strength $f$ of the low-pass filter $h_f$. The algorithm performs the synthesis starting from the initial coarse solution that corresponds to the low-pass filtered version of $R_t$ and then optimizes for fine consistency while using randomization to avoid getting stuck in the same solution. As the scale of randomly synthesized details increases with the increasing strength $f$ the resulting target animation $T$ appears to be more noisy to the observer (see Section 4.1.1 for evaluation).
3.3. Extensions

The proposed algorithm can be improved further to gain local control over the amount of temporal noise which can help to preserve salient structures (see Fig. 6), make the viewer pay attention to certain parts, or introduce additional channel of artistic expression (see supplementary videos). To enable this control, the isotropic $h_f$ in (2) is replaced with a spatially varying low-pass filter where for each pixel different strength $f_p$ is used. This change is incorporated into our algorithm by setting a different starting level for each pixel, i.e., at pixels with higher $f_p$ the synthesis starts at the coarsest levels of the image pyramid.

Figure 6: Local noise control—with higher levels of noise the overall shape consistency and presence of small but semantically important features are not guaranteed due to suppression of high-frequency details (left). By specifying the spatially varying strength $f_p$ of the low-pass filter $h_f$, sensitive parts can be synthesized with a lower noise level and thus preserved (right).

Orientation of the synthesized strokes (see Fig. 7) can also be controlled locally to emphasize the shape of the animated object or motion orientation. To do that the user can specify two additional orientation fields: $O_S$ for the source drawing medium and $O_R$ for the frames of the reference animation. These can either be obtained automatically, e.g., by computing the per-pixel structure tensor [BCK*13], or painted by the user. When fine consistency term (1) is evaluated $P$ is always rotated to compensate for orientation mismatch between $P$ and $Q$ and during the correspondence propagation in PatchMatch [BSFG09], axes-aligned directions are rotated to respect the actual orientation of $P$.

Figure 7: Local orientation control—prescribed orientations enable the algorithm to synthesize output that better follows the shape of the target region (right) in contrast to the uncontrolled synthesis (left).

Besides noise level and orientation, the choice of the source drawing medium can also be controlled locally to improve the quality of the synthesized image. Further details can be found in the supplementary material.

4. Results

We implemented our method using C++ except for PatchMatch [BSFG09], which was implemented in both C++ and CUDA. By default we use simple box filter for the low-pass filter $h_f$ of which the strength $f$ is expressed by the width of the box in pixels. When the source drawing medium contains sharp details a more accurate Lanczos3 filter [DSB*12] can be used to improve visual quality. For the fine consistency term we use patches of size $w = 7$ and perform 4 Wexler et al. [WSI07] optimization iterations using 8 PatchMatch iterations at each pyramid level. This number was set empirically to make a balance between effect of randomization and the final visual quality. A lower value causes visual artifacts while a higher value can suppress the effect of randomization as there is higher probability that the algorithm reaches a globally optimal solution.

Figure 8: Results—an additional set of 2D animations: (a) golem [crayon], (b) tree [watercolor], (c) dragon [fire]. See the supplementary video for animations in motion.

Figure 9: Examples from the evaluation dataset consisting of hand-colored snakes painted using different drawing media: (a) crayon, (b) chalk, (c) colored pencil, (d) regular pencil, and (e) watercolor. Top row: hand-made, bottom row: synthesized.

We applied our method to a set of four 2D and two 3D animations (see Fig. 1 and 8). For the 2D cases a shape in a rest pose was created and then a static textured image $R_0$ was synthesized using [LFB*13] based on a drawing medium $S$. This image was then deformed using as-rigid-as-possible deformation [SDC09] to produce the temporally coherent animation $R$. In 3D we mapped textures synthesized from $S$ using [LFB*13] on an animated triangle mesh and rendered the temporally coherent animation $R$. For each $R$ we synthesized $T$ based on $S$ in various noise levels and played them
Figure 10: Spectral analysis—chalk snake in Fig. 9b: (a) Average profiles of the spatial power spectrum of the target frame $T_1$ synthesized at 8 different strengths $f$ of the low-pass filter $h_f$ and a spectrum profile of the same frame drawn by an artist. (b) Average temporal power spectra of the target sequence $T$ synthesized at 8 different strengths of $f$ and the average spectrum of the animation drawn by an artist (motion in both sequences was compensated). (c) Normalized total power of visual channels for 8 different strengths of $f$. As $f$ increases, more information is processed by the temporal mechanism and accordingly more temporal noise is perceived.

While it would be possible to compare the visual plausibility of generated animations against these sequences using a two-alternative forced choice subjective experiment, it should be noted that such a comparison would not in itself be rigorous. This is because the natural animation contains multiple unknown hidden parameters, such as locally varying noise levels or orientation field flickering for anisotropic media, that would have to be matched first. Furthermore, such a comparison would be aesthetic at best, because it is impossible to judge the plausibility of the temporal noise separately from the plausibility of the still image, which is significantly affected by the selected synthesis method.

4.1. Spectral Analysis

We analysed the spectral properties of the synthesized sequences for increasing strengths $f$ of the low-pass filter $h_f$, both in spatial and temporal domains after motion compensation. Results for the chalk sequence (see Fig. 9b) are presented in Fig. 10 (for other media see supplementary material).

In the spatial domain the power spectra of the frames synthesized using different strengths are similar (Fig. 10a), i.e., the overall visual characteristic does not change significantly. The method does not introduce notable over-smoothing with increasing $f$; only a subtle sharpening effect is visible. When compared to the power spectrum of the real frame a more notable difference indicating subtle over-smoothing is apparent, i.e., the synthesized images do not look as sharp as the original painted by the artist. The amount of this smoothing effect varies across drawing media and is typically small enough so that the synthesized images look convincing (see supplementary material).

In the temporal domain the average power spectrum of $T$ has the energy distributed equally over all frequency bands (Fig. 10b), which corresponds to our aim to obtain characteristics of white noise. It is also visible that the higher the strength $f$, the higher the overall energy in the temporal spectrum. This indicates increased perception of temporal noise, which can be further verified by measuring power of sustained and transient channels [Win05, ACMS10]. Results
are illustrated in the supplementary material, and overall energies for the chalk sequence are plotted in Fig. 10c. These measurements confirm that the energy in the transient channel grows with the increasing strength $f$ of the low-pass filter $h_f$ and thus the perception of noise level increases.

### 4.1.2. Subjective Experiments

The spectral analysis above shows evidence that the increasing strength $f$ of the low-pass filter $h_f$ results in corresponding growth of temporal noise. However, the relation between the strength $f$ set by the user and the real quantity of perceived temporal noise remains to be investigated. Furthermore, it is also not clear how specific properties of the drawing medium (e.g., crayon, watercolor) affect the visibility of increasing noise level in animations and how this influences eye strain of the observer.

To that end we designed two subjective experiments with 50 and 64 participants, respectively. In the first experiment participants were asked to compare pairs of random sequences generated using our method and for each pair select a sequence that appears more noisy to them. In the second experiment we show just one sequence per question and ask the participants to rate the degree of eye strain they experienced while watching it. There were 4 simulated media: crayon, chalk, colored pencil, and watercolor (we excluded the failure case, regular pencil), and 8 generated levels of noise for each animation, i.e., 32 video stimuli in total.

![Figure 11: Overall results of subjective experiments—(a) the two-alternatives-forced-choice study on the perception of temporal noise and (b) the study of eye strain in hand-colored animations synthesized using our method. Error bars show the standard errors.](image)

The overall results of both studies are shown in Fig. 11. According the ANOVA tests [MR99] the null hypothesis “there is perceptually no difference between levels of temporal noise in the presented sequences” can clearly be rejected ($p < 0.001$), meaning that change of low-pass filtering strength $f$ produces sequences with perceptually different noise level. The same holds also for eye strain. The multiple comparison test (Tukey’s honestly significant differences [HT87]) returns an overall ranking of the individual noise levels and the eye strain with an indication of the significance of the differences. In the first experiment, there is a statistically significant difference between each level of temporal noise produced by each value of $f$. The second experiment exhibits two statistically significant groupings: chalk, crayon, watercolor (greater visual discomfort) and watercolor, pencil (lesser visual discomfort).

Furthermore, the first experiment did not show any statistically significant effect of the simulated medium on the level of perceived temporal noise. Nevertheless, the second experiment indicated that there may be a small effect of medium type on the eye strain. Results also indicate slight non-linear relationship between the strength of the low-pass filter $f$ and perceived amount of temporal noise. This motivated possible perceptual linearisation of our method, as shown in the supplementary material.

In summary, both studies confirmed there is a relationship between setting the strength $f$ of the low-pass filter $h_f$ and the levels of perceived temporal noise and eye strain. With increasing $f$ the level of noise and eye strain increases. More details about experiment setup and obtained results can be found in the supplementary material.

### 4.2. Comparison

For comparison purposes, we have attempted to adapt the methods of [LH05] and [BCK∗13] to synthesize results close to our hand-colored animation scenario. Clips comparing these algorithms with our approach are included in the supplementary video.

We have extended [LH05] in order to synthesize an animation with a configurable amount of temporal noise by manipulating the noise/scale settings and using an appropriate source of randomness. While the result was a reasonably consistent noisy sequence, as compared to our approach the algorithm is unable to preserve either high-frequency details of the original drawing medium or the prescribed low-frequency content. It also cannot easily provide local control over the amount of noise.

When attempting to use [BCK∗13] we encountered the problem that even when setting different weights to the temporal coherence term the synthesis tends to converge to near-identical results on consecutive frames when the shape or color of the object of interest do not change significantly. The only way to obtain noisy sequence was to deactivate advection vectors and let the algorithm synthesize each frame independently. However, this solution offers only one noise level which cannot be further controlled and there is no guarantee that the resulting sequence will be temporally coherent (see supplementary material for further details).

