Stylisation temporellement cohérente d'animations 3D basée sur des textures

Pierre Bénard

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques et Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Pierre Bénard

Thèse dirigée par **François Sillion**
et codirigée par **Joëlle Thollot**

préparée au sein du **Laboratoire Jean Kuntzmann**
dans **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Stylisation temporellement cohérente d'animations 3D basée sur des textures

Temporally Coherent Stylization of 3D Animations based on Textures

Thèse soutenue publiquement le **7 juillet 2011**,
devant le jury composé de :

Marie-Paule Cani
Professeur à l'université de Grenoble, Présidente

Bernard Péroche
Professeur à l'université Claude Bernard Lyon 1, Rapporteur

Aaron Hertzmann
Professeur associé à l'université de Toronto, Rapporteur

Adam Finkelstein
Professeur associé à l'université de Princeton, Examinateur

François Sillion
Directeur de Recherche à l'INRIA, Directeur de thèse

Joëlle Thollot
Professeur à l'université de Grenoble, Co-Directeur de thèse

RÉSUMÉ

Cette thèse s'inscrit dans le thème du rendu expressif qui vise à définir des outils de création et de traitement d'images ou d'animations stylisées. Les applications concernent tous les métiers nécessitant une représentation visuelle stylisée : création artistique (jeux vidéo, film d'animation, dessins animés), restitution archéologique, documentation technique, etc.

Outre les qualités et l'expressivité du style, un critère fondamental de qualité d'une image est l'absence d'artefacts visuels. Cette considération a toujours existé, mais elle est particulièrement importante dans le cas de l'informatique graphique. En effet, la nature même de l'image – des pixels discrets – est source d'artefacts. Les artefacts sont encore plus visibles lorsque l'on s'intéresse aux animations, des artefacts temporels s'ajoutant aux artefacts spatiaux. Définir précisément ces artefacts est complexe car certains imperfections d'un style peuvent aussi être désirables et faire partie de ses qualités.

L'objectif de cette thèse est d'une part d'analyser et d'évaluer perceptuellement ces artefacts, et d'autre part de proposer de nouvelles méthodes de stylisation d'animations 3D temps-réel.

Nous présentons tout d'abord un ensemble de techniques pour créer et assurer la cohérence de dessins au trait extraits de scènes 3D animées. Nous proposons ensuite deux méthodes de stylisation des régions de couleur permettant la créations d'un grand nombre de motifs. Le point commun à toutes ces approches est la représentation du médium simulé (pigment d'aquarelle, coup de crayon ou de pinceau...) par une texture évoluant au cours de l'animation. Nous décrivons enfin deux expériences utilisateurs visant à évaluer perceptuellement la qualité des résultats produits par ce type de techniques.

Mots clés : Rendu Non-Photoréaliste, Cohérence Temporelle, Textures.

ABSTRACT

This PhD thesis deals with non-photorealistic rendering, a sub-field of computer graphics which aims at defining creation and processing tools to stylize images and animations. It has applications in all the fields that need stylized depictions, such as entertainment (e. g., video games, animated films, cartoons), virtual heritage, technical illustration, etc.

Besides quality and expression of style, a crucial criterion to assert the quality of an image is the absence of visual artifacts. While already true for traditional art, this consideration is especially important in computer graphics. Indeed the intrinsic discrete nature of an image can lead to artifacts. This is even more noticeable during animations, as temporal artifacts are added to spatial ones. Precisely defining these artifacts is complex as certain flaws of a given style may be part of its unique and desirable quality (e. g., the imperfections in a hand-made work).

The goal of this thesis is twofold: (1) To analyse and perceptually evaluate these artifacts; (2) To propose new methods for stylizing real-time 3D animations.

First we present a set of techniques to ensure the coherence of line drawings extracted form 3D animated scenes. Then we propose two methods to stylize shaded regions, which allow to create a wide variety of patterns. The shared ground layer of all these approaches is the use of temporally varying textures to represent the simulated media (e. g., watercolor pigments, brush strokes). Finally we describe two user studies aiming at evaluating the quality of the results produced by such techniques.

Keywords: Non-Photorealistic Rendering, Temporal Coherence, Textures.

REMERCIEMENTS

La thèse est une aventure certes personnelle mais jamais solitaire, j'aimerais donc remercier tous ceux qui m'ont apporté leur aide et leur soutien durant ces trois dernières années. Merci en premier lieu à mes directeurs de thèse, Joëlle Thollot et François Sillion, pour leur encadrement, écoute, conseils et même amitié. Il est difficile de rêver d'un meilleur tandem pour être accompagné lors de cette formation à la recherche.

Je voudrais ensuite remercier l'équipe Artis pour sa cohésion, son atmosphère tant de travail que de franche camaraderie. Cela a été un immense plaisir que de travailler voire, il faut bien l'admettre, de vivre en son sein. Merci en particulier à mes « grandes soeurs et grands frères de thèse » : Adrien, Alex, Hed, David et Pascal. Vous m'aviez donné envie de me lancer dans cette aventure, et nos discussions n'ont ensuite eu de cesse que d'entretenir cette passion. Si je ne peux pas citer tous les membres de l'équipe, je tiens néanmoins à remercier explicitement mes collègues, et amis, de bureau et/ou de ~~voyages~~ conférences : Pierre, Thierry, Laurence, Cyril, Olivier, Charles, Nassim, Laurent et Kartic.

Au-delà de l'équipe, j'ai eu la chance et le privilège de travailler avec de très nombreuses personnes que je souhaiterais également remercier. Merci tout d'abord à Ares Lagae, Peter Vangorp, Sylvain Lefebvre et George Drettakis pour m'avoir faire partager non seulement le stress d'une première *deadline* Siggraph, mais surtout les bières de Saarbrücken.

I switch to English to thank Adam Finkelstein for the fantastic opportunities he opened up to me by welcoming me in Princeton. I would like to acknowledge all the members of the Princeton Graphics Group, with special thanks for Cynthia Lu and Martin Fuchs, which have made my stay at Princeton a pleasant and unforgettable experience. Last but not least, I thank Forrester Cole: it was a great pleasure and very enlightening to work with you.

I would also like to acknowledge Aaron Hertzmann and Bernard Péroche for agreeing to evaluate my dissertation. In addition, I thank Marie-Paule Cani and Adam Finkelstein for being part of this committee and attending my defense. I am glad you came in Grenoble and gave me such insightful comments.

Si mon existence ne s'est (heureusement) pas limitée à ma thèse, il faut en remercier mes amis : non seulement les anciens CPPistes et Ensimagiens pour nos dîners réguliers, mais aussi mes « petits du théâtre » – Adrien, Amélie, Valentine, Victor, Agnès, Chloé, Mathieu, Caroline, Pauline, Dimitri, Anne et Antoine – qui m'ont laissé les ~~materner~~ diriger.

Enfin, je tiens à remercier mes parents, mes grands-parents et ma soeur pour leurs constants encouragements, leur intérêt pour mon travail, et leur compréhension lors de mes, parfois bien longues, périodes de travail acharné.

PUBLICATIONS

The work presented in this manuscript appeared previously in the following publications:

[BBT08] BÉNARD P., BOUSSEAU A., THOLLOT J.: Textures volumiques auto-zoomables pour une stylisation temporellement cohérente en temps réel. In *AFIG '08 (Actes des 21èmes journées de l'AFIG)* (Nov. 2008). Best paper award.

[BBT09] BÉNARD P., BOUSSEAU A., THOLLOT J.: Dynamic solid textures for real-time coherent stylization. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games - I3D '09* (New York, New York, USA, 2009), ACM Press, p. 121. Best presentation award, 10.1145/1507149.1507169.

[BCGF10] BÉNARD P., COLE F., GOLOVINSKIY A., FINKELSTEIN A.: Self-similar texture for coherent line stylization. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering - NPAR '10* (New York, New York, USA, 2010), ACM Press, p. 91. Runner-up best paper award, 10.1145/1809939.1809950.

[BLC*10] BÉNARD P., LU C., COLE F., FINKELSTEIN A., THOLLOT J.: Contours actifs pour la stylisation cohérente de lignes animées. In *AFIG '10 (Actes des 23èmes journées de l'AFIG)* (2010). Best paper award.

[BLV*10a] BÉNARD P., LAGAE A., VANGORP P., LEFEBVRE S., DRETTAKIS G., THOLLOT J.: A Dynamic Noise Primitive for Coherent Stylization. *Computer Graphics Forum 29*, 4 (Aug. 2010), 1497–1506. 10.1111/j.1467-8659.2010.01747.x.

[BLV*10b] BÉNARD P., LAGAE A., VANGORP P., LEFEBVRE S., DRETTAKIS G., THOLLOT J.: NPR Gabor noise for coherent stylization. In *ACM SIGGRAPH 2010 Talks on - SIGGRAPH '10* (New York, New York, USA, 2010), ACM Press, p. 1. 10.1145/1837026.1837079.

[BTS09] BÉNARD P., THOLLOT J., SILLION F.: Quality assessment of fractalized NPR textures. In *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization - APGV '09* (New York, New York, USA, 2009), ACM Press, p. 117. 10.1145/1620993.1621016.

CONTENTS

1

INTRODUCTION

RENDERING consists in generating a picture from a geometric 3D model, or a collection of models organized in a scene, by means of computer algorithms. This process involves three main components: (1) *light transport* – the energy transfer in the scene, (2) *scattering* – the interaction between the light and the models – and (3) a *capture device*. From the beginning of computer graphics in the 60s, the leading quest of rendering has been *photorealism*. Photorealistic rendering implies a physically based simulation of light transport, a physically based definition of scattering functions (measured or phenomenological) and a photographic capture device, usually characterized by a perspective projection and a lens. This approach is motivated by the need to create images that resemble the output of a camera, in particular for special effects in films, and by the fascinating challenge of reproducing reality.

Photography is not in contradiction with expressiveness though. As noted by Durand [Dur02], photographers make artistic decisions during both the shooting (e. g., viewpoint, composition, focal, exposure) and the print (e. g., dodging, burning, masking) to express their personal vision of reality (Fig. 1.1b). Prior to photography, many other pictorial techniques (line drawing, painting, stippling, engraving, etc.) have been and are still used by artists for their *expressiveness* and power of *abstraction*.

These techniques allow to omit unnecessary details while emphasizing important information to guide viewers' eyes. Scientific and technical illustrations are the best examples of fields relying on such a tool. Barla [Bar06, p.6] perfectly sums up this idea: *"while the camera establishes and documents the existence of a subject, the illustrator illuminates its essence"*. They can also suggest uncertainty: architects take advantage of this property by drawing their preliminary sketches with pencils or watercolor. Similarly, why should 3D rendering be reduced to photorealism when digital imagery could open new horizons?

(a) "Grand Teton National Park", Latham Jenkins©

(b) "The Tetons and the Snake River", Ansel Adams©

(c) Watercolor rendering using the approach of Bousseau et al. [BKTS06]

Figure 1.1: Two photographs of a realistic subject (a,b) can have a drastically different mood thanks to artistic manipulations. Non-photorealistic rendering techniques (c) open an even broader realm of possibility.

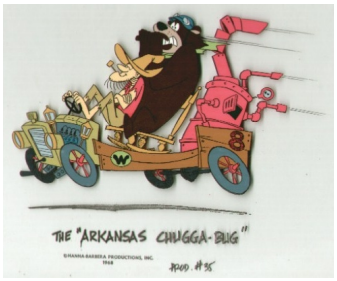
1. Non-Photorealistic Rendering

Led by this question, *Non-Photorealistic Rendering* (NPR) emerged in the 90s and proposed alternatives at each stage of the rendering process. For instance, the visibility term of the light transport can be altered to allow artistic control over shadows [DCFR07]. Scattering functions can be modified to better emphasize shape [VPB*09] or create completely new shading models [GSG*99, BTM06]. The projection can be distorted [Rad99] and the camera itself can be replaced by pre-existent or fully innovative styles. These manipulations aim at giving users a much broader range of possibilities to convey their impression of the scene, real or imaginary (Fig. 1.1c). We will refer to them as *stylization*.

Stylization. Extending Willats model [Wil97], Durand [Dur02] proposes to decompose depiction into four systems: the *mark* system (e. g., brush or pen strokes, watercolor pigments) which implements scene *primitives* (points, lines, regions) at a given *spatial* location with *attributes* (e. g., color, thickness, texture). The choices made by the user for each of these systems define the final style of the picture. Let us take two examples. The artist can decide to represent 3D silhouettes (*primitive*) projected using a linear perspective projection (*spatial*) with black (*attribute*) dotted lines (*mark*). Alternatively he can choose to depict shaded regions (*primitive*) orthogonally projected (*spatial*) with hatches (*mark*) whose thickness (*attribute*) varies according to shading.

In non-photorealistic rendering, the mark system represents not only the *medium* of the picture in its literal meaning (canvas, paper) but also the *pattern* produced by the drawing tool (strokes, hatches, stipples, etc.). Computer generated media and patterns will be the main focus of this thesis.

Although stylization is inherent to traditional drawing or painting, stylized computer depiction has many advantages for novice as well as professional digital artists. It helps casual users to produce visually complex pictures with a limited



(a) Cel from "The Wacky Racers", Hanna-Barbera©

(b) "My Neigbor Totoro", Hayao Miyazaki©

(c) Alexander Petrov's paint-on-glass technique

(d) "L'homme sans ombre", Georges Schwizgebel©

Figure 1.2: Examples of 2D animation techniques.

(a) "Oui-Oui 3D"© (b) "Mickey Mouse Club House", Disney© (c) "Iron Man: Armored Adventures", Marvel©

Figure 1.3: Examples of 3D cartoon series.

amount of time and "technical" skills – the whole or some parts of the stylization process being automatized. It allows to reproduce the same operations multiple times over various input models, as well as to progressively refine the definition of a given style by "trial and error" on the same scene. Finally, it makes possible interactive stylization of animations while allowing a certain amount of control through time. This last property is almost impossible to achieve with traditional media.

2D animations give to the viewer the illusion of a continuous motion from a sequence of still drawings. It makes a huge difference with video recording or 3D rendering. Both of them sample a continuous process at a sufficient framerate to make the discontinuities between snapshots indistinguishable. Conversely, each frame of a hand-made animation is drawn independently. Controlling the evolution of the medium throughout the animation is then a tedious process which is even impossible in many cases. For example, watercolor pigments will inevitably fall at different places between two frames producing *flickering* or *jittering*. Brush strokes will appear and disappear due to occlusions and disocclusions creating *popping*. These effects are often called *temporal incoherences* in the NPR literature.