### 5. Applications

By combining TexToons [SBCC∗11] with our approach, one can produce hand-colored animation from a sequence of
outline-only hand-drawn sketches. Moreover, our technique can mask shower door artifacts that sometimes appear because of the approximative nature of the original TexToons framework (see Fig. 12 and the supplementary video).

Figure 12: TexToons—the output from the TexToons algorithm is used as a reference for resynthesis. The shifted texture in the original sequence is denoted by the green curve (upper row). With our approach (bottom row) consecutive frames do not suffer from the “shower door” effect.

Other possible applications of our framework such as stylization and imperfection masking in particle simulations, or painterly rendering of photos and videos can be viewed in supplementary materials.

6. Limitations
An implicit assumption of our method is that areas in the reference animation \( R \) have counterparts in the source \( S \) that are similar in the RGB domain. As our method draws the samples exclusively from \( S \), absence of a suitable source will change the color of the output to match the most similar one in \( S \) (see Fig. 13a–c). If this is not acceptable, color matching could be applied or different exemplar images provided.

A similar situation occurs when \( S \) contains multiple areas that have similar average intensity and chroma values and are only distinguished by their fine-scale structure. As the filter eliminates this information above a certain width, the distinction between these areas is lost (see Fig. 13e–f). Synthesis in such situations could be improved if some sort of structural descriptor was taken into account.

When \( R \) contains large areas of solid color the algorithm starts to produce artifacts (see Fig. 13g–i). It will also cause the noise level settings to be ineffectual and hinder synthesized frames from carrying the coherency information between frames. To rectify this, one may add some temporally-coherent texture to the solid areas of \( R \) as an overlay, using the workflow described in Section 4.

7. Conclusion and Future Work
We presented a new framework that allows the transfer of a hand-colored look to 2D and 3D CG animations. Its ability to control the amount of temporal noise provides a new channel of artistic expression, and enables the creation of longer sequences that are less distracting to the observer yet still preserve a lively hand-colored look. We showed that simply varying the strength of the spatial low-pass filter is sufficient to control the amount of perceived temporal noise, and demonstrated that the algorithm can mask visual artifacts in temporally coherent animations. As a future work we plan to extend it to handle more challenging situations (such as automatically distinguishing areas with different fine-scale structure) and extend local noise control to automatically suppress temporal noise in areas with high edge or saliency detector response.

Acknowledgements
We would like to thank Bill Plympton and Kristýna Mlynaříková for providing excerpts from their animations and all anonymous reviewers for suggestions and constructive comments. This research was funded by Adobe and partly supported by the Technology Agency of the Czech Republic under the research program TEO1020415 (V3C – Visual Computing Competence Center), by the Czech Science Foundation under research program P202/12/2413 (OPALIS), by the Grant Agency of the Czech Technical University in Prague, grant No.SGS13/214/0HK3/3T/13 (Research of Progressive Computer Graphics Methods), and by SoMoPro II grant (financial contributions from the EU 7 FP People Programme (Marie Curie Actions), REA 291782, and from the South Moravian Region).

References
CHAPTER 4. COLOR ME NOISY

Figure 13: Limitations—synthesized frames ($T_{a,b,c}$) may not properly convey the look of the drawing medium ($S_{a,b,c}$) when it contains different colors ($S_a$) from the reference ($R_a$) or subtle high-frequency details ($S_b$) that cannot be distinguished by intensity level or color, or when the reference frame contains solid colors ($R_c$).


Brushables: Example-based Edge-aware Directional Texture Painting

M. Lukáč,1† J. Fišer,1 P. Asente,2 J. Lu,2 E. Shechtman,2 and D. Sýkora1

1CTU in Prague, FEE, 2Adobe Research

Abstract

In this paper we present Brushables—a novel approach to example-based painting that respects user-specified shapes at the global level and preserves textural details of the source image at the local level. We formulate the synthesis as a joint optimization problem that simultaneously synthesizes the interior and the boundaries of the region, transferring relevant content from the source to meaningful locations in the target. We also provide an intuitive interface to control both local and global direction of textural details in the synthesized image. A key advantage of our approach is that it enables a “combing” metaphor in which the user can incrementally modify the target direction field to achieve the desired look. Based on this, we implement an interactive texture painting tool capable of handling more complex textures than ever before, and demonstrate its versatility on difficult inputs including vegetation, textiles, hair and painting media.

1. Introduction

Example-based image synthesis enables transfer of visual characteristics from a given exemplar to a user-defined target image [Ash01, HJO01]. In this context a texture-by-

numbers metaphor is typically used to guide the transfer of textural information between specific locations in the source and target images. Ritter et al. [RLC06] showed that the quality of the synthesis can be improved when the algorithm takes into account specific effects that occur close to the boundaries of individual segments. This edge-aware approach was recently improved by Lukáč et
al. [LFB∗13] who showed how to synthesize boundary effects in a direction-aware manner independently from the interior—the direction of synthesized boundary features exactly follows the direction of the boundary shape. Direction awareness was also previously used in general texture synthesis [ZZV∗03, LH06, ELS08, DBP∗15] to allow control of the orientation of the synthesized texture.

A key limitation of existing techniques is that they treat edge and direction awareness independently, making it hard to produce images where the prescribed directionality of the shape’s interior interacts with the appearance of its boundaries.

In this paper, we propose a novel method for interactive example-based image synthesis that combines edge and direction awareness in a single algorithm. While these features are useful independently to synthesize textual areas and linear edge structures with user-specified orientation, combining them enables complex shape-aware effects that no previous method can handle. See, e.g., examples in Figure 1 where the appearance of boundaries (e.g., blades of grass or hair ends) depends on a specific context that is given by the directionality of the interior.

Our method builds upon the popular patch-based optimization scheme originally developed by Wexler at al. [WSI07] and later extended by others [BSFG09, DSB∗12, KNL∗15]. A key contribution of our work is that we provide a new extension of the original Wexler et al. formulation that combines both direction and edge awareness into one optimization problem. We further improve the visual quality of the synthesized result using a novel coherence weighting mechanism.

We also propose a unified interactive framework that helps the user prepare the necessary input data for the synthesis. We extend previous related techniques for detecting KLC09, Kyp11 and authoring ZMT06, FSDH07 direction fields by creating a new signed direction field. The sign was not considered previously, and we show that it helps the user specify semantically meaningful configurations where unsigned orientation fields are insufficient (see, e.g., direction of hair/grass growth in Figure 1).

2. Related Work

One of the first instances of combining texture synthesis with a painting interface was Synthesizing Natural Textures [Ash01]. The user painted an output suggestion in the color domain, and the synthesis created output that roughly matched the colors. However, color information was not enough to finely specify texture areas.

Image Analogies [HJO∗01] alleviated this limitation with a texture-by-numbers approach. The user pre-segmented the input image and directly painted a segmentation mask. However, the lack of additional information about boundary orientation led to visible inconsistencies.

Painting with Texture [RLC∗06] represented a further development in this area. It was the first approach explicitly designed for synthesizing stroke interactions and texture edge effects by introducing a shape mask into the patch distance term. The mask provided rudimentary edge awareness, but its small size could not represent subtle orientation changes and larger sizes would make the synthesis over-constrained, causing visible repetitions and other artifacts.

Painting by Feature [LFB∗13] presented an improvement over the previous techniques by treating lines and edges separately from the interior texture. Instead of relying on pre-segmented input images, the user interactively selected a line feature or a texture to be used as an example and then painted them into the output canvas. Despite full creative freedom, this technique could become tedious, requiring painstaking boundary tracing even when edges were obvious. This method also did not provide an explicit control over the directionality of the texture in the interior regions.

RealBrush [LDBF13] is a canonical example of stroke synthesis, capable of transferring the directionality and edge effects of the input strokes directly to the result using a painting metaphor. However, since it uses a lengthwise cut-and-stitch approach instead of full synthesis, it is strictly limited to 1D curves and cannot synthesize arbitrary area structures. Other stroke synthesis systems [LBW∗14, ZLL13] typically suffer from the same limitation.

Accounting for directionality in texture synthesis is a proven idea. It can compensate for transformations ELS08, LH06 or allow specification of direction in textures [ZZV∗03, DBP∗15]. Detecting orientation in images is also crucial for various stylization techniques [HE04, KLC09, Kyp11]. However, a painting scenario such as ours requires further considerations. The direction fields should be authored seamlessly using the basic brush metaphor, and the detection needs to be configurable to ensure compatibility of input and output direction fields.

Structure tensors [BWBM06] and edge tangent flow [KLC07] are common techniques to detect orientation in textures. Their key limitation is that they cannot provide consistent direction: the orientation sign is either omitted or inconsistent in the final solution. However, this is crucial in our scenario because real textures typically contain asymmetric structures. Although there are techniques that try to find consistent direction [KLC09, XCO∗09], they typically fail on larger scales or when singularities are present in the input field.

User-guided authoring of vector fields has been extensively studied in the context of 3D surfaces ZMT06, FSDH07, CDS10, MBS∗11. Although these techniques compute smoothly varying vector fields from a sparse set of user-provided constraints, their main drawback is that every new constraint has a global impact on the resulting field. In
our scenario we would like to modify the existing field on-the-fly by adding new directional strokes whose local impact is controlled by the user.

Optimization-based texture synthesis methods [KEBK05, WSI07] are the current state of art for synthesis applications [DSB*12, FLJ*14, KNL*15]. They accurately reproduce exemplar structures at interactive rates, thanks to fast approximate nearest-neighbor search [BSFG09]. We take advantage of the flexibility of this framework to introduce edge and direction awareness, and make adaptations to mitigate artifacts introduced by the free-form nature of our scenario.