Traditional artists have few solutions to overcome these incoherences. They can use multiple layers (multi-planing) to separate moving foreground objects or characters from static backgrounds. Cartoon animations are usually drawn on transparent celluloid sheets, called *cels* (Fig. 1.2a). Homogeneous media (e. g., smooth color gradients with thin outline) are often used for the animated parts of the scene because they are less prone to temporal artifacts. Backgrounds can be depicted with richer styles (Fig. 1.2b). In a similar way, Alexander Petrov uses glass planes (Fig. 1.2c) to paint foregrounds and backgrounds on different layer with slow-drying pastel oil. Doing so, he does not have to redraw the full picture at each frame. To reduce the flickering of the brush strokes further, he uses multiple exposures: frame t being a multiple exposure of frame $t-1$ and $t+1$. He still took two and a half years to create the 20 mn of "The old man and the sea".

The last solution is to accept the limitations of the medium, and to make temporal incoherences part of the style. For example, Georges Schwizgebel adds explicit jitters to the paint strokes of his animations to visually mask the temporal inconsistencies that would inevitably occur during wide camera trajectories (Fig. 1.2d).

(a) “Prince of Persia”, Ubisoft©
(b) “Borderland”, Gearbox software©
(c) “Okami”, Clover Studios©
(d) “Love”, Eskil Steenberg©

Figure 1.4: Examples of video games which are using non-photorealistic rendering techniques.

3D animation. 2D hand-made approaches are not always feasible, in particular for cartoon series. Time and budget constraints are high, and the rendering process needs to be as automatized as possible. It explains the current success of 3D modeling, animation and rendering approaches. Characters and objects only need to be modeled once, with various blend shapes or animation skeletons, and they can be reused many times. A large part of the animation and lighting process can be scripted to simplify and speed up the work of the artist. As a side effect cartoon series tend to use basic rendering techniques and to share a rather similar 3D look despite differences in terms of animation and modeling (Fig. 1.3a,b).

Nevertheless, some of them are using different rendering techniques, like toon shading with black outlines, (Fig. 1.3c). It allows to recover a cartoonish aspect, but more complex media are still uncommon. It comes from the fact that, even with computer depiction, the two naïve solutions for stylizing animations are not satisfactory. The first one is *texture mapping*. It consists in pasting up the medium directly on the surface of 3D models using classical texturing. The medium perfectly follows the motion of the objects – avoiding popping or jittering artifacts – but its 2D nature is broken. Traditional artists are usually not drawing or painting on the surface of 3D objects. Instead, they are depicting 3D primitives of the object with 2D marks on a flat paper or canvas. The other naïve solution covers the full picture with a static medium which does not follow the motion of the scene (like a photographic filter on a video camera). The 2D appearance of the medium is well preserved, but this approach breaks the connection between the mark and primitive systems which produces distracting *sliding* effects. Popping, flickering, jittering, 3D appearance and sliding are the various artifacts encompassed by *the temporal coherence problem* in non-photorealistic rendering (Sec. 2).

The video game industry is also interested in non-photorealistic depiction. Besides toon shading – extensively used since 2000 – some studios have attempted to develop more complex styles combining non-photorealistic illumination, outlines and painted textures (Fig. 1.4a-c). Stylization can not only help these games to

stand out among the crowd, but it can also improve the immersion of the player, in particular on computationally limited devices (e. g., Wii, PSP). Similarly to the *Uncanny Valley* introduced by Mashiro Mori for humanoids robots, the more realistic the rendering is, the higher are the player's expectations in terms of graphics but also motion and behavior of the virtual characters. With stylized renderings, these constraints can be relaxed.

The styles chosen by these video games are rather simple though. Eskil Steenberg's MMO "Love" has a much more abstracted Pointillism-like look (Fig. 1.4d). Unfortunately, this game exhibits temporal incoherences (sliding and popping) which can be very distracting. These examples show, once more, that the problem of temporal coherent stylization is crucial in NPR.

2. The Temporal Coherence Problem

Our definition of the temporal coherence problem in non-photorealistic rendering encompasses both spatial and temporal aspects of the stylization. More precisely, temporal coherence implies the concurrent fulfillment of three goals: *flatness*, *motion coherence* and *temporal continuity*. They are represented in Fig. 1.5 as three axes of a radar chart whose scales are purely qualitative. In Fig. 1.5a, we show the theoretical ideal solution: an equilateral triangle fully covering the axes, i. e., perfectly fulfilling the three goals. The subsequent subfigures (Fig. 1.5b-d) show the naïve solutions which completely neglect one of them and produce the opposite artifacts.

Flatness gives the impression that the image is drawn on a flat canvas rather than painted over the 3D objects of the scene [Mei96]. Flatness is a key ingredient in generating computer animations that appear similar to traditional hand-drawn animations. Several properties of the marks must be preserved to produce such a 2D appearance. In particular the size and distribution of marks should be independent of the underlying geometry of the scene. As a typical example, the size of the marks should not increase during a zoom.

Motion coherence is the concordance between the apparent motion flow of the 3D scene and the motion of the marks. As stated by Cunzi et al. [CTP*03], the goal is to provide in 2D screen space a perceptual impression of motion as close as possible to the 3D displacement in object space. A low concordance produces sliding artifacts and gives the impression that the scene is observed through a semi-transparent layer of marks, also referred to as the *shower door effect* [Mei96]. Compared with 2D animation, 3D rendering allows dramatic camera motions – complex rotations and extended zoom – that require special care.

Temporal continuity is the quality of minimizing changes from frame to frame to ensure fluid animations. Perceptual studies [YJ84, SS09] have shown that human observers are very sensitive to sudden temporal incoherence such as popping. The visibility and attributes of the marks should vary smoothly to ensure temporal continuity and fluid animations. This is especially important when zooming and at occlusion and disocclusion boundaries.

Unfortunately, for complex media, these goals are inherently conflicting and naïve solutions often neglect one or more criteria. While texture mapping can be used to apply the marks over the scene with high motion coherence and temporal

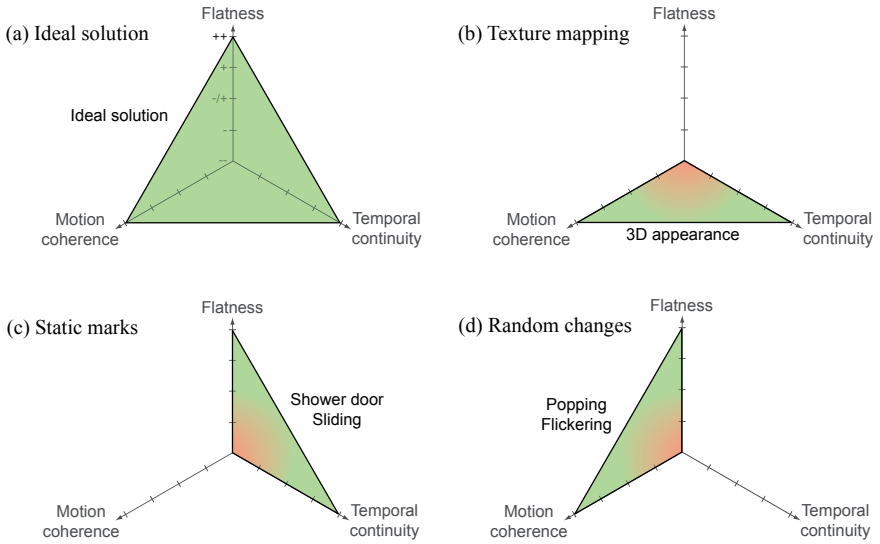


Figure 1.5: The temporal coherence problem involves three goals represented by the axes of these diagrams. Fully ignoring one of them produces the opposite artifacts (b,c,d).

continuity (Fig. 1.5b), perspective projection makes the size of the marks vary with depth which breaks the impression of flatness. Keeping the marks static from frame to frame (Fig. 1.5c) ensures flatness and temporal continuity but produces a strong shower door effect since the motion of the marks has no correlation with the motion of the scene. Finally, processing each frame independently as in hand-drawn animation leads to strong flickering and popping since the position of the marks varies randomly from frame to frame (Fig. 1.5d).

Perfectly respecting the 2D aspect of any medium while accurately following any 3D motion without any temporal discontinuity seems impossible. Consequently, the key problem is to find a "good" tradeoff between these goals, which is style, application and artist dependent. It raises a number of questions that we will try to tackle in this manuscript: How to evaluate the trade-offs produced by such methods? On which goal can we compromise with the least negative impact on the final result for a given style? How to explore the spectrum of solutions delimited by these goals?

3. Contributions

The thesis presented in this manuscript is twofold. We defend that *texture* is an appropriate representation for many media ranging from stochastic (e. g., watercolor pigments) to structured pattern (e. g., cross-hatches). There are no commonly accepted definition of texture; ours follows from Tamura et al. [TMY78]. We define it as an image region whose "structure is simply attributed to repetitive patterns in which elements or primitives [in our case, marks] are arranged according to placement rules" (potentially random ones).

We also claim that textures allow to design stylization techniques which provide well-balanced trade-offs between the three goals of flatness, motion coherence and temporal continuity.

We will demonstrate the validity of these statements for two primitives: lines and shaded regions extracted from animated 3D models.

I. Line stylization. For animated lines drawings, we propose a new texture pyramid that we call Self-Similar Line Artmap (Chapter 3) which allows to zoom infinitely on a textured line without distortion of its 2D appearance. We describe a method to automatically synthesize such a texture pyramid from a single exemplar. We then present two temporally coherent parameterization schemes for view-dependent lines. The simplest one (Chapter 4) targets smooth shapes and propagates the line parameterization from one frame to the next by a robust voting mechanism. The second method (Chapter 5) handles the case of complex models using an explicit and persistent 2D representation of the extracted lines. This method permits a higher amount of stylization and abstraction than any previous approach.

II. Region stylization. In the second part of this manuscript, we propose two trade-offs for temporally coherent stylization of shaded regions. The first one (Chapter 6) relies on texture mapping to ensure motion coherence. Nevertheless, it extends previous 2D texture fractalization algorithms to solid textures in order to better fulfill the goal of flatness. Targeting real-time applications, this method fits seamlessly in any modern game engine. The second approach (Chapter 7) is based on a noise function defined on the surface of the models but evaluated in 2D to perfectly preserve the characteristics of the medium. Due to its procedural definition, this method avoids the texture mapping problem of the first approach, ensuring the goal of flatness at best. We show the broad range of styles that can be created with these two methods in Chapter 8.

III. Evaluation. We finally discuss the complex question of the evaluation of such techniques (Chapter 9) and present two perceptual studies we conducted in the context of shaded regions. The first user-study aims at evaluating fractalized media. We derive from its results an objective metric for automatic quality assessment (Chapter 10). In a second study (Chapter 11), we compare six region stylization methods – including ours – based on the three goals involved in the temporal coherence problem. This study not only demonstrates the effectiveness of the solutions we proposed, but also opens new perspectives for future research that will be discussed in conclusion (Chapter 12).

2

RELATED WORK

THIS CHAPTER reviews the main contributions in non-photorealistic rendering to the question of the temporally coherent stylization of 3D animations. We refer the reader to the books of Gooch and Gooch [GG01] and of Strothotte and Schlechtweg [SS02] for a more general survey of non-photorealistic algorithms, in particular for static images.

Sec. 1 focuses on line drawings while Sec. 2 concentrates on shaded regions stylization. These two primitives share a common issue when dealing with 3D animations: the problematic case of the zoom. We put a special emphasis on this problem throughout the text.

1. Coherent Line Stylization

Line drawings can effectively depict complex information with simple means. In traditional line drawing, artists use ink, pencil or charcoal to draw image discontinuities such as silhouettes, contours or shadow boundaries. Line drawing algorithms often replicate this artistic workflow by first identifying the lines, and then rendering them with a particular medium. Both steps require special consideration to produce temporally coherent animations.

1.1. Extracting Coherent Lines

Many researchers have focused their efforts on the extraction of lines from images and 3D scenes (see the annotated bibliography of Rusinkiewicz et al. [RCDF08]). We classify existing line extraction algorithms as either performed in image space or object space. In each case we review algorithms to extract the lines and methods to build correspondences between lines in multiple frames.

1.1.1. Image Space Lines

Image processing is the simplest and most efficient way to extract lines from 3D scenes, making it particularly suitable for video games [MESA*10]. The seminal work of Saito and Takahashi [ST90] filters depth and normal maps to extract

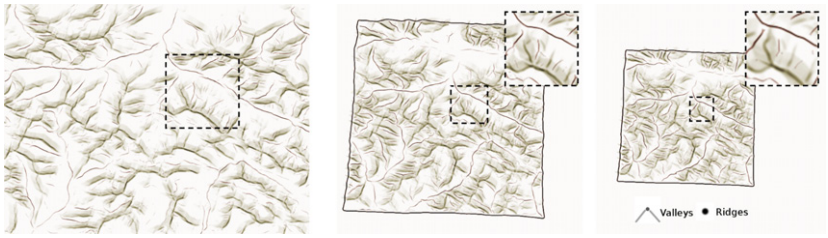


Figure 2.1: Three frames from a zoom on a terrain model rendered with the "Implicit Brushes" of Vergne et al. Surface features exhibit natural LOD and coherence. From [VVC*11].

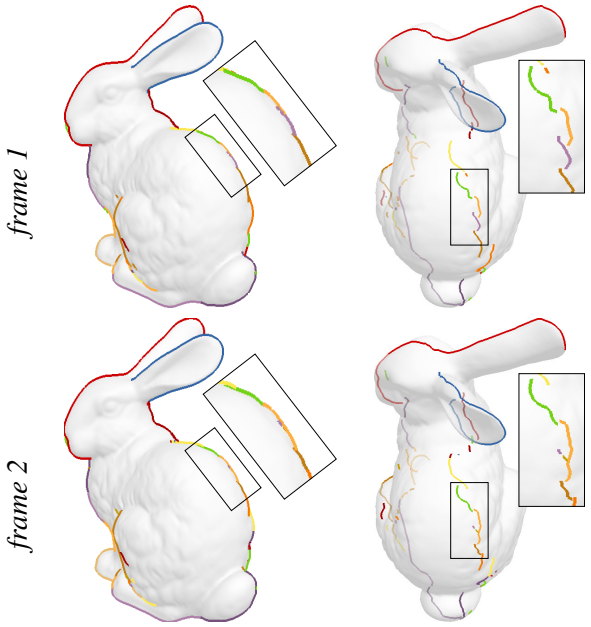
contours and creases. The modern GPU implementation of Nienhaus and Döllner [ND05] uses depth peeling to extract both visible and hidden lines in real-time. These methods produce line drawings with natural level of detail (LOD) based on screen space proximity (Fig. 2.1). LOD improves the readability of the drawing at every scale by preventing clutter.

Lee et al. [LMLH07] propose a more involved filtering technique allowing lines of controllable thickness. They extract luminance features from shaded 3D models using a 2D local fitting approach. Generalizing this method, Vergne et al. [VVC*11] use differential geometry on view-centered buffers (surface gradient and curvature tensor) and cubic polynomial fitting to extract various features and their profile: surface edges, ridges, valleys, inflections, occluding contours and luminance edges. Their method also applies to videos.

Image space methods produce lines made of independent, unconnected pixels, offering neither parameterization nor correspondence across frames. The computer-aided rotoscoping approach of Agarwala et al. [AHSS04] address this limitation by explicitly tracking edges in videos with Bézier curves. These curves can serve as a scaffold to animate brush strokes drawn by the artist. The system is however designed for offline processing and relies on heavy-weight optimization and manual correction.

1.1.2. Object Space Lines

Line drawings can also be generated by extracting feature lines on the surface of 3D models. However, animating such feature lines without explicit control of temporal coherence produces a variety of artifacts such as sliding, popping, and splitting.



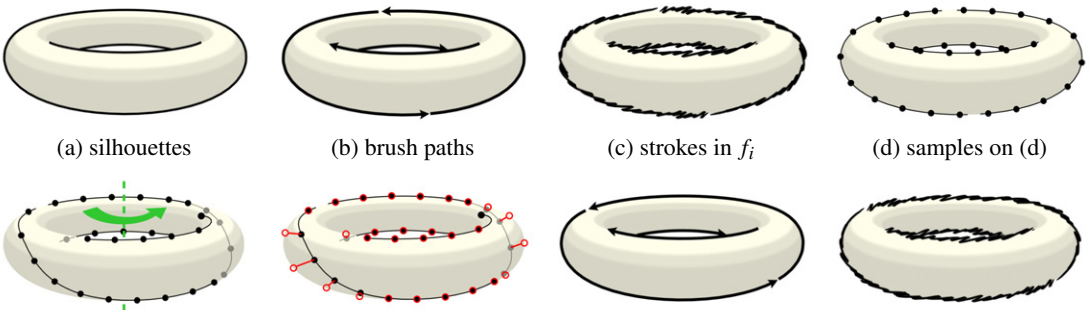
(a) silhouettes in 3D (b) side view of (a)

Figure 2.2: Silhouette coherence. Frame 1: (a) Silhouettes in object space appear to be continuous in the image, but (b) are extracted as many disconnected parts. Frame 2: (a) The silhouette is visually coherent with frame 1, but is (b) composed of a different set of disconnected parts.

Some abrupt temporal changes originate from instabilities in the line extraction with respect to small changes in the viewpoint. DeCarlo et al. [DFR04] propose line trimming and fading policies which greatly improve coherence when animating suggestive contours. They analyze the speed of motion of lines over the surface and discard the ones that are moving too fast.

Visibility. Line visibility computation based on 2D buffers (an *item buffer* [NM00, KMM*02, CDF*06] or a *depth buffer* [IHS02]) can also be responsible for some popping due to aliasing. Algorithms for hidden line removal in object space [EC90, MKG*97, HZ00, GTDS10] avoid this problem at the price of high computational complexity. The *segment atlas* of Cole and Finkelstein [CF10] provides similar quality at interactive frame rates by exploiting graphics hardware.

Connectivity. The connectivity of the extracted lines can be inferred from their 3D structure. For polygonal meshes, the connectivity in-



(a) silhouettes (b) brush paths (c) strokes in f_i (d) samples on (d)

(e) new camera for f_{i+1} (f) propagation (g) new brush paths (h) new strokes in f_{i+1}

Figure 2.3: Overview of Kalnins et al. "Coherent Stylized Silhouettes". The input silhouettes (a) are split into continuous brush paths, parameterized and textured (b-c). This parameterization is sampled and propagated by reprojection in the new camera (d-e) and local search in 2D (f). New coherent brush paths are recovered from the votes (g); their parameterization is optimized to compromise between uniformity in 2D and coherence in 3D. Taken from [KDMF03].

formation between edges can be used to grow paths by linking together visible pieces of lines [NM00, IHS02]. Simple heuristics on distances and angles are defined to solve ambiguities at lines intersection. These linking algorithms may however be unstable under change of viewpoint.

After projection and clipping, feature lines can be parameterized in each frame using image-space arc-length. Such parameterization allows a broad range of styles like textured lines, wiggles or tapering [NM00, IHS02, GVH07, GTDS10] but the lack of correspondence in parameterization from frame to frame prevents temporally coherent animations. View-independent lines such as creases, ridges and valleys can rely on the underlying surface parameterization to ensure such a correspondence. In contrast, view-dependent lines such as silhouettes, suggestive contours, and apparent ridges move over the surface and have different 3D geometry and topology for each viewpoint [RCDF08]. Two main classes of approaches have been proposed to build a correspondence between view-dependent lines: the first class focuses on real-time applications, while the second one targets precomputed animations.

Parameterization by 2D sample propagation. Building on the seminal work of Masuch et al. [MSS98] and Bourdev [Bou98], Kalnins et al. [KDMF03] propose the first complete method to tackle coherent parameterization of feature lines, using the connectivity of smooth silhouettes [HZ00] to propagate parameterization from frame to frame.

The three key ideas of their system are:

1. propagating the line parameterization through samples in image space from one frame to the next to provide temporal continuity (Fig. 2.3(d-f)),
2. splitting continuous paths into possibly multiple brush paths corresponding to the paths used in previous frames (Fig. 2.3(b,g)), and
3. optimizing an energy function that includes the competing goals of uniform image-space arc-length parameterization, coherence on the object surface, and attempting to merge multiple paths where possible.

This approach offers temporally coherent parameterization for feature lines extracted from simple, smooth objects. Unfortunately, models of moderate complex-

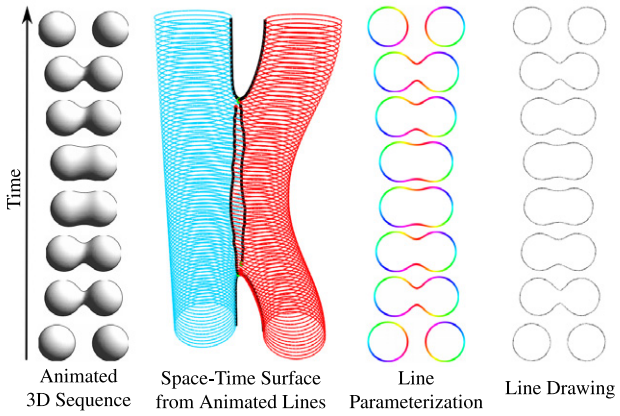


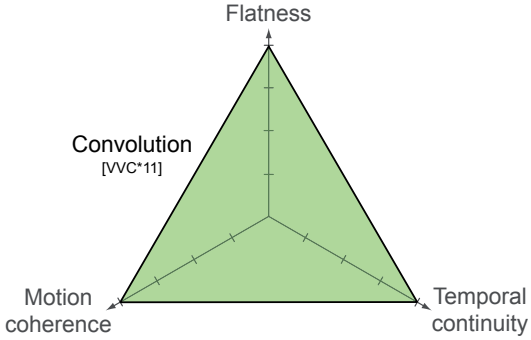
Figure 2.4: A temporally coherent parameterization for line drawings is computed by parameterizing the space-time surface swept by the line over time. From [BFP*11].

ity, for example the Stanford bunny, generate silhouettes made of many tiny fragments (Fig. 2.2).

Karsch and Hart [KH11] use *snaxels* to extract and track visual, shadows and shading contours. Snaxels are closed active contours [KWT88] that minimize an energy on a 3D meshed surface. They define this energy as an implicit contour function corresponding to silhouettes or isophotes. This approach deals robustly with topological events, provides long connected paths and natural correspondences across frames of an animation. These paths can be used as input for the system of Kalnins et al. [KDMF03] to determine a coherent parameterization.

Spatio-temporal formulation. Buchholz et al. [BFP*11] formulate the line correspondence problem as the parameterization of the space-time surface that is swept by the lines during the animation (Fig. 2.4). They propose a robust algorithm to construct this surface, taking into account merging and splitting events of silhouettes. The surface is then parameterized by an optimizer that ensures motion coherence and flatness. Users can control the trade-off between these constraints with few meaningful parameters. The robustness of this approach comes however at the price of expensive computations (several minutes for few seconds of animation).

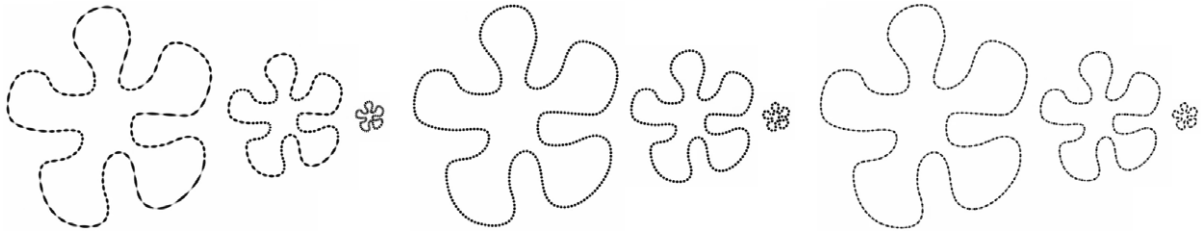
1.2. Line Drawing Rendering



Convolution. Although the lack of parameterization limits the range of styles of image space lines, Vergne et al. [VVC*11] extend the work of Lee et al. [LMLH07] and formulate the mark rendering process as a spatially-varying convolution that mimics the contact of a brush with the line. This implicit style definition ensures flatness (Fig. 2.1) and produces temporally continuous stylized lines from dynamic 3D scenes and videos in real-time.

Texture Mapping. Parameterized lines allow the use of texture mapping to produce dots and dashes or to mimick paint brushes, pencil, ink and other traditional media.

There are two simple policies for texturing a path. The first approach, that we call the *stretching policy* (Fig. 2.5a), stretches or compresses the texture so that it



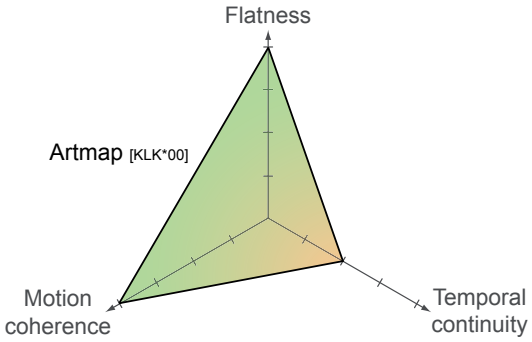
(a) Stretching (b) Tiling (c) Artmap

Figure 2.5: Three strokes texture mapping policies. (a) Stretching ensures coherence during motion but deforms the texture. Conversely (b) tiling perfectly preserves the pattern, but produces sliding when the path is animated. (c) Artmap avoids both problems at the price of fading artifacts because the hand-drawn texture pyramid is usually too sparse.

fits along the path a fixed number of times. As the length of the path changes, the texture deforms to match the new length. The second approach, called the *tiling policy* (Fig. 2.5b), establishes a fixed pixel length for the texture, and tiles the path with as many instances of the texture as can fit. Texture tiles appear or disappear as the length of the path varies.

The two policies are appropriate in different cases. The tiling policy is necessary for textures that should not appear to stretch, such as dotted and dashed lines. Because the tiling policy does not stretch the texture, it is also usually preferred for still images. Under animation, however, the texture appears to slide on or off the ends of the path similarly to the shower door effect (Fig. 2.5b) . In contrast, the stretching policy produces high motion coherence under animation, but the stroke texture loses its character if the path is stretched or shrunk too far (Fig. 2.5a). Kalnins et al. [KMM*02] combine these two policies with 1D texture synthesis using Markov random field to reduce repetitions.

The *artmap* method [KLK*00](Fig. 2.5c) is an alternative to the simple stretching and tiling policies. This method uses texture pyramid, where each texture has a particular target length in pixels. At each frame, the texture with the target length closest to the current path length is selected and drawn. This mechanism ensures that the brush texture never appears stretched by more than a constant factor (often 2×).



Nevertheless, stretching artifacts still appear when the length of the path extends beyond the length of the largest texture in the artmap. Fading or popping artifacts can also occur during transitions between levels of the texture pyramid. Finally, a major drawback of the artmap method resides in the manual construction of the texture pyramid. Artists need to draw each level of the pyramid tacking care of the coherence across levels. As a result, current artmap implementations such as “Styles” in Google SketchUp© use as few as four textures, which accentuates artifacts during transitions.

1.3. How to choose a line extraction and rendering method?

Tab. 2.1 compares the various line extraction techniques we have presented with respect to two key properties: their ability to provide a coherent parameterization

		Coherent Parameterization	Level of Details
Image space lines	Filtering [ST90, ND05, LMLH07, VVC*11]	−−	++
	Rotoscoping [AHSS04]	+	+
Object space lines	Sample propagation [KDMF03]	+	−−
	Vote splatting [BCGF10]	+/−	−−
	Snaxels [KH11]	+	−−
	Spatio-temporal formulation [BFP*11]	++	−−

Table 2.1: Summary of the properties of the different line extraction approaches.

		Flatness	Coherent motion	Temporal continuity
Convolution [VVC*11]		++	++	++
Texture mapping	Tiling	++	−−	++
	Stretching	−−	++	++
	Artmap [KLK*00]	++	++	+/−

Table 2.2: Summary of the trade-offs made by the different line rendering approaches.

and to deal with level of detail. Object space methods provide line parameterization which allows complex rendering effects, in particular stroke texture mapping, but may introduce strong temporal artifacts. Image space lines do not offer easy texturing, but naturally support LOD control.