For edge awareness, we build upon shape descriptors, commonly used in computer vision [BMP02, BM01]. They examine large areas of the shape to properly consider context, which is computationally expensive. Because texture synthesis requires numerous evaluations in a short time-frame, this can be a bottleneck.

3. Our Approach

Figure 2 illustrates the workflow of our method. The user starts with a regular RGB image and uses interactive image segmentation and matting to extract the area of interest along with the opacity values. The resulting RGBA image source $S$ then serves as the basis for further processing (see Figure 2a).

Initially we take all pixels in $S$ with non-zero alpha to form a binary shape mask $M_s$ and then let the user determine the edge extent, i.e., how wide the boundary effects are. Finally, we employ direction analysis to obtain a source direction field $d_s$ with a desired level of smoothness and consistent sign of the tangent vectors (see Section 3.1).

In the following painting phase, the user uses a brush tool to paint a mask that defines the set of pixels to be synthesized—a he direction field $d_t$ and its shape mask $M_t$.

Then the use can alter or refine the target direction field (the combing process). For this our novel direction diffusion algorithm (Section 3.2) gives precise control over the stroke extent, seamlessly combining multiple strokes, and combining new strokes with the pre-existing direction field.

Finally, given the source image $S$, source and target masks ($M_s$ and $M_t$) and direction fields ($d_s$ and $d_t$) we run our direction- and edge-aware texture synthesis (Section 3.3). We synthesize the output texture, using a novel Shape Hint to ensure that boundary effects are synthesized appropriately in a context-sensitive way, enforcing the prescribed direction, and using a coherence weighting mechanism to improve the final visual quality of the synthesized image $T$ even under strong non-rigid deformation.

### 3.1. Direction Analysis

Before we can paint taking the directionality of the source into account, we need to estimate it. Our first step is to create a direction field $d_s$ that specifies local direction at all pixels of $S$. To support arbitrary input exemplars and have a self-contained approach, we determine the direction field using only the RGB color information.

For best results, a reasonable direction field $d_s$ should be locally smooth and perpendicular to the gradient field of $S$—a tangent field. Because smoothness and perpendicularity cannot usually be satisfied simultaneously, additional filtering is required. We also take the sign of the tangent vectors into account, since they are often semantically significant.

Estimation of smooth tangent fields has been explored before, predominantly in image stylization techniques [KLC07, Kyp11]. However, these approaches typically ignore the sign of the tangent vector, since the filters they ultimately employ are symmetric with respect to the sign. In particular, the multi-lateral filter employed by Kang et al. uses a non-linear term to preserve the sign of the tan-
CHAPTER 5. BRUSHABLES

3.2. Direction Diffusion

In previous approaches [DSB*12, LFB*13], the texture direction emerges implicitly from the color domain so as to match the boundary conditions. In contrast, we give the user explicit control over texture direction, much like stroke synthesis approaches do [ZLL13, LBW*14].

Related approaches use various optimization processes to construct a smooth direction field from sparse user-specified constraints [ZMT06, FSDH07]. However, these techniques are global by nature and do not provide for a localized, controlled way to combine new strokes with an existing direction field, which is needed to permit combing and general refinement. We use a kernel-based diffusion scheme to smoothly diffuse and blend the direction of an arbitrary number of strokes of variable radius, while also permitting blending with a pre-existing field.

Given a stroke path \( K \) consisting of all points \( k \in K \), we calculate the direction \( d_k(p) \) diffused from this stroke at a point \( p \) as follows:

\[
d_k(p) = \frac{1}{w_k(p)} \int_{k \in K} G(||p-k||^2, \sigma_k^2) \cdot d'(k)
\]

where \( G(x, \sigma^2) \) is a gaussian kernel with the standard deviation set to half the stroke width, \( d'(k) \) is the local normalized tangent, and

\[
w_k(p) = \int_{k \in K} G(||p-k||^2, \sigma_k^2)
\]

This yields a smooth interpolation that can be evaluated analytically if the input stroke is approximated as a polyline, and the generalization to multiple simultaneous strokes is straightforward (see Figure 4a). If we need to combine the diffused direction of the current stroke with the aggregated direction field of all the previous strokes (as in Figure 4b), we calculate the convex mix of the previous value \( d^{p-1}(p) \) and the new one \( d_k(p) \) like so:

\[
d^p(p) = w_k(p) \cdot d_k(p) + (1-w_k(p)) \cdot d^{p-1}(p)
\]

assuming \( w_k(p) \) is clamped to remain in the convex interval \([0,1]\).

Figure 4: A demonstration of direction field authoring and refinement. (a) a composition of thick strokes made with a 120px wide brush next to its synthesis result; (b) direction field with two 80px refinement strokes and the refined synthesis result.

3.3. Example-based Synthesis

Once source and target direction fields \( d_s \) and \( d_t \) are prepared we proceed to the synthesis phase, generating the output image while respecting the principles of edge and directional awareness we have described earlier.

We build our synthesis algorithm upon established the patch-based optimization framework introduced originally by Wexler et al. [WSI07]. We chose this framework for its flexibility: we can substantially alter its behavior by substituting our own patch distance measure and patch voting logic, making it fit our own requirements.

We introduce edge-awareness into the synthesis by adding a new \textit{shape distance term} to the energy function we minimize:

\[
E(T,S) = \sum_{q \in T} \min_{p \in S} \left( D_{\text{patch}}(p,q) + \lambda D_{\text{shape}}(p,q) \right)
\]

\( D_{\text{patch}}(p,q) \) measures the color distance of patches and \( D_{\text{shape}}(p,q) \) the distance of local shapes around pixels \( p \in S \) and \( q \in T \).

Direction awareness is added to these distance measures by taking local direction at both \( p \) and \( q \) into account. We do this by introducing a rotation operator \( \circ \alpha_{pq} \), which rotates the local frame of reference for the patch or shape descriptor.
Figure 3: An illustration of the output of Kang et al.’s [KLC07] method for orientation detection with and without our unified tangent sign initialization: (a) original synthetic image (radial stripes), (b) false-colour visualization of the initial direction field (gradients rotated 90° to the left), (c) converged result after a few ETF filter iterations, (d) initial direction field after our unified tangent sign initialization, (e) result after single ETF filter iteration applied on our unified tangent sign initialization.

by the difference in local direction at \( p \) and \( q \), i.e., \( \alpha_{pq} = d_s(q) - d_s(p) \).

We can then calculate the color distance as the direction-aware sum of squared differences:

\[
D_{\text{patch}}(p, q) = \left| \left| P_{sp} - P_{tq} \otimes \alpha_{pq} \right| \right|^2
\]

(5)

between the source patch \( P_{sp} \) centered on \( p \in S \) and the rotated target patch \( P_{tq} \) centered on \( q \in T \). Similarly, the direction-aware shape distance is evaluated as:

\[
D_{\text{shape}}(p, q) = \chi^2(\Pi_p, \Pi_q \otimes \alpha_{pq})
\]

(6)

i.e., the distance between source and target Shape Hint histograms described below, which introduce shape awareness by considering both the spatial distance from the texture boundary, and its shape relative to the local direction field.

Figure 5: Importance of Shape Hints: (a) synthesis without Shape Hint (using just distance to the boundary) and (b) synthesis with Shape Hint.

Shape Hint: To introduce edge-awareness into the synthesis we use Shape Hints—a local shape descriptor derived from the shape context [BMP02], which we have simplified and adapted for interactive use. Shape descriptors like these are a powerful tool commonly used to find similar locations within shapes. Compared to previous context aware solutions based on a distance transform [LH06, BCK13], a shape descriptor considers a larger context, allowing it to distinguish between locations at edges and corners or around interior holes; this is vital for our concept of edge-awareness, since it lets us pick patches from appropriate regions more contextually (see Figure 5). It is also more flexible than comparing mask patches, as in Painting with Texture [RLC06]; the distance measure is continuous rather than discrete and degenerates gracefully, without overconstraining the synthesis at texture edges.

Like the shape context, our descriptor counts the edge pixels that fall into “bins” mapped to image space. These counts are then treated as histograms of edge pixels and can be compared using the \( \chi^2 \) metric. This creates a descriptor that is capable of capturing the shape of the object boundary with a configurable level of tolerance to high-frequency variations, based on how large is the spatial support of the bins. To date, performance considerations precluded the use of shape context in texture synthesis, as typically local descriptors need to be evaluated repeatedly at many points of the image, and the computational complexity of evaluating a shape context scales quadratically with its radius.

To overcome this limitation, we propose an adaptation wherein we change the shape of the bins used to count edge pixels (see Figure 6). Instead of annular sections, we use circular bins, similar to image descriptors like...
CHAPTER 5. BRUSHABLES

Lukić et al. / Brushables

FREAK [AVO12], but we keep the shape context’s compact representation based on edge pixel counting and its method of calculating similarity.

As circles are rotationally invariant, the shape of the bin becomes constant with respect to both the orientation of the descriptor as a whole and the bin’s position therein. The value of any bin at any point can thus be pre-computed by convolving the edge pixel map with a disc filter of the appropriate radius, and consequentially, we can evaluate the descriptor with a constant number of bitmap queries regardless of its spatial support or the number of edge pixels in the image.

This not only leads to faster evaluation, making use in texture synthesis possible, but also permits free-form continuous rotations of the descriptor at no additional computational cost. We have found it sufficient to only use a single radial layer of bins, although the descriptor naturally generalizes to multiple layers.

In synthesis, we use the source and target masks \( M_s \) and \( M_t \) to calculate the Shape Hints, with the radius of the descriptor set to the edge extent that the user defined earlier in the source analysis phase. This value should roughly correspond to the width of the boundary effects, i.e., how “deep” into the texture they extend. Content within this range is implicitly treated as the boundary, while content deeper inside is considered to be in the interior.