Tab. 2.2 summarizes the compromise made by line rendering approaches. Note that the temporal continuity of the texture mapping methods highly depends on the continuity of the input parameterization.

2. Coherent Region Stylization

Color regions can be represented in a variety of styles including watercolor, oil painting, cross hatching and stippling. This wide range of appearance has conducted researchers to propose different rendering algorithms adapted to the properties and constraints of each style. We first review methods to extract coherent regions in an animation and then discuss the different families of algorithms to render marks over these regions. We highlight for each type of algorithm the range of styles it can produce.

2.1. Extracting Coherent Color Regions

Maintaining the temporal coherence of stylized color regions often requires the definition of a correspondence between the regions from frame to frame. In the case of 3D scenes, the correspondence can be directly obtained from the scene geometry. In the case of videos, computer vision algorithms can be used to estimate region correspondence from video streams.

Extraction from 3D scenes. Most algorithms dealing with the stylization of 3D scenes assign a unique identifier to each object of the scene. Rendering this

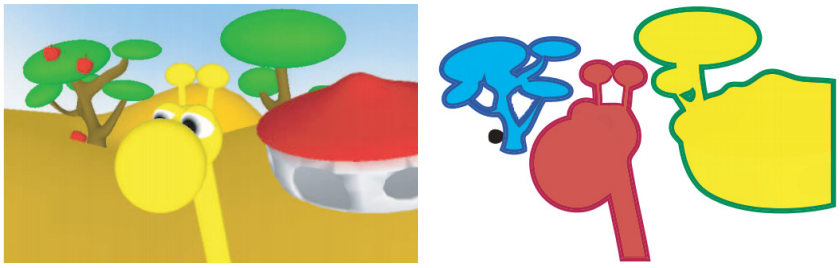


Figure 2.6: Bezerra et al. "3D dynamic grouping". Objects of the input scene (left) are grouped based on their 3D spatial proximity (right). Adapted from [BEDT08].

identifier in an *ID buffer* provides a segmentation of the image that evolves coherently with time. This simple approach can however produce many small color regions when objects are distant from the camera. Kolliopoulos et al. [KWH06] propose to abstract out such small details by grouping pixels using a spectral clustering algorithm. Their method clusters pixels based on their similarity in color, ID, normal and depth. They achieve temporal coherence by linking each frame to its predecessor, so that the segmentation of a frame is biased to be similar to the previous frame. Temporal incoherence may however still occur in case of fast motion or occlusions. Bezerra et al. [BEDT08] address these issues with an object-space grouping method based on the mean-shift clustering algorithm (Fig. 2.6). Their implementation of the mean-shift algorithm offers real-time performances, while the spectral clustering of Kolliopoulos et al. [KWH06] requires a few seconds of processing per frame.

Extraction from videos. Extracting coherent regions from videos remains an active area of research in computer vision and several algorithms from this domain have been applied to the stylization of videos.

The *Video Tooning* system [WXSC04] relies on a spatio-temporal mean-shift algorithm to extract 2D+t volumes from a video. Smoothing this volume simplifies the corresponding color regions and allows the creation of different levels of detail. The extracted volume can also be used to interpolate user-specified brush strokes between key-frames. Note however than an additional spatio-temporal parameterization in the spirit of [BFP*11] would be needed to ensure high temporal coherence. Collomosse et al. [CRH05] describe a similar system that first segments the video on a frame-by-frame basis. The algorithm then links the segmented regions from frame to frame (Fig. 2.7). This frame-by-frame approach requires less memory and reduces over-segmentation of the spatio-temporal volume in case of fast motion. Segmentation-based methods remain however computationally demanding and often need user assistance.

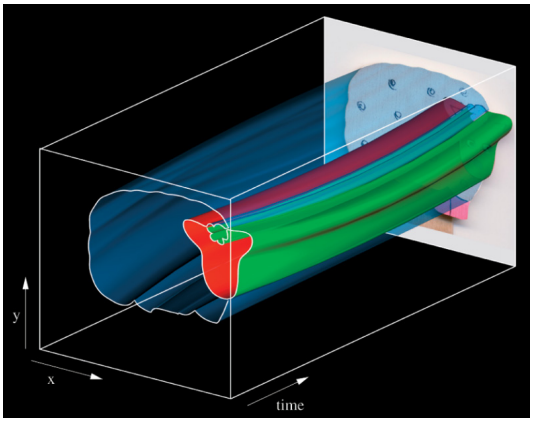


Figure 2.7: Collomosse et al. "Stroke surfaces". Visualization of the spatio-temporal volume extracted from an animation. Intersecting the volume with a time plane generates a temporally coherent segmentation. From [CRH05].

Color regions can also be simplified by means of local image filters, similar in spirit to image space line extration. Winnemöller et al. [WOG06] apply a bilateral filter followed by soft quantization to produce cartoon animations from videos in real time. The soft quantization produces results with higher temporal coherence than hard quantization. Various filters have been proposed for similar purpose, such as the coherence-enhancing filter of Kyprianidis and Kang [KK11]. These filtering approaches are fast to compute as they can be implemented on the GPU.

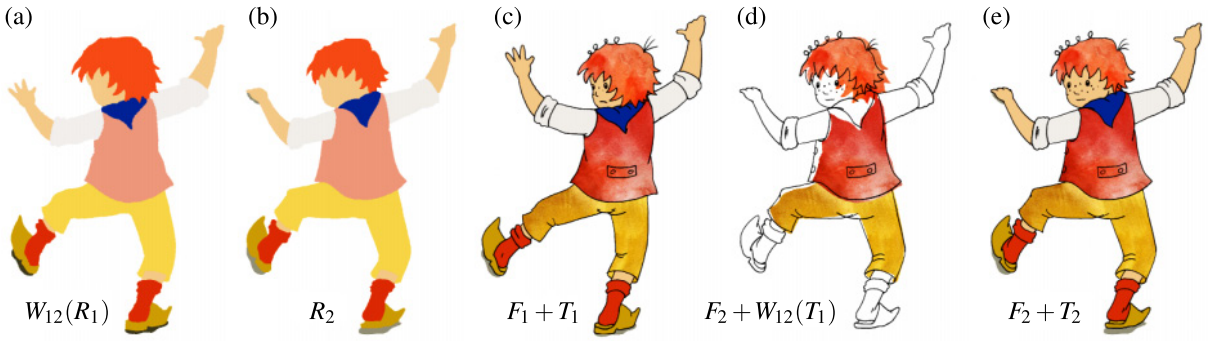


Figure 2.8: Sýkora et al. "TexToons" method. (a) segmentation R_1 deformed by the mapping to the second drawing W_{12}, (b) segmentation R_2, (c) source frame F_1 with textured regions using coordinates T_1, (d) textured regions mapped by deformed texture coordinates $W_{12}(T_1)$, (e) target frame F_2 with textured regions after texture coordinates extrapolation. Adapted from [SBCv*11].

However color regions generated this way are defined implicitly, which prevents more complex stylization effects.

Finally, frame-to-frame correspondence can be specified on a pixel basis using motion estimation. Optical flow [BB95] and related methods have been used to maintain temporal coherence for painterly rendering [Lit97, HP00, HE04, KBD*10], watercolor rendering [BNTS07] and hatching [SZK*06]. Bousseau et al. [BNTS07] also use optical flow to orient a spatio-temporal filter along motion trajectories. Their morphological filter removes small features from the video with limited popping and flickering but are slower than filters that operate in the image domain only.

While optical flow provides per-pixel motion estimates, local errors in the motion field can lead to swimming artifacts in the stylized output. Feature tracking produces sparser but more robust estimates [LZL*10]. The *particle video* algorithm [ST06] produces motion estimates that are both dense and robust, but it has not yet been used for non photorealistic rendering to the best of our knowledge. Complementary to these automatic approaches, user-assisted rotoscoping allows refinement and additional control over the motion estimation [AHSS04, OH11].

For the special case of 2D cartoon animations, Sýkora et al. [SBCv*11] extract dense correspondences between hand-made drawings using as-rigid-as-possible image registration [SDC09]. These correspondences allow the propagation of color and texture coordinates over the animation. In case of disocclusions, the algorithm extrapolates texture coordinate to cover the appearing pixels, which can produce sliding artifacts.

In summary, clustering and segmentation-based approaches explicitly extract color regions that can be used as input for subsequent mark- and texture-based rendering. However, they are usually computationally expensive, and can require user guidance. Conversely, filtering methods perform in real-time with intuitive controls, but they don't provide time coherent parameterization.

2.2. Color Regions Rendering

We classify previous work on coherent region rendering into two main categories: mark-based and texture-based methods. Choosing one or the other has important

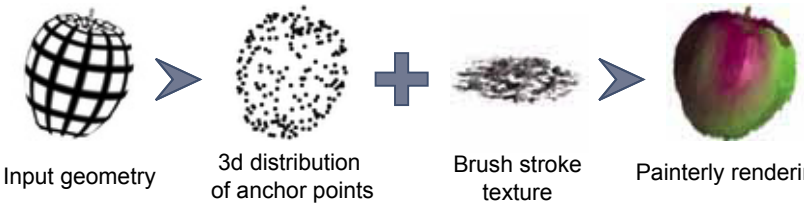


Figure 2.9: Meier's painterly rendering algorithm. The input geometry is sampled into a 3D distribution of anchor points. For each visible sample, a brush stroke texture is drawn in screen space. Adapted from [Mei96].

consequences. The first class of methods tends to compromise temporal continuity, whereas the latter tends to compromise either motion coherence or flatness.

Multiple strategies have been proposed, for both mark- and texture-based approaches, to deal with extended or even infinite zoom. The two contradictory goals of the infinite zoom illusion are to maintain the size of the texture elements as constant as possible in screen space while preserving the enlarging or shrinking of texture elements required for a convincing feeling of zooming in or out.

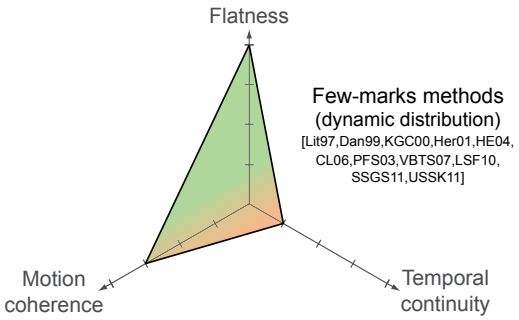
2.2.1. Mark-based Methods

Few-mark methods are mostly used for patterns composed of uniformly distributed features, typically painterly rendering and stippling. See [Her03] for a survey.

Meier [Mei96] introduced the idea of attaching anchor points to the 3D surface and using them to draw overlapping marks in 2D (Fig. 2.9). Since the marks are usually small with respect to object size, their motion remains very close to the original motion field of the 3D scene, providing good motion coherence. Flatness is preserved by drawing the marks as 2D sprites.

In its original algorithm, Meier draws anchor points from a static uniform distribution over the object surface. Such an object space distribution is however non-uniform after projection in the image. Consequently, anchor points can be too dense for objects that are far from the camera, and too sparse for nearby objects. Many subsequent work address the issue of anchor point distribution, providing various trade-offs between flatness and temporal continuity. Methods have been proposed to process 3D scenes [KGC00, CL06, PFS03, VBTS07] and videos [Lit97, Her01, CRL01, HE04, VBTS07].

A first solution proposed by Daniels [Dan99] gives control to the user. In this workflow, artists first draw strokes over the object in a given frame of the animation. The strokes are projected and stored on the 3D surface and back-projected in the subsequent frames. Artists can then add or remove strokes in other frames to fill-in holes and avoid clutter. Schmid et al. [SSGS11] extend this approach by allowing strokes to be embedded anywhere in a 3D implicit canvas surrounding proxy surfaces. These methods provide precise control over the final look of the animation but require significant time and expertise.



Dynamic distributions aim at automating this process by adapting the distribution of anchor points in each frame. The goal is to maintain a uniform spacing between anchor points (Poisson disk distribution) while avoiding sudden appearance or disappearance of marks.

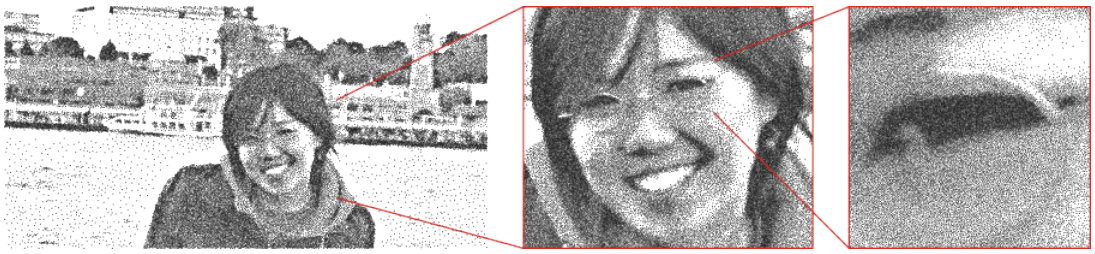


Figure 2.10: "Recursive Wang Tiles" of Kopf et al. While zooming in, more stipples appear to maintain the apparent point density while preserving its blue noise property. From [KCODL06].

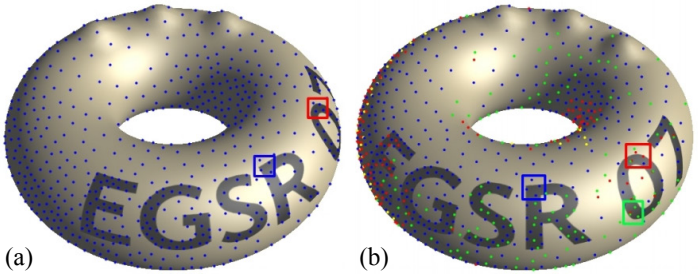
Extending the Poisson disk tiling method of Lagae et al. [LD05], Kopf et al. [KCODL06] propose a set of recursive Wang tiles which allows to generate point distributions with blue noise property in real-time and at arbitrary scale (Fig. 2.10). This subdivision mechanism ensures the continuity of the distribution during the zoom, whereas the recursivity of the scheme enables infinite zoom. However, because they rely on pre-computed tiles, their method can only deal with 2D rigid motions (zooming and panning inside stills).

A dynamic distribution can be precomputed as a hierarchical 3D distribution over the objects surface [CRL01, PFS03, NS04]. At each frame, anchor points are selected as a cut in the hierarchy according to depth and surface orientation. These approaches ensure a good temporal continuity as long as the amount of zoom does not exceed the limit of the hierarchy. Although the dynamic cut prevents the distribution to be too sparse or too dense, the distribution is often not Poisson disk after projection in image space [PFS03].

Vanderhaeghe et al. [VBTS07] propose a hybrid technique which finds a more balanced trade-off. They compute the distribution in 2D – ensuring blue noise property – but move the points according to the 3D motion of the scene by re-projecting the points on the 3D geometry (Fig. 2.11). At each frame, the distribution is updated: new points may be added and old points may be erased based on a Poisson-disk criterion. The temporal continuity is enhanced further by (1) fading appearing and disappearing points over subsequent frames; and (2) allowing points in overly dense regions to slide to close under-sampled regions. Despite these two policies, residual popping is still noticeable, mainly due to occlusions and disocclusions. Their approach also apply to videos using optical flow to move the anchor points.

The data structure required to manage the anchor points and the expensive rendering of each individual stroke makes this family of methods not well-suited for real-time rendering engines. Nevertheless, Lu et al. [LSF10] proposed a full GPU implementation with a simplified stochastic strokes density estimation which runs at interactive framerates but offers less guarantees on the point distribution..

Figure 2.11: (a) Points are distributed in 2D and re-projected on the 3D torus. (b) After being moved to the next frame, points are reprojected to screen-space. Red points are rejected based on a Poisson-disk criterion; blue points are still valid; green points are added; and yellow points are removed due to visibility. Adapted from [VBTS07].



(a) (b)

When dealing with videos, an additional spatio-temporal segmentation step [WXSC04, KWH06, CRH05, LZL*10, KBD*10, OH11] can improve the results. It consists in defining temporally coherent regions (2D + t volumes) which are guiding the stylization. Compared to optical flow, it can reduce the flickering by confining brush strokes inside the segmented regions, and permits a certain amount of abstraction by controlling the size of these regions.

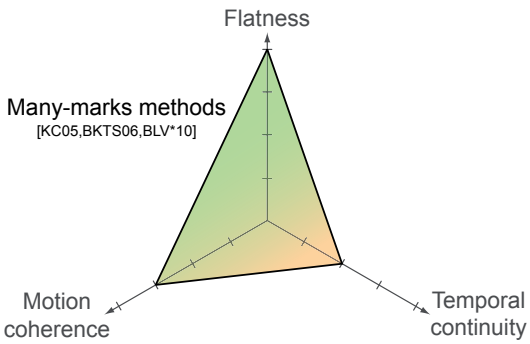
In summary, all these methods offer coherent motion at the price of either a poor flatness due to non-uniform anchor point distribution, or poor temporal continuity due to appearance and disappearance of anchor points. Most methods limit popping by carefully blending new strokes or fading out deleted ones, but such visibility events may still be perceivable.

Many-mark methods. Blending a large number of marks, many-marks methods allow the animation of continuous textures where each individual mark is indistinguishable from its neighbors. These methods often produce less popping artifacts than few-marks methods because each individual mark has much less influence on the final image.

In procedural texturing, sparse convolution noise [Lew84, Lew89], including spot noise [vW91] and Gabor noise [LLDD09], can be classified as many-mark methods. For an overview of procedural noise functions, see Lagae et al. [LLC*10]. These methods allow to create a wide variety of textures using procedural texturing [EMP*02] but do not address the problem of temporal coherence.

In NPR, the only two previous methods that can be classified as many-mark methods are the dynamic canvas of Kaplan and Cohen [KC05] and the interactive watercolor rendering of Bousseau et al. [BKTS06]. These methods take into account the problem of temporal coherence, but are limited to a specific medium: canvas fibers and watercolor respectively. In both cases the connection with sparse convolution noise is not made explicit, even though Bousseau et al. use Perlin noise [Per85].

To deal with the infinite zoom problem, Kaplan and Cohen [KC05] propose a mechanism which subdivides the input triangles if they require more canvas fibers to be covered than the precomputed target. Thanks to the high number of fibers, this produces a rather continuous generation of new details on the fly. Bousseau et al. [BKTS06] follow a similar approach in conjunction with an octree structure over the mesh triangles. It accounts for the case of too small triangles to represent low-frequency octaves of the noise. Small scale popping artifacts might be visible and the performances of their system may drop significantly.



In summary, these approaches bridge the gap between marks-based and texture-based methods by generating a continuous pattern from discrete marks. It allows them to improve temporal continuity while preserving strong flatness and good motion coherence. However they prevent the creation of styles with clearly independent marks. Moreover, because they perfectly preserve the 2D characteristics of the pattern, their results can show some residual popping/flickering, contrast oscillations or second order motion.

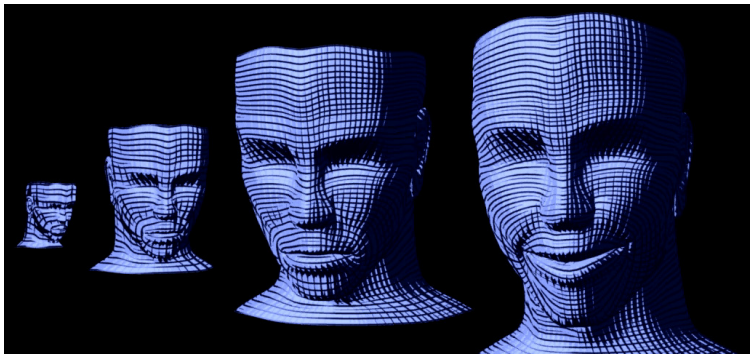
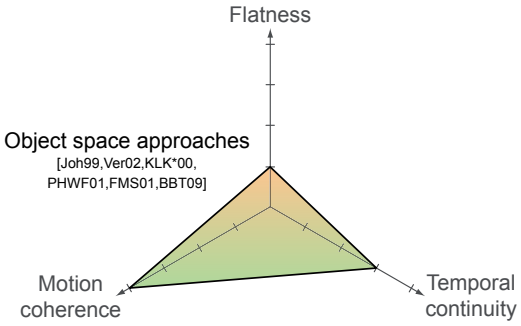


Figure 2.12: Veryovka "threshold textures". The spacing of hatching lines adapts to orientation, scale and deformation of the face model. Adapted from [Ver02].

2.2.2. Texture-based Methods

Texture-based approaches are mostly used for continuous textures (canvas, watercolor) or highly structured patterns (hatching). By embedding multiple marks, textures facilitate and accelerate rendering compared to mark-based methods. Textures can be applied over the entire image or over the 3D objects in the scene.



Object space approaches compromise flatness in favor of coherent motion by relying on texture mapping. Using object space textures, the pattern is perfectly attached to the object but is dependent of the zoom level and is severely distorted by the perspective projection. To reduce perspective distortions and scale variations, these methods try to adapt the texture to maintain a near-constant size of the pattern in screen space.

Johnston [Joh99] generates procedurally hatching patterns as a function of texture coordinates. In order to guarantee an approximately constant size of hatches in screen space, he re-maps these coordinates according to the scale of the projected texture. Veryovka [Ver02] extends this approach to obtain a more uniform distribution of hatches at all intermediate scales and better filtering (Fig. 2.12). These approaches are restricted to procedural patterns and are highly dependent on the input texture parameterization.

The *artmaps* solution [KLK*00] relies on *mipmapping* [Wil83] to adapt the scale of the texture according to the distance to the camera. This approach corrects the texture compression induced by depth and can be extended to the correction of perspective deformation using the more complex *ripmaps* mechanism. As noted by Praun et al. [PHWF01], higher temporal coherence can be obtained for hatching styles by including the binary hatches of one "tonal artmap" level into the next level (Fig. 2.13).

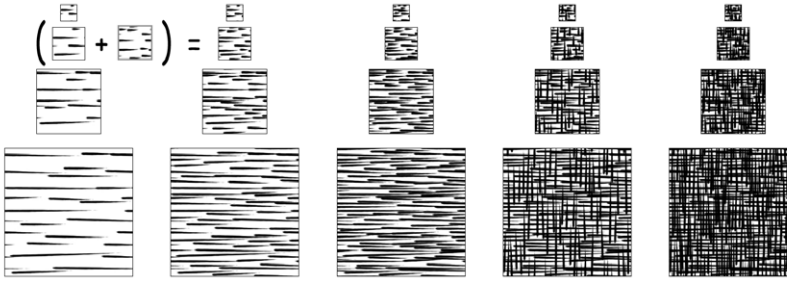
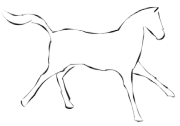


Figure 2.13: Praun et al. "Tonal Art Map". The hatching pattern of one texture level appears in all the subsequent – above and to the right – levels of the pyramids, ensuring continuity when zooming. From [PHWF01].

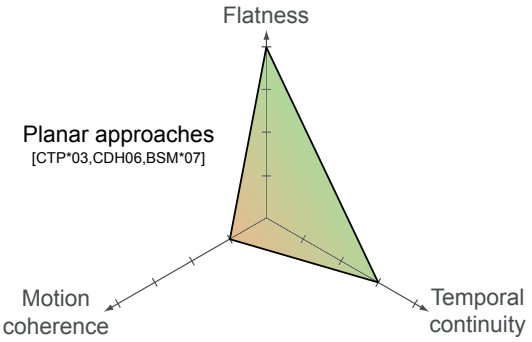


Fung and Veryovka [FVA03] propose an automatic algorithm to generate pen-and-ink tonal artmaps from grayscale textures. Freudenberg et al. [FMS01] prove the performance of these methods by integrating an artmap-based rendering in the Fly3D game engine.

In summary, object space methods offer a perfect motion coherence and compensate for most perspective distortions. They are however less effective for structured patterns for which popping or blending artifacts are more visible. Thanks to the hardware acceleration of texture mapping these approaches work in real-time and fit seamlessly into most rendering pipelines.

Image space approaches. Applying textures over the image favors flatness over motion coherence. The challenge is to deform the texture so that it follows the scene motion while preserving the appearance of the original pattern. We can distinguished between two sub-families of approaches: planar methods that deform the texture with global as-rigid-as-possible transformations, and texture advection and filtering approaches that work at the pixel level.

Planar approaches. Cunzi et al. [CTP*03] apply a 2D paper texture over the image in order to stylize a 3D environment during a real-time walkthrough. They approximate the 3D motion of the camera with 2D transformations of the texture. This approach provides a convincing trade-off between motion coherence and flatness but is limited to navigation in static scenes with a restricted set of camera motions. Cunzi et al. also introduce an infinite zoom mechanism called *fractalization* to produce a convincing feeling of zoom during camera motion. Taking inspiration from sound illusions (see insert p.23), the infinite zoom generates a dynamic texture by alpha-blending multiple scales of the original texture (Fig. 2.14). The scaling factors and blending weights are derived from the camera motion to cycle over the multiple scales and produce the illusion of a globally constant scale.



Texture fractalization achieves a very good impression of flatness, but alters the original pattern by replicating and blending multiple scales. While replicates are almost unnoticeable for self-similar stochastic textures (paper fibers, watercolor pigments), artifacts are more visible for structured patterns (brush strokes, stipples). The checkerboard in Fig. 2.14 is an extreme example of such blending artifacts. To better preserve the original pattern, Han et al.[HRRG08] propose an example-based texture synthesis algorithm that generates a texture at multiple scales during a zoom. This method has yet to be applied to non photorealistic walkthrough scenarios.

The image space mechanism proposed by Cunzi et al. models the scene as a single plane which leads to sliding artifacts for strong parallax. A better approximation of the scene motion can be obtained by modeling the scene as multiple local planes [CDH06, BSM*07]. In such local approaches, the projected 3D transformation of each part of the scene is approximated by the closest 2D rigid transformation in the least-

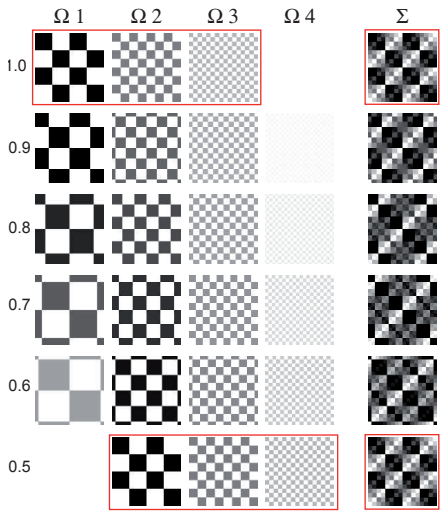


Figure 2.14: "Dynamic Canvas" infinite zoom mechanism applied on a chequerboard texture. For each zoom factor (left) the four octaves as well as their weighted sum is shown. From [CTP*03].

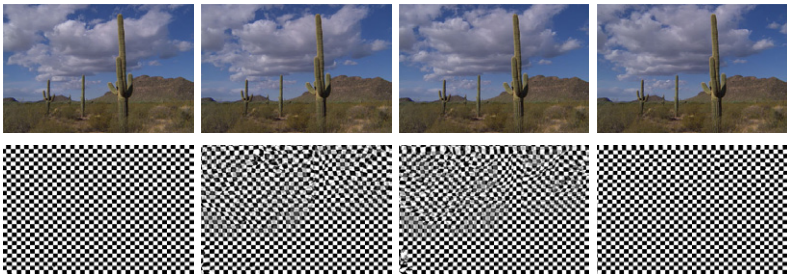
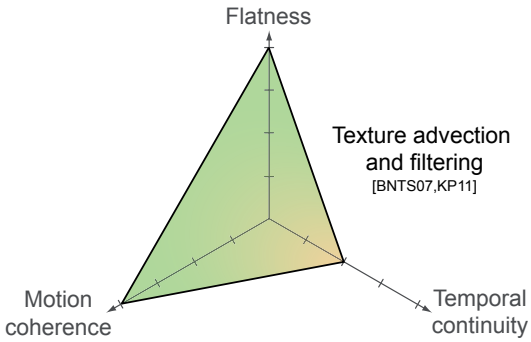


Figure 2.15: Bousseau et al. "Video Watercolorization". The checkerboard texture is advected forward and backward in time according to the optical flow of the video, and alpha-blended to minimize the distortion. Adapted from [BNTS07].

square sense. The 2D rigid transformations preserve the flatness of the texture but sliding artifacts still occur for extreme 3D motions.