**Alpha Channel:** To further improve the quality of the synthesis at boundaries we add an alpha mask as an additional pixel channel. This has two effects. It lets us synthesize opacity, and together with the Shape Hint, guides the synthesis towards a solution where pixels close to boundaries in the source are more likely to be matched with boundary pixels in the target. To give the opacity comparable weight to color we multiply the difference in alpha channel by 3 when computing the sum of squared differences in (5).

While the alpha channel gives us the ability to synthesize opacity and “fading out” at the boundaries, it is in itself not sufficient to capture longer-range edge effects, and cannot discriminate boundaries from the interior in textures with partially transparent interiors. Therefore, a combination of alpha channel synthesis and shape matching is optimal for synthesis of edge effects in our scenario.

**Optimization:** To minimize (4) we use the Expectation-Maximization optimization outlined by Wexler et al. [WSI07] that consists of alternating search and voting steps on an image pyramid in a coarse-to-fine order. To improve texture coherence and richness in the synthesized image, we propose an improvement to the voting step to take both local nearest-neighbor field coherency and the color histograms of both images into account. When evaluating the final color \( C(p) \) of a pixel \( p \), we iterate through the overlapping patches mapped to its neighborhood and perform a weighted average of the candidates \( c \), gathered from them:

\[
C(p) = \frac{\sum_{q \in N} w_c(q) \cdot w_h(q) \cdot C(q)}{\sum_{q \in N} w_c(q) \cdot w_h(q)}
\]

where \( w_h \) is the color histogram weight of the candidate pixel, as detailed by Kopf et al. [KFCO07] and \( w_c \) is the coherence weight, which serves to propagate coherent arrangements of patches from the source. As described in the original paper, the histogram weight promotes pixel candidates with relatively underrepresented colors, improving the diversity of the synthesized image.

The coherence weight is vital in our scenario, since free-form rotations of the texture tend to induce non-rigid mapping in the nearest-neighbor field, which in turn causes blurry and visually displeasing results (see comparison in Figure 7). By increasing the weight of coherently-mapped configurations of patches, we encourage forming larger, coherently mapped areas over multiple iterations. This preserves high-frequency detail and causes less blurring.

![Figure 7: A comparison of results (a,c) and corresponding nearest-neighbor fields (b,d) synthesised with (left) and without (right) the coherence weight. The details (e,f) show how structural details of individual blades are better preserved with the coherence weight. Note also how the coherence weight leads to larger patches in the nearest-neighbor fields.](image)

To calculate the coherency weight, we examine the coherency of mapped pixel configurations as follows:

In effect, a nearest-neighbor match is a rigid mapping from \( T \) to \( S \). The matched coordinates and relative rotation at a pixel \( q \) thus define a mapping \( R_q \), which maps the pixel grid in \( T \) to a rotated and offset pixel grid in \( S \). Because the optimization is based on the assumption that these mappings are approximately identical for the group of pixels within the area of a patch, we design our coherency measure as a quantification of how this assumption holds. To evaluate this
measure, we examine the patch neighborhood of a pixel \(q_0\) and the induced mappings therein (c.f. Figure 8):

\[
\begin{align*}
    w_c(q_0) &= \sum_{q \in N(q_0)} G(||R_{q_0}(q) - R_q(q)||^2, \sigma_c^2) \\
            &= \frac{1}{\sigma_c^2} \exp\left(-\frac{||R_{q_0}(q) - R_q(q)||^2}{2\sigma_c^2}\right)
\end{align*}
\]

where \(\sigma_c^2\) is the coherency range, which we set to 2 throughout.

Figure 8: Calculating the coherence of a patch: we examine all pixels \(q\) in a patch around the pixel \(q_0\) in the target image \(T\). The position of pixel \(q\) is projected into the pixels \(p\) and \(p'\) in the source image \(S\) using both the rigid transformation induced by the match at \(q_0\): \(p = R_{q_0}(q)\) and its own transformation: \(p' = R_{q}(q)\). The more coherent the matching is, the lower the sum of distances \(||p - p'||\) (red arrow).

Multiplying these weights, along with the guaranteed range on both of them, ensures that the weighting scheme degenerates gracefully in any edge case.

3.4. Implementation Details

We have implemented the described algorithm in C++11 and run it on a desktop computer. The synthesis takes approximately 5 seconds for a megapixel output image, with 80% of the time spent calculating the nearest-neighbor field. The texture analysis step was more computationally intensive, taking up to 30 seconds for larger settings of the ETF filter; however, this only needs to be done once for each source texture as a pre-process, and the results can be efficiently stored. The rest of the method operates interactively.

As most of our parameters have intuitive semantics, they were set contextually as appropriate. The range of the edge tangent flow filter was usually set to a default value of 10 pixels. This setting was only increased for noisier textures to approximately 35 pixels. \(\lambda\) in Equation 4 was hardcoded to a value of 25 (equal to the number of pixels in a patch used to measure color distance), and the width of the brush was interactively adjusted as appropriate.

Edge extent was the crucial parameter to achieve edge awareness; setting it too low can cause boundary patches to be randomly used in the interior, while setting it too high can cause the extent of the synthesized texture to visibly deviate inwards from the user-specified mask. Ultimately, the value in pixels should be set approximately equal to the width of the boundary effects the user wishes to capture.

4. Results

Figures 1 and 14 shows the synthesis results of a variety of natural textures. Our approach coherently synthesizes textured outputs with direction configurations not present in the original source (see e.g., the crochet results). Plank example demonstrates how our algorithm picks the semantically correct edge features according to local direction. The grass and the colored pencil examples show that even transparency is synthesized correctly both on the boundary and in the interior. The braided wig example serves to highlight the strength of shape descriptor-based edge awareness; the narrower sections are synthesized from braid patches, while the wider parts are synthesized out of the upper, combed part of the exemplar. Again, current approaches do not have such capabilities. The ornamental leaves are an example of a relatively simpler stroke synthesis application. It shows how our approach organically synthesizes branching by virtue of not relying on stroke synthesis. The red wig shows how locally-variant anisotropic textures can be coherently deformed to novel configurations.

Note that after texture analysis, the only user input we require are the brush strokes. Thus, our tool places no more burden on the end user than a regular brush. The overall interaction takes only a couple of seconds, depending mainly on the ability of the user to draw individual strokes (see supplementary video for an example of interactive sessions). This brings an improvement over Painting by Feature [LFB+13], which requires more elaborate input to achieve similar results (see Figure 10).

Figure 9: A example result when using multiple textures. The top row is the example and its segmentation, bottom row the synthesized output and its painted segmentation. Source credit: Radu Bercan @ shutterstock

Our method can also be easily extended to process images with multiple, segmentable textures (see Figures 9 and 11). In this case, we require that a segmentation map be provided for the input texture, and the direction brush is concurrently used to paint also the output segmentation. The synthesis is then adjusted so that it only maps patches between compatible segments. Segment boundaries are considered in the same way as foreground boundaries for the Shape Hint.
Figure 10: Comparison of the amount of user interaction required to create a similar output using Painting by Feature (PBF) [LFB’13] and our approach. Each colored line in the PBF example represents a user stroke (area selections are not shown); in contrast, our approach produced the result with only two strokes. Also compare the coherence of texture on the interior and the tassels.

Figure 11: Comparison of our approach with Painting by Feature (PBF) [LFB’13] with respect to ability to handle edges with highly varying width. Note how our unified approach integrates interiors with edges smoothly, whereas in the PBF output there are discontinuities between the areas synthesized as edges and those synthesized as interiors (see red arrows). In the bottom example, PBF is able to more closely match the user-specified shape, but does so at the cost of faithfulness to the example and visual richness. Source credit: monkey: © ACM; hedge source: Joe Shlabotnik @ flickr

Multiple texture extension allows us to make a comparison with Painting by Feature [LFB’13] (see Figure 11 and supplementary material). Our method produces comparable or better visual quality without the necessity to use a custom synthesis algorithm for the boundaries. It also notably improves the look of the interior parts by maintaining the appearance of the original source and creating seamless transitions from the edge that follows prescribed direction field.

Our method can also be used to synthesize example-based brush strokes of comparable quality to those produced by RealBrush framework [LBDF13] (see Figure 12). In addition, the same algorithm can be applied to fill larger areas, which the original RealBrush method cannot do.

In Figure 13 we show results where only the edge or direction awareness is taken into account. This example demonstrates limitations of previous approaches (such as [RLC’06] or [LH06]) where a joint edge- and direction-aware formulation was not considered.

4.1. Limitations

Our algorithm does not automatically take changes in texture scale into account, nor does it natively compensate for perspective. Support for these could be added by pre-processing the input image to compensate for these.

Because our algorithm does not take advantage of any domain knowledge, it cannot replicate stroke-specific effects that require such knowledge. Most significantly, the smudging and smearing effects supported by RealBrush [LBDF13] cannot be replicated. Instead, overlapping strokes merge into a single larger area and are synthesized as such (see Figure 11 right).
Sources Results

Figure 12: Brushables can also be used in the RealBrush scenario [LBDF13]. Our approach can synthesize new strokes like RealBrush can, and also synthesize regions of arbitrary shape.

Furthermore, because we rely on an area representation rather than an outline-based one, our approach does not natively handle interior lines like Painting by Feature [LFB+13] does. This effect could be emulated by selecting the line in the example as a separate texture, painting that and combining the results. Still, the nature of our brush-based interaction model makes this less convenient than similar operations are for vector-based tools.

In some textures, there may be hidden variables not related to direction that affect incidence of features both on the edges and on the interior; these might include e.g. the holes in the cracker, or the precise position of the hairband in the braid. In such cases, our approach is unable to distinguish the underlying semantics and will distribute these features randomly. The ability to specify manual constrains, such as the ones used in appearance-space texture synthesis [LH06], could allow the user to resolve these cases.

Only direction-aware Only edge-aware

Figure 13: Results from Figure 11 with the shape awareness and direction awareness turned off. Those examples demonstrate importance of joint formulation proposed in our framework and illustrate limitation of previous approaches, which take into account only direction [LH06] or edge [RLC+06] awareness.

Our algorithm also exhibits some of the artifacts of the original synthesis method of Wexler et al. [WSI07], namely the repetition of textural features. Extensions to this optimization scheme that eliminate these have been proposed [KNL+15, JPA+15]; we consider these to be orthogonal to, and compatible with, our work.

5. Conclusion and Future Work

As discussed above, our approach handles complicated natural textures using a simple mode of interaction demonstrated earlier. Adding direction awareness to the synthesis process lets us handle textures with locally-variant anisotropic properties without requiring large exemplars or losing information. Our direction detection and authoring framework give users control over the output direction field that is semantically significant for many textures.

Adding the shape hint to texture synthesis enables robust handling of edge effects. Combined with alpha-channel synthesis, our approach can reproduce edge effects present in partially transparent textures. As a result, edges need not be explicitly drawn by the artists any more.

When combined, these two features become even more powerful, allowing semantically significant edge areas to be used for synthesis in different places. This allows artists to use previously unaccessible textures for true interactive texture painting.

For future work, we would like to better handle the cases where the direction configurations in the source do not match the target direction field. One possible solution is to automatically adapt the target direction field in a constrained and meaningful way. Furthermore, we would like to experiment with our shape hint in the domain of shape synthesis. It might be able to give rough user sketches the same type of high-level detail that a source shape does. Another possible avenue is to synthesize the mixing of textures using a blending approach like Image Melding [DSB+12].

Our approach integrates naturally into digital painting pipelines, thanks to its intuitive mode of interaction. Its ability to handle painting media exemplars lends itself to the creation of digital art. The ability to synthesize complex natural textures with edge effects make it useful for photo editing or matte painting applications.

Acknowledgements

We would like to thank all anonymous reviewers for their constructive comments. This research began as an internship by Michal Lukáč at Adobe Research and has been
Figure 14: Various sources (top): cracker, crochet, denim, sample of color pencil, bread, red wig, braided wig, ornamental leaves, plank, and grass were used to synthesize target images (below). Note how our approach handles both linear structures and regions with boundaries and how user-specified directions are gracefully preserved in the result. Source credits: cookie: Alessandro Paiva @ rgbstock; crochet: anneheathen @ flickr; denim: inxti @ shutterstock; bread: Giles Hodges @ DeviantArt; red wig: Lenor Ko @ shutterstock; braided wig: Karina Bakalyan @ shutterstock; ivy leaves: Michael & Christa Richert @ rgbstock; plank: My Life Graphic @ shutterstock; grass: varuna @ shutterstock
CHAPTER 5. BRUSHABLES

Lakáš et al. / Brushables

supported by the Technology Agency of the Czech Repub-
lic under research program TE01020415 (V3C) and by the
Grant Agency of the Czech Technical University in Prague,
grant No. SGS13/214/OH3/3T/13 (Research of Progres-
se Computer Graphics Methods).

References


6 Conclusion

We have presented three novel methods for example-based image creation. In this chapter, we will summarize the developments presented therein, briefly mention concurrent developments and touch on possible avenues for future work.

6.1 Summary

Chapter 3 presented an approach for example-based painting founded on treating boundaries and curvilinear features as a special case with its own synthesis algorithm. The important conceptual point was introducing the idea of example-based toolsets with multiple complementary tools, each focusing on transferring a different aspect of the exemplar’s visual style. Independent synthesis of line features is also necessary in order to represent exemplars with high-order structure, as the classical texture synthesis approaches lose this structure on 1D features when downsampling during image pyramid construction.

In Chapter 4, we presented an approach for example-based video stylization. The novel aspect of this approach is noise control; rather than trying to suppress any sort of temporal incoherence, this approach offers a spectrum of options between full suppression to high noise, which can further be locally and temporally variant. This allows for a new channel of artistic expression that was not available with previous approaches, and permits the use of styles that could not have been emulated before, such as hand-coloured animation.

Chapter 5 presented an example-based painting approach focusing on difficult texture examples. At the core of the approach was the integration of shape into similarity function, which facilitates seamless handling of complex boundary effects. In conjunction with modern texture analysis and segmentation tools, this new approach is especially suited for painting with textures acquired “in the wild”.

6.2 Concurrent and Future Work

Along with the one presented in Painting by Feature, several approaches intended strictly for synthesis of curvilinear features were developed [Lu et al., 2013; Zhou et al., 2013; Lu et al., 2014]. These were more specialised and produced better results in their chosen cases but did not integrate with area texture synthesis. While more advanced texture painting approaches like Brushables can synthesise texture boundaries satisfactorily, they still tend to have trouble with highly structured regular line features. A joint optimization approach that somehow synthesizes structured line features in the context of a textured area could be able to address this shortcoming while integrating seamlessly with pure texture synthesis, but such has yet to be developed.

The work on video stylization [Fišer et al., 2014; Bénard et al., 2013] has been followed up by further video stylisation work [Jamriška et al., 2015] which, like concurrent texture synthesis work [Kaspar et al., 2015], focused on implementing a new synthesis approach which would not only faithfully reproduce individual textural features but carefully preserve their relative frequency. Furthermore, satisfying both of these requirements together prevents loss of detail, such as plagued previous approaches when used for incremental re-synthesis. While the cited work solves this for a single-texture case, or even for cases with known ground-truth segmentation, preserving the proper ratio of textural features in stylisation scenarios with a continuous
label space on guidance channels (such as is supported by e.g. Image Analogies [Hertzmann et al., 2001]) is as yet an unsolved problem.

In the area of variation synthesis, the recent advances in stylisation feed back into this area as well, but synthesis of variation in structure remains a problem. Former approaches have used multi-pass synthesis to first synthesize a structure map which is then stylised [Rosenberger et al., 2009] but current texture synthesis approaches cannot capture long-range structural dependencies, or meaningfully synthesise a variation in them. There are computer vision approaches which may provide a functional generative model of such structure [Eslami et al., 2012] but there are currently issues (such as low working resolution and long training times) preventing their use in image synthesis. An example-based synthesis approach for shape and structure could be combined with image stylisation pipelines to make variation synthesis feasible for consumer use.

In conclusion, research, of which this thesis was a part, has in recent years both laid the groundwork for example-based image creation and successfully applied it for consumer use (e.g. in software like Adobe Photoshop). Unsolved problems remain, beside those detailed above, but this area is one of active research. In the near future, we expect new methods to take advantage of the cornucopia of exemplar data available from acquisition by mobile devices and the emergence of example-based toolsets that will integrate with the traditional ones in consumer software. Likewise, new development in variation synthesis approaches can be expected to tie in with procedural methods for generating virtual worlds, such as CityEngine.
References


REFERENCES


REFERENCES


REFERENCES


A  Author’s Publications

The following publications were co-authored by the author of this thesis and published in high-impact journals, as indexed by ISI.


Cited in:


B Authorship Contribution Statement

This statement describes the specific contributions of the author of this thesis to the publications presented therein,

Painting by Feature: Texture Boundaries for Example-based Image Creation (50%)

In Painting by Feature, I have formulated the initial idea of combining contour synthesis with area fill and exposing both to the user to facilitate the creation of structured images. I developed the prototype implementation, save the area fill and snake algorithms. I created some of the results and provided the algorithmic support for my collaborators to create more. I wrote the initial draft of the paper and created the illustrations, and took part in editing of both.

Color Me Noisy: Example-based Rendering of Hand-colored Animations with Temporal Noise Control (15%)

In Color Me Noisy, I have contributed to the development of the prototype by attempting to adapt PBF edge synthesis to video, which was however later discarded. I assisted in the development of the final software, conducting the user study, as well as the writing of the final paper and the creation of the accompanying video.

Brushables: Example-based Edge-aware Directional Texture Painting (50%)

In Brushables, based on the use-case presented by my internship mentor, I formulated the algorithms for both direction detection and shape synthesis, including the use of a shape term to guide the synthesis of texture edges. I have implemented the entire pipeline in a prototype, tested it and produced many of the initial results. I assisted with the creation of further results, wrote the draft of the paper and created the illustrations, and contributed to the editing and polishing of both. I also implemented the extensions to the algorithm required by the reviewers, such as multi-texture painting.
Supplementary material for the paper
“Painting by Feature: Texture Boundaries for Example-based Image Creation”

Figure 1: Additional results of interactive example-based painting: (1) source image, (2) source image with annotations, user-selected line and area features, (3) target strokes (lines) and fills (dots) correspond to the previous input selection, (4) resulting image. Source credits (top to bottom): (a) Vincent Van Gogh; (b) Sam Howzit via flickr; (c) Anifilm, clipartsy; (d) bitbox via flickr