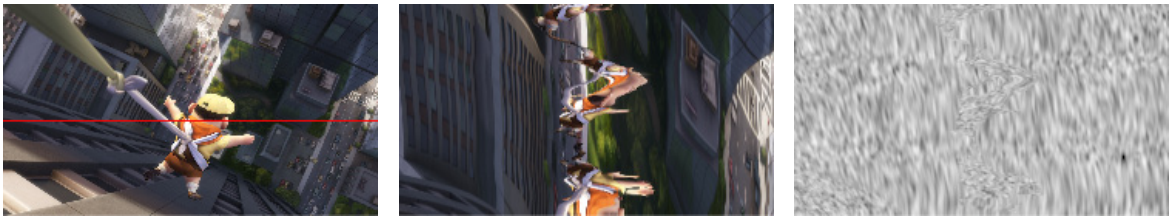


Texture advection and filtering. Bousseau et al. [BNTS07] apply non-rigid deformations to animate a texture according to the optical flow of a video [BB95]. This approach extends texture advection methods used in vector field visualization [Ney03] by advecting the texture forward and backward in time to follow the motion field. This *bidirectional* advection allows the method to deal with occlusions where the optical flow is ill-defined in the forward direction but well defined in the backward direction.

The non-rigid deformations can however distort the texture and alter the original pattern. Bousseau et al. propose an advanced blending scheme that periodically regenerates the texture to cancel distortions and favor at each pixel the advected texture with the least distortion. Similarly to texture fractalization, the method is very effective for stochastic patterns, but blending artifacts and small distortions are visible for structured patterns. In addition the bidirectional advection scheme requires the entire animation to be known in advance, which prevents the use of this method for real-time applications.

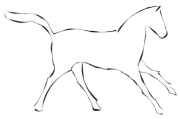
To overcome this limitation, Kass and Pesare [KP11] propose to filter a white or band-pass noise along the motion trajectory of each pixel (Fig. 2.16). Their recursive filter produces a coherent noise with stationary statistics within a frame (high flatness) and high correlations between frames (high motion coherence). This approach is fast enough for real-time application, but is restricted to isotropic procedural noise.

In summary, image-space methods ensure very good flatness but introduce either sliding artifacts, blending artifacts or local distortions. These methods are es-



(a) Selected line of original (b) Selected line through time (c) Same line of noise through time

Figure 2.16: Kass and Pesare "coherent noise". A scan line of the original image (a) is represented over time (from bottom to top) in image (b). Figure (c) displays the evolution of the noise for the same line. The smooth variations of the noise along time and the high correlation with the scene motion illustrate the temporal coherence of this approach. From [KP11].



pecially well suited to stochastic patterns such as watercolor pigments and canvas where blending and distortions are less noticeable.

2.2.3. How to choose a region extraction and rendering method?

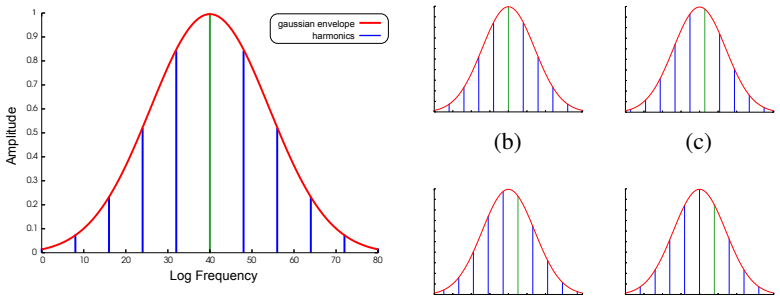
Tab. 2.3 compares extraction methods according to their ability to compute a coherent parameterization and different levels of detail. Clustering and segmentation-based methods can greatly simplify color regions but as a side effect do not preserve well fine details. Although most motion estimation methods were not designed for the particular goal of simplification, Bousseau et al. [BNTS07] demonstrate how they can be used to reduce flickering and popping when removing small details in videos.

Self-Similarity Illusion

Figure 2.17: *Ascending and Descending* by M.C. Escher (1960).

For a 1d signal, i.e. a sound, the self-similarity illusion corresponds to an endlessly increasing pitch, as demonstrated by Shepard [She29]. A Shepard tone is a complex tone with all tones in octave distance to their adjacent tone (Fig. 2.18). A gaussian envelope ensures that the tones uniformly disappear towards low and high frequencies. Playing these tones in order, looping around forever, produces the pitch "paradox" similar to the endlessly ascending staircase of Eschler (Fig. 2.17). Continuously moving the tones lead to Risset *Glissando* [Ris86].

Glassner [Gla99] extended this approach to procedural 2D signals, i.e. synthesized images. He generates visual harmonics from periodic functions such as $A(x,y,f) = \sin(fax) * \cos(fby)$. For each frame i of an animation cycle of F frames, he uses a center frequency $g = f + (i/F)f$ and evaluates at each pixel (x,y) the sum $\sum_j w_j A(x,y,2^j g)$ scaled by a bell curve (weight w_j). This approach produces endless animation, but they do not really seem to grow forever, the way the audio signal seems to rise forever (Fig. 2.19).



(a) Initial configuration

Figure 2.18: Shepard scale [She29]. The base pitch of the tone (in green) is moved progressively upwards (b-e) creating the auditory illusion of a tone that continually ascends in pitch.

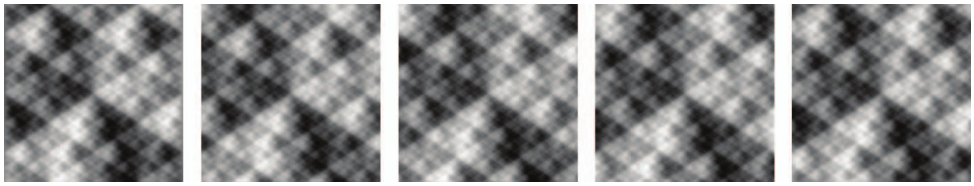


Figure 2.19: Glassner "Endless zoom" with $A(x,y,f) = \sin(2f\pi x) * (\sin(\pi f y) + \cos(3\pi f y))$ where f is evaluated at a series of upper and lower harmonics. From [Gla99].

		Coherent Parameterization	Level of Details
3D scenes	ID buffer	++	−−
	Image-space clustering [KWH06]	+/−	+
	Object-space clustering [BEDT08]	++	+
Videos	Spatio-temporal segmentation [WXSC04, CRH05, SBCv*11]	+	+
	Image filters [WOG06, KK11]	−−	++
	Optical flow [Lit97, HP00, HE04, SZK*06, BNTS07, KBD*10]	+	+
	Rotoscoping [AHSS04, LZL*10]	++	+

Table 2.3: Summary of the trade-offs made by the different families of methods for extracting color regions.

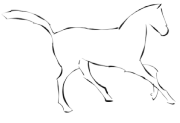
		Flatness	Coherent motion	Temporal continuity
Naïve	Static marks	++	−−	++
	Texture mapping	−−	++	++
	Random marks	++	++	−−
Few-marks	Fixed distribution [Mei96]	−/+	+	++
	Dynamic distribution [Lit97, USSK11]	++	+	−
Many-marks	Discrete LOD [KC05, BKTS06]	++	+	−
Image space texture	Planar [CTP*03, CDH06, BSM*07]	++	−	+
	Advection and filtering [BNTS07, KP11]	++	++	−/+
Object space texture	Artmaps [KLK*00, PHWF01, FMS01]	−	++	−/+

Table 2.4: Summary of the trade-offs made by the different families of methods for rendering color regions.

Tab. 2.4 summaries the trade-offs made by the different regions rendering techniques with respect to the three goals of temporal coherence. However, additional constraints can be taken into account when choosing between methods. Besides performance and ease of implementation, the range of styles that can be produced by each approach needs to be considered carefully. Tab. 2.5 compares the aptitude of each class of approach to handle stochastic (pigments, canvas), irregular (hatching, halftoning) and near regular (paint strokes, stipples) patterns.

		Stochastic (pigments, canvas)	Irregular (hatching, halftoning)	Near regular (paint strokes, stipples)
Marks	Few	−−	++	++
	Many	++	+	−
Textures	Procedural	−−	++	−−
	Artmap	+/−	++	−
	Fractalization	++	+/−	−−
	Advection	++	−	−−

Table 2.5: Summary of the media and patterns best-supported by the different families of methods for rendering regions.



3. Conclusions

Even if temporal coherence is an important problem in non-photorealistic rendering which has received a growing attention during the last fifteen years, existing solutions leave room for improvement and major challenges remain unsolved.

Progress can be made with regards to the three goals involved in the temporal coherence problem by looking for better balanced trade-offs. But beyond these goals, these solutions have to take into account additional constraints: performance, robustness, scalability, artistic control, visual complexity and diversity.

Concurrently the need for a formal evaluation of these techniques is striking. Despite the absence of absolute ground truth, perceptual measurement of the artifacts produced by each trade-off should allow to deduce objective trends from individual subjective judgments.

Part I

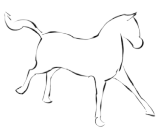
Line Stylization

Summary

Stylization is an integral part of effective line rendering. Artists vary effects such as width, texture, and precision to convey weight, shape, motion, or abstraction. A dotted line might suggest an invisible object, while an over-sketched contour might indicate emphasis or movement.

A common approach for simulating these effects in computer graphics is to define a parameterized space curve (known here as a *brush path*), and apply a stroke texture along its length. Applied textures may include dots, dashes, or captured images of marks made with pen, pencil, or other natural media. Stroke textures create attractive imagery that can appear as drawn by hand.

We investigate in this part methods to give artists such tools and control while rendering animated 2D and 3D models. The main challenge when using stylized lines in animation is to parameterize them with temporal coherence. We first present a new texture data-structure and an automatic by-example synthesis algorithm to solve the special case of the zoom on view-independent lines. We then propose two mechanisms to propagate coherent parameterizations for view-dependent lines during arbitrary 3D motion.



3

SELF-SIMILAR LINE ARTMAPS

STYLIZED line rendering for animation has traditionally traded-off between two undesirable artifacts: stroke texture sliding and stroke texture stretching. In this chapter, we propose a new stroke texture representation, the self-similar line artmap (SLAM), which avoids these two artifacts. SLAM textures provide continuous infinite zoom while maintaining approximately constant appearance in screen-space, and can be produced automatically from a single exemplar.

In the conventional artmap approach [KLK*00] (demonstarted on 2D textures), the number of textures required is $O(\log_\sigma L_{max})$, where L_{max} is the length of the longest path and σ is the maximum stretch factor allowed. Unfortunately, L_{max} is effectively unbounded for several interesting cases, including continuous zoom. Our new artmap construction (Sec. 1) is self-similar, allowing a small number of textures to smoothly cover paths of arbitrary length.

When σ is large, visible pops may appear at transitions. Blending adjacent artmap levels reduces popping, but tends to produce a blurry result. The solution is to create a dense artmap, with many textures and a small σ. We propose an example-based artmap synthesis approach (Sec. 2) that generates an arbitrarily dense artmap based on a single exemplar. Our synthesis approach also guarantees that each artmap level blends seamlessly into the next.

1. SLAM Definition

A line artmap can be visualized as a pyramid, with base width L_{max} and each level shrinking in size by a factor of σ (Fig. 3.1). For convenience, we define the artmap as a function $A(u_1, u_2, l)$, where u_1 and u_2 are the left and right texture coordinates and l is the level in the artmap, normalized such that $l = 1.0$ corresponds to texture length L_{max}. For example, $A(0, 1, 0.5)$ corresponds to the entire texture with target length $0.5L_{max}$, and $A(0.5, 1.0, 0.2)$ corresponds to the right half of the texture with target length $0.2L_{max}$. For concise notation, let $A_l = A(0, 1, l)$.

In order to support paths of arbitrary length with infinite zoom, the full artmap domain $u_1 \in \mathbb{R}, u_2 \in \mathbb{R}, l \in \mathbb{R}^+$ must be mapped somehow onto a finite set of textures

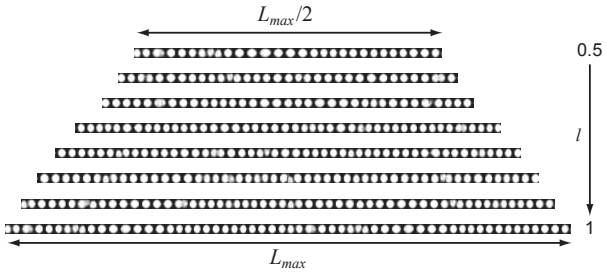


Figure 3.1: Example SLAM for a dotted line texture. This set includes eight levels, but an arbitrary number may be generated. As the scale grows, the dots tend to stretch out. When they grow too large, they split into two smaller dots. Irregularly sized dots are necessary for temporal coherence; other small imperfections are the consequence of texture synthesis.

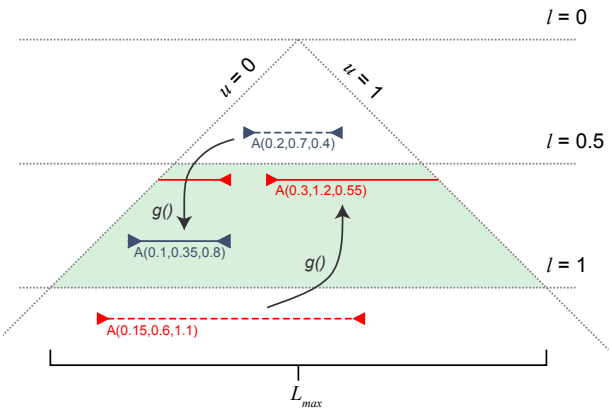


Figure 3.2: Parameterization space of a self-similar line artmap. Horizontal distance corresponds to length in screen-space, vertical distance to scale. The shaded region is where $A(u_1, u_2, l)$ is defined. Paths with scale $l < 0.5$ or $l > 1.0$ (dotted lines) are mapped to coordinates with $0.5 < l < 1.0$ under the mapping function g (solid lines). Each level of the pyramid tiles seamlessly with itself, so long lines may safely be wrapped (e.g., solid red line).