References


Figure 2: Additional results of interactive example-based painting: (1) source image, (2) source image with annotations, user-selected line and area features, (3) target strokes (lines) and fills (dots) correspond to the previous input selection, (4) resulting image. Source credits (top to bottom): (a) bitbox via flickr; (b) Martina Cecilia via deviantART; (c) Georgia Democrats via flickr; (d) Claude Monet.
Figure 3: Additional results of interactive example-based painting: (1) source image, (2) source image with annotations, user-selected line and area features, (3) target strokes (lines) and fills (dots) correspond to the previous input selection, (4) resulting image. Source credits (top to bottom): (a) Andrea Garcia via flickr; (b) Nicolas Bonneel via flickr; (c) rmkoske via flickr; (d) Pavla Sýkorová. Result copyright: (c) CC-BY-SA Jakub Fišer
Figure 4: Fully annotated sources and resulting images shown in the main paper (Figure 1): (1) source image, (2) source image with annotations, user-selected line and area features, (3) target strokes (lines) and fills (dots) correspond to the previous input selection, (4) resulting image. Source credits (top–bottom): (a) Sarah G via flickr, fzap via OpenClipArt; (b) Pavla Sýkorová, cliparty; (c) bittbox via flickr, papapishu via OpenClipArt; (d) Anifilm, Pavla Sýkorová
Figure 5: Fully annotated sources and resulting images shown in the main paper (Figures 2, 7 and 12): (1) source image, (2) source image with annotations, user-selected line and area features, (3) target strokes (lines) and fills (dots) correspond to the previous input selection, (4) resulting image. Source credits (top to bottom): (a) Wednesday Elf – Mountainside Crochet via flickr; (b) Hrishikesh Premkumar via flickr; (c) Paul Cézanne (top); (d) Vincent Van Gogh, Kaldari via Wikimedia Commons.
Figure 6: Fully annotated sources and resulting images shown in the main paper (Figure 11): (1) source image, (2) source image with annotations, user-selected line and area features, (3) target strokes (lines) and fills (dots) correspond to the previous input selection, (4) resulting image. Source credits (top to bottom): Martouf via OpenClipArt (monkey head vector image); (a) Joe Shlabotnik via flickr; (b) Andrea Garcia via flckr; (c) Pavla Sýkorová; (d) Alessandro Andreuccetti via deviantART
Figure 7: An application to image compression in spirit of [Guo et al. 2007]: (a) The original image, (b) Image vectorized by tracing, (c) The saved representative samples of individual features, (d) reconstructed image. Source credit: Animfilm

Figure 8: For completeness, we present the full input to create the results by Image Analogies [Hertzmann et al. 2001] shown in Figure 2b of the main paper. Given the pair (a,b) and the sketch (c), Image Analogies creates the result (d) in such a way that the relation between (a) and (b) is the same as for (c) and (d). Both sketches (a) and (c) have been hand drawn. Source credit: Wednesday Elf – Mountainside Crochet via Flickr

Figure 9: Fill tool interaction with open brush strokes. (a) Source image with delineated area features. (b) Linear features specified by the user. Result of the fill tool: (c) when the whole image was taken as a source [green rectangle in (a)] and (d) when only a small portion of it was used [cyan rectangle in (a)]. Note how the absence of suitable area features in the restricted source caused less pleasing result (d). Source credit: bittbox via Flickr
Figure 10: An example of potential application of our proposed approach in the context of image inpainting, in a similar way to Sun et al. (2005): (a) original image, (b) artificial hole for inpainting with green curves representing line features we want to preserve, (c) intermediate result with our line feature synthesis, (d) final result after filling in the unknown regions with our fill tool. Source credit: Pavla Sýkorová
Supplemental Materials for the Paper:

Color Me Noisy: Example-based Rendering of Hand-colored Animations with Temporal Noise Control

1. Introduction
In this document, we present extra material for the paper Color Me Noisy: Example-based Rendering of Hand-colored Animations with Temporal Noise Control. Section 2 describes other possible extensions of our algorithm and how they can be used. Section 3 contains detailed results of the perceptual studies that we conducted to verify our method. Section 4 compares our method to some of the previous approaches, and Section 5 discusses further results. Finally, Section 6 demonstrates two additional applications of our method.

2. Extensions
In the main paper we mentioned extensions of the basic algorithm which can improve the quality of the resulting sequences. In this section we add few additional remarks and mention other possible modifications.

2.1. Local source selection
Hitherto we considered nearest neighbor candidates from one source $S$ in the fine consistency term. Nevertheless, in some specific cases, locally controlled selection of a different source $S'$ might improve the quality of the synthesized image (see Fig. 1). Recently, Lukáč et al. [LFB'13] demonstrated the importance of texture boundaries for the synthesis of believable synthetic images. A solution was proposed where line features are synthesized independently using an algorithm that takes into account their specific 1D structure. To reach visual quality comparable to Lukáč et al. we can pre-render high-quality linear strips of source pixels using [LFB'13] and then run the algorithm while constraining boundaries to map only to pixels in this additional source $S'$.

2.2. Local control as artistic expression
In the main paper we noted that local temporal noise control can be utilized as a separate channel of artistic expression. In the supplementary video, we demonstrate two use cases for this option. In the strongman clip, the impression of the strongman’s effort is emphasized by setting the biceps to be locally more noisy. Similarly, in the snake example created using interactive deformation, the noise level was set to convey the local amount of tension in the deformation model.

2.3. Interactive feedback
Because synthesis of an animation with a variable noise level takes as much time as regular synthesis, it is not inherently interactive. When interactivity is needed, however (e.g. when the artist is editing the control map for the noise and requires immediate feedback) we are able to get a preview by synthesizing the animation several times with different noise levels, and blending them according to the control map. Any blending artifacts are removed in the final synthesis step.

3. Perceptual Experiments
For the purposes of this paper, we have conducted a pair of perceptual experiments. In the first experiment the aim was to confirm that the setting of the noise parameter in our method indeed corresponds to the perceived amount of visual noise. The second experiment was meant to verify our claim that watching a hand-colored animation causes strain to the viewer, and that the amount of this strain corresponds to the amount of visual noise in the animation. The detailed setup and results of both experiments are described below.

3.1. Noise Perception
The Fourier noise analysis (see Section 4.1.1 in the paper, and Figures 14, 15) shows that increasing the low-pass filter strength $f$ results in a corresponding growth of temporal noise. However, the relation between $f$ and the perceived amount of $i$ is as yet unclear. Furthermore, the choice of drawing medium might further effect the relationship between these quantities. In order to determine the nature of these

submitted to Eurographics Symposium on Rendering (2014)
relationships, we have designed a subjective experiment as follows.

3.1.1. Experimental Setup

The observers were presented with a pair of animations generated using our technique with different noise level settings, and subsequently asked to indicate which seems more noisy to them. To prevent the choice of animation from affecting the results, the sequences were generated using a static image as the reference animation. A total of four media were used (crayon, chalk, pencil and watercolor) and 8 noise levels were generated from each, netting a total of 32 animations.

3.1.2. Design and Procedure

The experiment followed a two-alternatives-forced-choice procedure [Dav88] implemented as a web page for online testing. Accordingly, the evaluated sequences were presented on uncalibrated displays, the observing distance was not controlled and no time constraints were imposed. The presented sequences as well as noise levels were randomized. The experiment took on average 10 minutes per observer. Each of our 64 observers (we did not store information about participant’s gender and age) was introduced to the problem before the experiment as follows: “Thank you for agreeing to participate in this experiment. Please, wear your prescription glasses, if you would normally wear them to work with a computer or if they improve your vision. The purpose of the experiment, which will take about 10 minutes to complete, is to measure the visibility of temporal noise in hand-colored animations. During the experiment, you will see several video pairs, one video pair on the screen at a time. Your task is to select the video you perceive as temporally more “busy” by clicking at the button below it, i.e., select the video that appears more noisy to you.”

3.1.3. Agreement and Consistency

The agreement between observers was quantified by a coefficient of agreement \( u \), ranging from \( u = -1/(s-1) = -0.016 \), where \( s = 64 \) is the number of observers, (which indicates no agreement) to \( u = 1 \) (all observers responded the same). The obtained value \( u = 0.46 \) is quite high for this kind of experiment, meaning the observers had similar preferences during the judgement. Accordingly, the \( \chi^2 \) test of significance clearly shows the statistical significance of \( u \) (\( \chi^2 = 1777.6, p < 0.001, 28 \) degrees of freedom), meaning the measured subjective responses are not random.

The coefficient of consistency \( \zeta \) shows internal reliability of each observer’s set of choices and it ranges from \( \zeta = 0 \) (no consistency) to \( \zeta = 1 \) (ideally consistent responses). The average consistency over all participants \( \zeta_{avg} = 0.847 \) indicates that the observers were quite consistent in their responses and most of them did not change their preference during the experiment.

3.1.4. Study Results

Each observer performed 60 randomized pairwise comparisons out of \( m \times n(n-1)/2 = 112 \) possible, where \( m = 4 \) media (crayon, chalk, pencil, watercolor) and \( n = 8 \) is the number of levels of low-pass filter strength \( f \). The video chosen by an observer was given a score of 1, the other a score of 0. The data was stored in four \( 8 \times 8 \) frequency matrices for each observer, where the value in column \( i \) and row \( j \) repre-

submitted to Eurographics Symposium on Rendering (2014)
sent the score of tested sequence \(i\) compared with sequence \(j\).

We acquired standard \(z\)-scores [LM00] from the cumulative frequency matrices using the maximum likelihood estimation [TG11]. The obtained overall results and the scores for each medium separately are shown in Fig. 3. The plots depict \(z\)-scores for all observers along with the confidence intervals. The lowest value of \(f = 1.3\) results in the lowest perceived temporal noise for all simulated media. However, the level of perceived noise here is a bit lower than expected. This is due to the fact that the synthesis process starts in full resolution for the lowest \(f\), but it operates on lower resolutions for higher values of \(f\). The initial level is then slightly undervalued in the number of temporal changes in the sequence. In any case, it is clear that the perceived noise level grows with increasing the value \(f\) of the low-pass filter for all the tested media. This is also in accord with simulated temporal visual mechanisms, the transient and sustained channels. Similarly to Wikkeloo [Win05] we simulate the temporal behavior using two temporal filters, see Fig. 2. The content of transient channel is perceived as temporal noise, stimulus flickering or apparent movement [MRW94]. The simulation results (see Fig. 4) exhibit that increasing the \(f\) leads to growth in energy of transient channel, while the energy of sustained channel decreases.

![Figure 2: Frequency and impulse responses of the temporal sustained and transient visual channels exhibiting low-pass and band-pass behaviour.](image)

The \(z\)-scores calculated from the observation data are normally distributed, and we can therefore utilize classic parametric statistics in the further analysis. To evaluate the significance of the results we apply one-way ANOVA test [MR99] per simulated medium, see Fig. 5. The null hypothesis “there is perceptually no difference between levels of temporal noise in the presented sequences” can clearly be rejected (\(p \ll 0.001\)) for all media. This means that changing the strength \(f\) of the low-pass filter in the proposed method produces sequences with perceptually different noise level for all tested media. Furthermore, we are curious to see if the drawing medium affects the level of perceived noise and if the observers had particular preferences during the experiment. To that end, we apply three-way ANOVA test on the full set of 8 \((f\) levels) \(\times 64\) (subjects) \(\times 4\) (media) = 2048 \(z\)-score values. Three-way ANOVA enables us to assess the effect of all three following factors at once: low-pass filter strength \(f\), simulated medium, and observer; see Fig. 6. As expected, the results indicate that the filter strength \(f\) has indeed significant effect on data. More surprisingly, there is no statistically significant effect of the simulated medium; i.e., our experiment did not show statistical difference in perceived noise levels due to different media. The test revealed that there is an interaction between specific observer’s responses and the filter strength (statistically significant interaction effect). This may indicate that some subjects had different interpretation of the noise strengths. The multiple comparison test (Tukey’s honestly significant differences [HT87]) returns an overall ranking of the \(f\) levels with an indication of the significance of the differences. The overall scores of perceived temporal noise produced by tested values of \(f\) can be divided into seven statistically significant groupings; i.e., for our sample size, there is statistically significant difference between each level of temporal noise produced by each value of \(f\).

![Figure 5: Results of one-way ANOVA tests for perceived temporal noise level per simulated medium (SS is the Sum of Squares, \(d.f\) is the Degrees of Freedom, MS is the Mean Square, \(F\) is the value of the F-statistic (ratio of the mean squares), and \(p\) is the \(p\)-value for the null hypothesis.](image)

### 3.1.5. Perceptual Linearization of the Method

Since the levels of perceived temporal noise are similar for different simulated media, we may perceptually linearize our method using only one non-linear transform. To that end we plot measured temporal noise levels \(l\) against the real values of low-pass filter strength parameter \(f\) to fit a function \(l = f(l);\) see Fig. 7. We choose \(f\) to be a logarithmic function in the form \(l = a \times \ln(f) + b\), where \(a,b\) are optimization constant coefficients. Such a function was selected because it is nondecreasing, invertible, and it resembles the Weber-Fechner law well known in perception [Pa99]. We fit the parameters to our data using trust region based nonlinear least squares solver [MS83]. For the overall temporal noise levels we obtain the following values: \(a = 4.832, b = -0.221 (R^2 = 0.978)\). Inverse function \(f^{-1}\) is then as follows: \(f' = 0.955e^{l/4.832}\), where \(f'\) is a new value of the filter strength \(f\). In other words, this function is used to transform linear values of filter strength \(f\) to achieve perceptually linear temporal noise response.
### Appendix D. Color Me Noisy Supplementary

/ Color Me Noisy (supplementary materials)

**Figure 3:** Results of the subjective experiment on perception of temporal noise in hand-colored animations. In the reading order: overall results (standard z-scores) and results per stimulus (crayon, chalk, pencil, watercolor animation).

**Figure 4:** Normalized total power of visual channels for 8 different settings of the low-pass filter strength $f$. The bigger the $f$, the more information is processed by the temporal mechanism and accordingly more temporal noise is perceived. In the reading order: results per stimulus (crayon, chalk, pencil, watercolor animation).

**Figure 7:** Logarithmic approximation curves fitted to perceived temporal noise levels per medium and to overall data.

#### 3.1.6. Conclusion

This study has determined that the low-pass filter strength $f$ is positively correlated to the perceived amount of temporal noise. Setting the $f$ level to a higher value will thus generate perceptually more noisy sequences. Our experiment did not show any statistically significant variation of the amount of perceived noise due to the simulated medium.

#### 3.2. Visual Discomfort

We claim that watching a hand-colored animation causes visual discomfort or eye strain to the viewer. Furthermore, we claim that the greater the temporal noise, the greater the eye strain. To test this claim, we designed a subjective experiment similar to the protocol for measuring eye strain in stereoscopic displays by IJsselsteijn et al. [IDR00] and adapted from the “Methodology for the subjective assessment of the quality of television pictures” by the International Telecommunications Union, an international standards body [IR12].

#### 3.2.1. Experimental Setup

The observers were presented with a sequence of 8 videos of varying noise levels in random order. The videos in the sequence were shown for three seconds each, followed by a one second break, to give the observer a frame of reference. The videos were subsequently shown again, in the same order. The second time a video was displayed, a question about eye strain appeared after three seconds while the video looped indefinitely; upon answering the question, the next video in the sequence appeared. The videos used were hand-colored animations of static images, to eliminate any effect of the choice of animation dynamics on the result. There were four different video sequences, each simulating a different medium: crayon, chalk, pencil, and watercolor.

#### 3.2.2. Design and Procedure

The experiment was implemented as a web page for online testing. The evaluated sequences were presented on uncalibrated displays and the observing distance was not controlled. 50 observers were recruited for each sequence from Amazon Mechanical Turk. Observers were allowed a maximum of 10 minutes to complete a sequence, a duration chosen to be extremely generous. Observers were introduced to the experiment with the following text: "A sequence of animations will be displayed automatically with a short pause in between. Then the sequence will repeat with a question submitted to Eurographics Symposium on Rendering (2014).
after each animation. Please wear your prescription glasses or contact lenses, if you would normally wear them to work with a computer or to watch a movie. The purpose of this experiment, which will take about 5 minutes to complete, is to measure the quality of animations. During the experiment, you will see a series of animations, with a one-second break in between. You will then see the animations again, in the same order. The second time, you will be asked to evaluate each animation after it plays.”

Observers were asked to “Please rate the degree of eye strain you experienced while watching this video:”

1. Imperceptible, i.e., no eye strain
2. Perceptible, but not annoying
3. Slightly annoying
4. Annoying
5. Very annoying

3.2.3. Study Results

Each sequence was rated by 50 distinct observers. A minority of observers rated multiple sequences, for a total of 159 unique observers across all four sequences in the experiment. Observers spent approximately 2 minutes per sequence on average; the longest any observer spent was 7 minutes and 51 seconds. Observers rated each video on a 5-point Likert scale. Overall results and results for each sequence are shown in Fig. 10. The plots depict the average score and standard error for each noise level.

To evaluate the significance of the results, we apply the two-way ANOVA test comparing visual discomfort to noise level and media. We further compare visual discomfort to noise level within each medium with the one-way ANOVA test. The results of the test are summarized in Figures 8 and 9. The null hypothesis “different noise levels have no effect on perceived eye strain” can clearly be rejected ($p << 0.001$). Tukey’s honestly significant differences [HT87] shows that in all cases the noise levels can be divided into at least three statistically significant groupings.

There is a small but significant effect of medium on perceived eye strain ($p < 0.01$). Tukey’s honestly significant differences reveals two statistically significant groupings: chalk, crayon, watercolor (greater visual discomfort) and watercolor, pencil (lesser visual discomfort). There is no significant interaction effect between noise level and medium.

![Figure 6: Results of three-way ANOVA on perception of temporal noise, testing for effects of the following factors: low-pass filter strength $f$, simulated medium (crayon, chalk, pencil, watercolor), and observer. (SS is the Sum of Squares, $d.f.$ is the Degrees of Freedom, $MS$ is the Mean Square, $F$ is the value of the $F$-statistic (ratio of the mean squares), and $p$ is the $p$-value for the null hypothesis.](image)

![Figure 9: Results of the one-way ANOVA tests comparing visual discomfort to noise level within each medium. SS is the Sum of Squares, $d.f.$ is the Degrees of Freedom, $MS$ is the Mean Square, $F$ is the value of the $F$-statistic (ratio of the mean squares), and $p$ is the $p$-value for the null hypothesis.](image)

3.2.4. Conclusion

This experiment has determined that the noise level is positively correlated with the degree of experienced eye strain. We do not yet know what level of temporal incoherence is acceptable for long-term viewing. We note, however, that even videos with highest temporal noise level were reported as having only “slightly annoying” eye strain on average.

4. Comparison

This section contains further details about the comparison with previous algorithms [LH05, BCK*13]. The results of all of these comparisons are available for viewing in the supplemental video.
4.1. Parallel Controllable Texture Synthesis

Parallel Controllable Texture Synthesis (PCTS) by Lefebvre and Hoppe [LH05] is based on a multi-scale iterative jitter-correct approach and was originally designed to synthesise infinite textures. The authors also presented an extension of the original method, in which they allowed the user explicit control over the per-level jitter scale, effectively allowing them to modify the scale-distribution of randomness in the resulting texture.

For the purposes of comparison, we had both our method and PCTS synthesise a sequence of small squares of texture using varying noise levels (a sampling of these can be seen in Figure 11). In PCTS they were set by changing the synthesis level at which the most jitter occurs, while in our method we set the low-pass filter strength $f$ to produce sequences that would be visually similar. The results show—as was in fact indicated by the authors—that PCTS can produce convincing results, but only for a limited range of noise levels, which are characteristic for the medium. Forcing high jitter on low levels breaks the result beyond the synthesis’ ability to correct, which is why doing so had not been recommended. Furthermore, jitter on very high levels ultimately only results in the overall texture being translated, as the offset-based synthesis does not produce new structure without the jitter at appropriate scales.