(Fig. 3.2). To accommodate arbitrary u_1 and u_2, we assume that every texture in the artmap tiles seamlessly with itself. This leaves the problem of $l > 1$. The naïve solution is to simply tile the path with the longest texture available ($A_1 = A(0, 1, 1)$), but this policy has all the negative aspects of the tiling policy.

■ *Self-similarity implies that the base of the pyramid is equivalent to its top level concatenated twice.*

Our approach is to change the problem by placing an additional constraint on the artmap: it must be *self-similar*. Precisely, there must exist at least one pair of textures A_x and A_y such that $A_x = [A_y A_y]$ (i. e., the concatenation of A_y with itself). If such a pair exists, then arbitrary stretching or shrinking can be handled by looping between the two similar textures. In practice, A_y is the top level of the pyramid while A_x is the bottom one. Fig. 3.1 shows an example of such artmap for a dotted line. Note that the bottom texture is visually equivalent to a tiled version of the top texture. For simplicity, we will use $x = 1.0$ and $y = 0.5$. The mapping from the full domain to the self-similar domain is then defined by a mapping function $g(u_1, u_2, l) \in \mathbb{R}^3$:

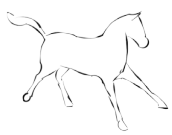
$$\begin{aligned} g(u_1, u_2, l) &= (2^t u_1, 2^t u_2, 2^{-t} l) \\ t &= \lceil \log_2 l \rceil \end{aligned} \tag{3.1}$$

If the path to render has length s in pixels, the texture used is $A(g(0, 1, \kappa s / L_{max}))$, where κ is a user-defined scaling parameter. This parameterization may tile the texture, but will not cause the texture to slide on and off the path as it grows or shrinks.

2. Artmap Synthesis by Example

Given the definition of a SLAM, the challenge is to construct one without a huge amount of manual effort. The two necessary conditions for a well-behaved SLAM are that $A_1 = [A_{0.5} A_{0.5}]$ (self-similarity), and that $A_l \approx A_{l+\varepsilon}$ (smooth variation). A desirable third condition is that any texture extracted from the SLAM closely resembles the exemplar. Our synthesis approach satisfies the first two conditions and, for many types of textures, the third condition as well.

The basis of our approach is the parametric texture synthesis (PTS) method of Portilla and Simoncelli [PS00]. Each artmap texture A_l is the result of PTS applied to a filtered white noise seed W_l. We notice that the PTS method seems to have the vital property that small changes in the seed result in small changes in the output. We exploit this property by constructing a set of seeds W_l that satisfy the SLAM



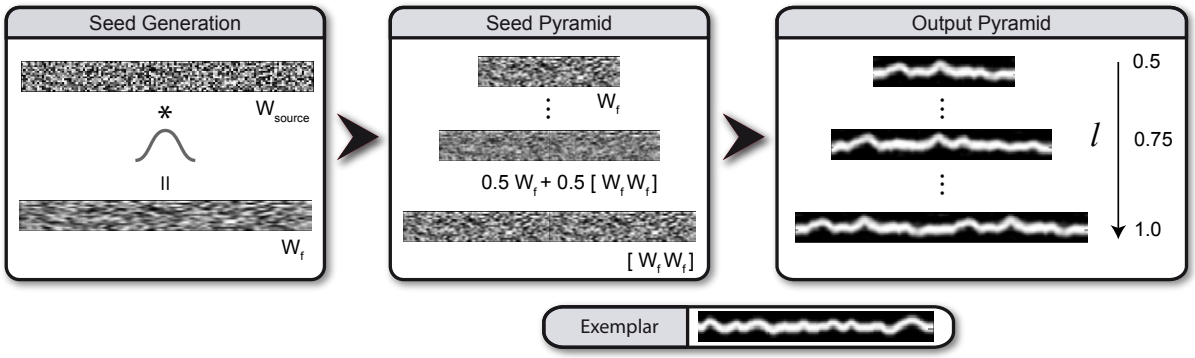


Figure 3.3: Synthesis of a self-similar line artmap. An initial white noise seed W_{source} is slightly filtered to produce a filtered seed W_f. The filtered seed W_f is scaled to half its original length and placed at the top of the seed pyramid. The scaled W_f is concatenated with itself and placed at the bottom of the pyramid. Interior levels of the pyramid are created by linearly interpolating the top and bottom. Finally, each seed is used to synthesize the corresponding level of the artmap pyramid.

conditions of self-similarity and smooth variation. Because of the continuous behavior of PTS, if the W_l satisfy the SLAM conditions, then A_l should also satisfy the conditions.

■ *Parametric texture synthesis preserves the self-similarity and smooth variation properties of the seed pyramid.*

Using PTS, the quality of each individual A_l is less than what might be achieved using a modern non-parametric synthesis method (e. g.[LH06]). However, non-parametric methods tend to lack the property that small changes in the synthesis seed produce small changes in the output texture. This property arises because parametric methods are based on iteratively adjusting statistics of a random noise seed, and each of these adjustments (e. g., histogram matching) is a continuous operation. By contrast, non-parametric methods are based on iterative neighborhood matching, which produces a chain of discrete decisions that can change drastically with small perturbations to the input.

Note, however, that we do not have a formal proof of this desirable behavior of PTS. In our experiments, PTS provides continuous behavior for a variety of textures, but there may still be cases for which PTS fails to create smooth variation.

2.1. Construction of Synthesis Seeds

The seeds W_l are scaled and filtered versions of a fixed white noise texture W_{source}, which has length L_{max}. Ideally, all W_l have perfect white noise statistics, but perfect white noise is not necessary as PTS will coerce the statistics to match the exemplar. We therefore only need to ensure that the W_l satisfy the self-similarity and smooth variation conditions, and have roughly white noise statistics. In practice, we start from a single filtered version of W_{source}, and scale and linearly interpolate it to produce each W_l (Fig. 3.3).

The original white noise texture W_{source} is filtered with a small Gaussian filter (we use a width of 5 pixels) to produce a smoothed version W_f. This smoothing is necessary to avoid aliasing when the seed is scaled. The smoothed seed W_f is then adjusted using histogram matching to restore a uniform distribution across the range $(0,1)$. The top level of the seed pyramid $W_{0.5}$ is simply W_f scaled to half its original length. The bottom level of the pyramid is $W_{1.0} = [W_{0.5} W_{0.5}]$, that is the

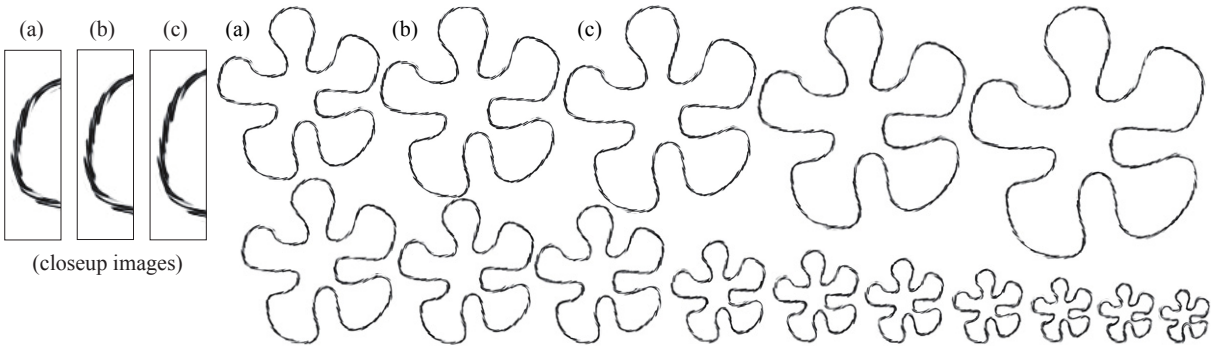


Figure 3.4: Zoom sequence on a 2D vector-art flower. Note that at every frame the screen-space size of the scribbles is well preserved and that no alpha-blending artifacts are noticeable. See especially the inset closeups of the upper left three flowers for continuity.

concatenation of $W_{0.5}$ with itself. The interior levels of the pyramid are generated by scaling $W_{0.5}$ and $W_{1.0}$ to the proper length, then linearly interpolating:

$$\begin{aligned} W_l &= \alpha W_{1.0} + (1-\alpha) W_{0.5} \\ \alpha &= 2l - 1 \end{aligned} \tag{3.2}$$

The seed pyramid satisfies the self-similarity condition by construction, since $W_{1.0}$ is a tiled version of $W_{0.5}$. The smooth variation condition is satisfied by the linear interpolation of the top and bottom of the pyramid.

2.2. Extension to 1.5D Textures

So far, we have considered only 1D textures in our description. However, all the results in this paper were generated with stroke textures with width between 8 and 32 pixels. We call these brush textures 1.5D, since they are essentially 1D textures with a finite width. A few modifications to the synthesis algorithm are necessary to accommodate 1.5D textures.

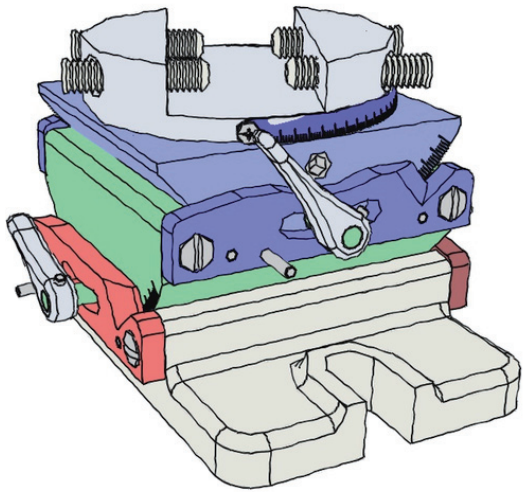
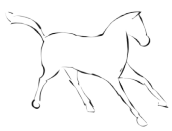


Figure 3.5: 3D creases of a vise model stylized using SLAM.

The construction of the seed pyramid is similar to the 1D case. W_{source} becomes a $d \times L_{max}$ white noise texture, where d is the number of rows. The initial filtering operation from W_{source} to W_f is only performed in 1D (the horizontal direction in Fig. 3.3), because W_f is only scaled in one dimension. The remaining operations to produce the seed pyramid are identical to the 1D case.

The largest changes are required in the texture synthesis, because our 1.5D stroke textures violate the stationarity assumption of the PTS method. More specifically, the statistics of each row of a 1.5D texture are not identical: the top and bottom rows are usually dark (transparent), while the middle rows are usually bright (solid).

Our approach is to apply different statistics to each row of the texture. While the PTS method stores and applies many different statistics of the source texture,



through experimentation we found that varying only the *mean* and *variance* by row sufficed to produce good results. We therefore modify the PTS method to gather and apply the mean and variance per-row, and all other statistics per-texture. This modification is only a few lines of MATLAB©code.

As illustrated by Fig. 3.4 and Fig. 3.7, applying 1.5D SLAMs to 2D vector-art animations ensures a strong temporal coherence of the stylization, while preserving the 2D characteristics of the stroke textures. Note in particular the quasi-constant size of the scribbles at each zoom level, and the absence of blending artifacts. This approach can also be used for fixed 3D segments, such as the edges of a cube or sharp creases (Fig. 3.5).

3. Results

Our SLAM construction process leverages the texture synthesis code of Portilla and Simoncelli [PS00], and our implementation is available for download.[1] Results for several different brush textures are shown in Fig. 3.6.

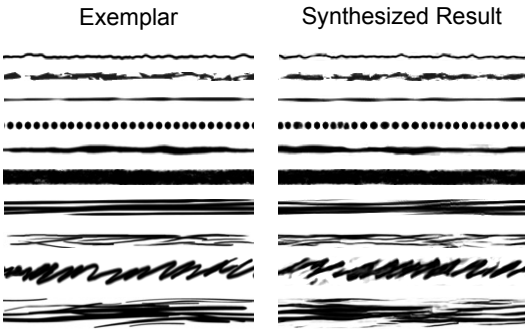


Figure 3.6: Texture synthesis results. The method of Portilla and Simoncelli [PS00] produces good results for many types of brush textures, especially small or simple textures (top half). More complex textures (bottom half) are reproduced credibly, but have imperfections in detail.

For simple textures, the PST algorithm produces results almost indistinguishable from the exemplar. For more complex textures such as the scribble and the jumbled lines (bottom of Fig. 3.6) the overall impression of the synthesized result is similar, but some details are lost. High quality SLAMs are composed of 17 grayscale images of 1024×32 pixels (100 to 300 KB in PNG). Synthesis typically takes a few minutes.

We believe the quality of our synthesized textures is more than sufficient for many styles of rendering, but the quality is not perfect. As shown in Fig. 3.6, complex textures may not be reproduced faithfully. Conventional non-parametric synthesis does not satisfy the condition of smooth variation (Sec. 1), but texture advection techniques

Figure 3.7: Zoom on a purely 2D figure drawn with stylized lines using self-similar textures. At different scales the lines appear qualitatively similar, and transitions exhibit temporal coherence. See the accompanying video of [BCGF10] to better appreciate the continuity.

[1] http://www.cs.princeton.edu/gfx/proj/dpix/

[KEBK05, BNTS07] may provide an alternative for synthesis. In our experiments, however, the animation of the PTS textures was more visually appealing than basic advection.

We believe a more promising route to higher-quality is to use the synthesized textures *themselves* as seeds for a final rendering pass using procedural strokes. For example, our synthesized dotted line texture could be used as a guide for the size and placement of particle sprites.

4

LINE PARAMETERIZATION BY ROBUST VOTES PROPAGATION

APPLYING Self-Similar Lines Artmaps to complex curved paths – represented as a list of segments – requires the definition of a parameterization T. The most straightforward choice for T is the screen-space arclength of the path s. However, for any view-dependent feature lines (silhouettes, suggestive contours, apparent ridges, etc.) this approach produces "swimming" artifacts, as no temporal coherence is ensured from one frame to the next.