4.2. Stylizing Animation by Example

Stylizing Animation by Example by Bénard et al. [BCK*13] is a filtering approach that facilitates the conversion of 3D animation with deep frame information to an apparently hand-drawn video, where the visual style is presented by example and can be further specified by explicitly re-drawing keyframes. Because it is aimed at mimicking animation in general, it seems straightforward to try and use it to produce a hand-colored animation.

To compare with their method, we created a simple 3D model following the animation dynamics of our hand-made sequence, and passed a frame of the latter to the algorithm as the style source, as indicated in Figure 13. The results produced, some of which are shown in Figure 12, are indicative of the original method’s heavy focus on coherence. However, the correspondence propagation necessary for the method to function makes it impossible to tune the amount of noise in the animation. The only two possible options—synthesis with coherence and frame-wise independent synthesis—make for almost smooth and somewhat noisy sequences, respectively.

5. Results

As mentioned in the main paper we created a simple animation of a striped snake, printed the outlines of the snake in individual frames as a template and asked artists to color them using various media. We have then rendered the same animation using our method on various noise settings in order to produce sequences for comparison. One noise level alongside the original hand-made sequence is shown in Figures 16 through 20. Whole sequences are shown in the supplementary video. These were then used to generate the power spectra shown in the main paper and in this document (Figures 14 and 15).

As with the chalk (Fig. 17) sequence, already presented in the main paper, spatial power spectra of different noise levels (Fig. 14) are similar for remaining media, nevertheless, the sharpening effect is more pronounced.

Crayon (Fig. 16) also has a notably different spatial power spectrum from the original. This is an artifact of the fine consistency iterations (originally proposed by Wexler et al. [WSI07]) used in our method as well as in [LFB*13] to create the reference frames. It causes synthesized images to look slightly over-smoothed when compared to the original source of drawing medium.

The colored pencil (Fig. 19) and watercolor (Fig. 20) sequences show a higher power for the highest spatial frequencies in the synthesized sequences compared to the original. These values point to a subtle noise on very fine scale, which is not, however, visually disruptive, as can be seen in the video results.

As for the temporal spectra (Fig. 15), all sequences again exhibit a relatively flat distribution, with sequences synthesized using higher strength $f$ of the low-pass filter $h$ having more energy overall. The rate of growth varies slightly per medium. We consider this rate of growth to be one of inherent characteristics of the medium used. Furthermore, the energy level of the original hand-made animation is invariably higher than that of synthesized sequences, which we believe to point to the inherent visual noisiness of the technique of hand-colored animation.

The only sequence that exhibits anomalous behaviour is our failure case—the pencil sequence (see Fig. 18). It is of note that the power spectrum in the spatial domain is signif-
icantly flatter than that of other sequences, giving it a more noise-like look. We can also see in the temporal domain that the spectral functions of sequences with various strength \( f \) exhibit significant overlap, compared to other sequences, and are clustered together, indicating that the setting of the \( f \) parameter does not have much effect there.

6. Applications

Besides the synthesis of hand-colored animations and improving rendering in the TexToons framework [SBCC+11] as presented in the main paper, the proposed method is applicable also to other related problems such as stylization of particle simulations, and painterly rendering of photos and videos. In this section we describe them in more detail.


Because our method can separate animation dynamics from fine-scale appearance, we can apply it to produce stylized particle effects even from very rough simulations. We first simulate a small number of large textured particles to generate the motion dynamics and then use our method to provide the final, consistent, stylized look (see Fig. 21 and the supplementary video).

6.2. Painterly rendering of photos and videos.

A real video or a single photograph can be passed to the algorithm as the reference \( R \) and a real painting as the source \( S \). Our algorithm then produces stylized image \( T \), which emphasises low-frequency content of the original video/photograph while preserving high-frequency details of the source painting (see Fig. 22 and the supplementary video).

References


Figure 8: Results of the two-way ANOVA test comparing visual discomfort to noise level and medium. SS is the Sum of Squares, \( d.f. \) is the Degrees of Freedom, MS is the Mean Square, \( F \) is the value of the \( F \)-statistic (ratio of the mean squares), and \( p \) is the \( p \)-value for the null hypothesis.

Figure 10: Subjective experiment (eye strain)—results of the study on perception of eye strain in hand-colored animations synthesized using our method. In the reading order: overall results, results per drawing medium (crayon, chalk, pencil, and watercolor). The error bar depicts the standard error.

Figure 11: Comparison with Parallel Controllable Texture Synthesis—forcing high jitter on low levels makes it impossible for the correction step of PCTS to consolidate the high frequency content and produces somewhat blurry results.
our approach

Figure 12: Comparison with Stylizing Animation by Example–frame-wise independent synthesis enables the original algorithm to produce noisy sequence but without any ability to control it. Also notice the visible seems between coherent regions.

Figure 14: Spatial power spectra.

Figure 15: Temporal power spectra.
Figure 16: Crayon.
Figure 17: Chalk.
APPENDIX D. COLOR ME NOISY SUPPLEMENTARY

Figure 18: Pencil.

submitted to Eurographics Symposium on Rendering (2014)
Figure 19: Colored pencil.
Figure 20: Watercolor.
Figure 21: Smoke stylization—a simple particle-based simulation with textured billboards (left) is used to provide a reference animation with basic motion dynamics that is subsequently re-synthesized using our approach to deliver final stylized look (right).

Figure 22: Painterly rendering of photographs—a reference photo of a parrot (R) is re-synthesized (T) using artistic source (S). Note how T starts to mimic Vincent van Gogh’s painterly style. Spatially variable blur was used to preserve details around the eye.
Supplementary material for the paper
"Brushables: Example-based Edge-aware Directional Texture Painting"

M. Lukáˇc,1 † J. Fišer,1 P. Asente,2 J. Lu,2 E. Shechtman,2 and D. Sýkora1

1CTU in Prague, FEE, 2Adobe Research

1. Direction Analysis

The two alternative approaches to orientation field detection in images [Kyp11, KLC07] differ in their representations of the field, and in the methods used to smooth it. Kyprianidis uses a multi-scale filtering approach, whereas Kang employs a multi-lateral filter of variable support on a vector field. We find that these two approaches actually differ in effect. Using a larger filter suppresses local variance, whereas smoothing on a higher scale suppresses high-frequency detail, even if it is coherent. Because both of these effects may be desirable in different scenarios, we let the user choose which to employ.

Figure 1 shows an example where, due to coherent high-frequency detail, the result of the orientation analysis may not accurately reflect the orientation as perceived by the user. The orientation of the main fabric appears vertical, but a closer examination shows that the predominant edge direction at the pixel level is diagonal. This case can not be addressed by increasing the width of the ETF filter, as that would merely lead to a more coherent slanted field. Instead, the user can apply a pre-smoothing Gaussian filter to remove the high frequency detail before input gradients are calculated, making the more visible vertical lines dominant.

Conversely, some input sources, like the fur in Figure 2, have a lot of local variation in their tangents. For these, the user is faced with a choice: should the directionality of the synthesized result precisely follow the painted directionality, or should the local source variation be preserved and propagated to the result? By letting the user tune the spatial extent of Kang et al.’s [KLC07] filter, we give the user this control.

There are cases where the detected direction does not fit the semantics desired by the user. The fabric sample in Figure 1 shows an example: the local direction on the side fringe is horizontal, but semantically, these are the sides of the sample, requiring a vertical direction. We provide the option to use the same direction brush that is employed to the painting phase to alter the direction of the such regions. In such cases,

Figure 1: The effect of the pre-smoothing Gaussian filter on the orientation analysis: The source image S (a) appears to have a vertical directionality but the detail (b) shows high-frequency diagonal edges. Result (c) shows the direction field calculated at full resolution and (d) the field after passing the source through a 4 pixel wide Gaussian filter. Source credit: denim: inxti @ shutterstock
the tangents of the pixels in the area affected by the brush are flipped to whichever direction follows the suggested direction more closely, and the harmonization pass starts at the boundary of this constrained area. Once the rest of the pixels are harmonized, the tangent field is still filtered normally.

2. Result Details

Figures below display the results of our method more closely along with their direction fields $d_s$. Also shown are the direction fields $d_s$ and the sources we used to produce these.

References


Figure 3: The detailed comparison against Painting by Feature [LFB'13].
Top row: the input image and its detected direction.
Second row, left to right: source with overlaid feature picks; output paths and areas.
Third row, left to right: source segmentation (smoothed from ground truth); output direction field.
The comparison below shows the same images as the figure in the main paper. Source credit: grass: varuna @ shutterstock
Figure 4: Input textures (odd columns) with their detected direction fields $d_i$ (even columns). Source credits: braided wig: Karina Bakalyan @ shutterstock; plank: My Life Graphic @ shutterstock; ivy leaves: Michael & Christa Richert @ rgbstock; red wig: Lenor Ko @ shutterstock; crochet: anneheathen @ flickr; denim: inxti @ shutterstock; cookie: Alessandro Paiva @ rgbstock; bread: Giles Hodges @ DeviantArt; grass: varuna @ shutterstock.
Figure 5: Selected results (odd columns) along with their direction fields $d_t$ (even columns)
Figure 6: Selected results (odd columns) along with their direction fields $d_t$ (even columns) contd.
Figure 7: Selected results (odd columns) along with their direction fields $d_t$ (even columns) contd.
Figure 8: Selected results (odd columns) along with their direction fields $d_l$ (even columns) contd.