To ensure coherence, two questions need to be answer: (1) what are the correspondences between the two sets of paths at frame f and $f+1$; (2) how to make their parameterization evolve when their arc-length changes? If our SLAMs solve this last question, finding relevant correspondences is not trivial because view-dependent lines are sliding on the surface of the object, and because they may appear, disappear, merge or split between two frames. Our solution to this problem doesn't explicitly compute these correspondences. Instead, we rely on the fact that view-dependent lines enjoy a natural coherence in screen space to design a 2D voting scheme that propagates the parameterization of the current paths to the next frame.

At each frame, for each path, our goal is to find the parameters (ρ, ϕ) such that the path's parameterization $T(s) = \rho s + \phi$ best fits the parameterization of the previous frame. To do so, we splat the parameterization of all the current paths in a 2D buffer (Sec. 1). At the next frame, this buffer is read back, and parameterization votes are recorded for each path. The new parameters that best fit these votes are estimated using a robust data fitting algorithm (Sec. 2) and combined with SLAMs to render strokes (Sec. 3). Fig. 4.1 gives an overview of the whole process.

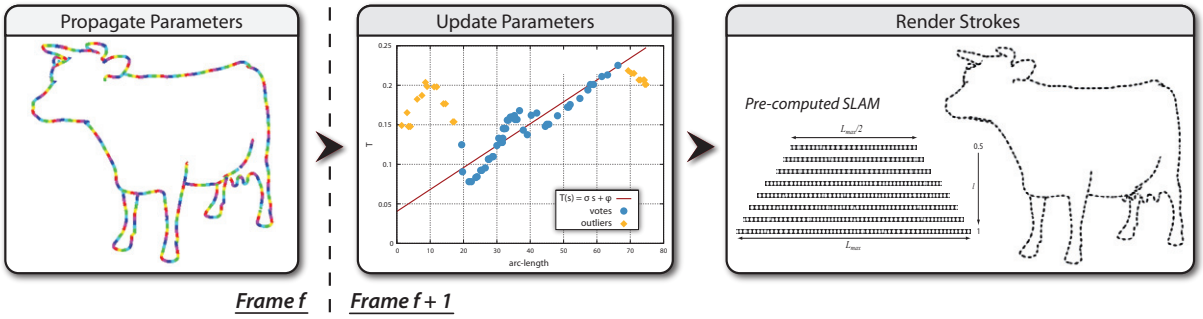


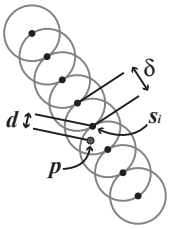
Figure 4.1: Three steps pipeline for coherent line parameterization. At frame f, the parameterization of all the visible brush paths is splatted into a screen-space buffer. At frame $f+1$, this buffer is read back, and parameterization votes are recorded for each path. The new scale factor s and phase ϕ that best fit these votes are robustly estimated using RANSAC. Finally, paths are rendered according to this parameterization, choosing the appropriate level in a pre-computed SLAM.

1. Propagating Parameters

The first step consists in propagating the parameterization of all the paths at frame f to the next frame $f+1$. At frame f we evenly sample each visible path in screen-space at a user-defined sampling rate δ. Each sample is parameterized by its 2D arclength s_i and the current parameters of the path (ρ, ϕ).

Observing that view-dependent lines are moving in a rather smooth and continuous way in screen-space, the key idea of our approach is to extend the size of each sample and splat its corresponding parameterization into a screen-space buffer (the *parameterization buffer*). View-dependent lines at frame $f+1$ can thus look up their closest correspondent at frame f. The contributions of overlapping paths are added, tending to unify their parameterization. If the paths shift too far between frame f and $f+1$ the lookup will fail, but coherence of the stylization may be unnecessary or even undesirable in that case.

The sample parameterization $t_i = \rho s_i + \phi$ is written as a textured sprite of radius δ in a floating point off-screen buffer (inset left). Each location p at a distance d from the center s_i of the sprite determines a weight for the splatted parameterization t_i by:



$$w_i = clamp(1 - \frac{d}{\delta}, 0, 1)$$

We keep track of the weight by splatting w_i in another color channel. Thus, at the end of this step, each pixel of the buffer covered by j overlapping samples contains $(\sum_j w_j t_j, \sum_j w_j)$. Note that by design this weighting scheme corresponds to the exact linear interpolation of the parameterization along the spine of the line, if no path overlaps. The weight of the splat varies with the partial visibility [CF10] of the sample, or the z-depth of the sample at that point.

Unlike that of Kalnins et al. [KDMF03], our method does not rely on an ID image (item buffer). The item buffer approach can introduce aliasing artifacts, and requires a local neighborhood search for each sample. Our splatting approach avoids this search, and also allows the additional control of varying the sprite size to trade-off coherence of parameterization against parameterization overlap. Increasing the size of the sprite makes it less likely that a path will move outside the sprite radius in a single frame, but increases the chance that separate lines will interfere with each other.

2. Updating Parameters

At frame $f+1$, we uniformly sample the new visible brush paths in screen-space and reproject each sample to the previous frame's parameterization buffer, using the camera from frame f. If $\sum_j w_j$ is defined at this pixel location, we record a vote:

$$v_k = \left(\frac{\sum_j w_j t_j}{\sum_j w_j}, s_k \right) \tag{4.1}$$

which is a pair of values that associates the averaged parameterization at screen-space location j in frame f with the current arclength parameter s_k of this sample.

For each path, we compute the scale factor ρ and the phase ϕ that best fit the votes in the least-square sense. As the input votes can be very noisy when paths overlap, we use the RANSAC algorithm [FB81] to robustly estimate these two parameters. This iterative method allows a trade-off between speed and accuracy: the user estimates the proportion of outlying votes (refined according to data during the computation) and a target probability for inlying votes, which control the number of required iterations.

If fewer than two votes have been recorded for a path, we parameterize the path by $T(s) = s/L$ where L is the path length, such that T ranges in $[0,1]$. The same default parameterization is used to initialize the first frame rendering.

Note that this approach is similar in spirit to the mixed policy described by Bourdev [Bou98] and used by Kalnins et al. [KMM*02]. As this policy mixes information from multiple paths, popping can occur when two paths merge. Our method shares this limitation, however the robust fitting operation reduces sensitivity to outlying votes, allowing paths to quickly find a common parameterization. We thus avoid the fragmentation problem encountered by Kalnins et al. [KDMF03].

3. Stroke Rendering

In the last stage, we use the path parameters to determine the appropriate SLAM level l and render the final textured stroke. To ensure that paths moving off-screen remain at the same level in the SLAM, we define l as a function of the scaling factor ρ. Unlike the length of the path, ρ does not change as the path is clipped. Taking into account the self-similarity of the SLAM, the formula is:

$$
\begin{aligned}
d &= \rho k L_{\max} \\
l &= \left\{ \begin{array}{ll} \omega/d & \text{if } d > 2 \\ 1/d & \text{if } d \in [1,2] \\ 1/(d\Omega) & \text{if } d < 1 \end{array} \right| \quad \in [0.5,1]
\end{aligned}
$$

with:

$$\omega = 2^{\lfloor log_2(1/d) \rfloor} \quad \text{and} \quad \Omega = \lfloor 2/d \rfloor$$

where $L_{\max}$ is the length of the longest texture in the SLAM and k is a user-defined scaling factor. When looping from one end of the truncated pyramid to the other, the path parameterization T has to be scaled by $1/\omega$ and Ω respectively.

A SLAM can be simply represented as a 3D texture, allowing us to make use of trilinear filtering on the GPU. We assign 3D texture coordinates to each segment according to the path parameterization and SLAM level ($w = 2l - 1$).

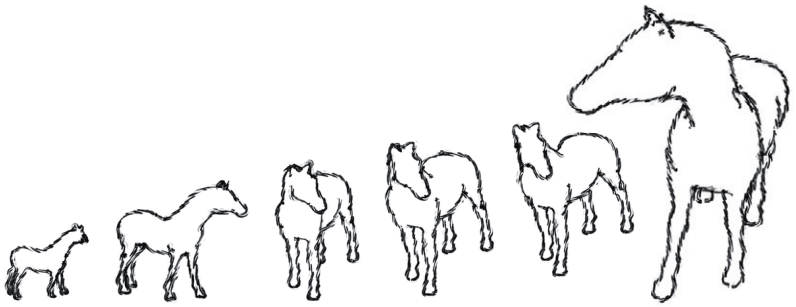


Figure 4.2: Zoom and rotation around a 3D model with stylized silhouettes. See the accompanying video of [BCGF10] to appreciate the temporal cohrence of the stylization.

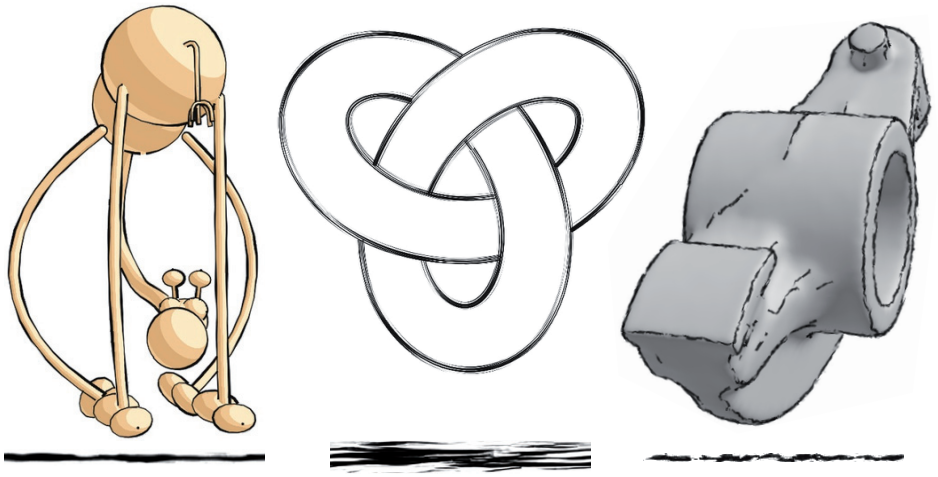


Figure 4.3: Various styles obtained by applying SLAMs on view-dependant lines extracted from 3D models. The bottom row shows the line texture of one level of the associated SLAM.

4. Results and Discussion

We implemented the entire coherent line parameterization method on the CPU without fine optimizations. We use the segment atlas data-structure describe by Cole and Finkelstein [CF10] to compute the visibility of the paths and perform the final rendering of the strokes on the GPU. Nevertheless, our approach could be used with any system that renders lines as textured quads, such as [MH04, KMM*02, IHS02, MKG*97].

Our system runs at interactive framerates for scenes of moderate complexity on a commodity PC with an Intel Core 2 Duo 2.4 GHz CPU and 4 GB RAM. The bottleneck of our approach is the RANSAC algorithm, which varies in performance depending on the number of visible segments and the number of outliers. However, this algorithm is intrinsically parallel since the parameterization of each path can be fitted independently. Our system uses OpenMP to accelerate this step on the CPU. Although we currently do not have a GPU implementation of this algorithm, we believe it should be practical and would significantly improve performance for complex models.

Fig. 4.3 illustrates the wide variety of styles which can be achieved applying different SLAMs on view-dependent lines extracted from 3D models. Note that creating new styles only requires to synthesis new SLAMs by example. We refer the reader to Fig. 4.2 and the accompanying video of [BCGF10] to appreciate the temporal coherence of this stylization during the animation. Even if some popping is noticeable when brush paths merge, our simple parameterization scheme in conjunction with SLAMs provides a convincing screen-space behavior for most textured lines.

While this parameterization scheme provides a good trade-off between performance and implementation complexity, the general problem of temporally coherent line parameterization remains unsolved. In particular, neither our method nor previous work (e. g. [KDMF03]) works well for complex models with many overlapping lines, because the parameterization of all the lines is mixed in a 2D buffer. Satisfactory results for complex models will likely require a new approach that does not rely solely on a screen-space buffer to propagate parameterization from frame to frame. We present such an approach in the next chapter.

5

SNAKES-BASED LINE TRACKING AND PARAMETERIZATION

THE APPROACH we describe in this chapter reconciles the simplicity and natural coherence of image space lines with the richness of stylization available to object space methods while providing robust and coherent parametrerization for complex models.

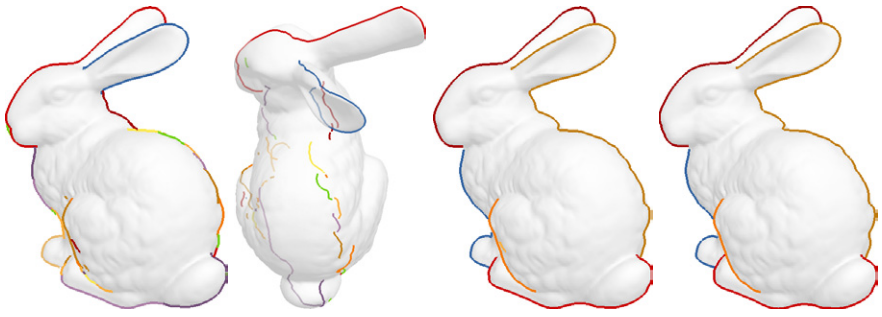
Our input is a set of unorganized 3D line samples (with tangent and velocity), which may be produced by any line extraction method in either image or object space. The key insight by which our method straddles image and object space is the use of active contours (see insert p.42) to recover connectivity for the independent feature samples; and we do so in a way that supports temporal coherence by propagating the snakes from frame to frame following the motion field in the scene (Fig. 5.1).

While the traditional snakes formulation allows them to conform to the changing shape of the features in the scene, an added challenge is that they also need to adapt their topology and overall arrangement during animation. Therefore we present new mechanisms to add, remove, trim, extend, split and merge these contours so as to match the animated feature lines.

Over these snakes, a set of brush paths carry consistent stretches of parameterization from one frame to the next (Fig. 5.3). In support of abstraction, these paths may deviate substantially from the underlying snake without harming its ability to track the motion in the scene.

1. Overview

Our method uses active contours (a.k.a. "snakes") [KWT88] to construct a smooth parameterization of the feature lines extracted from the 3D model, and to propagate this information with coherence during animation.



(a) silhouettes in 3D (b) side view of (a) (c) snakes at frame 1 (d) snakes at frame 2

Figure 5.1: (a) Object space silhouettes appear to be continuous in image space, but (b) are extracted as many disconnected parts. (c-d) Snakes recover 2D connectivity and provide coherence.

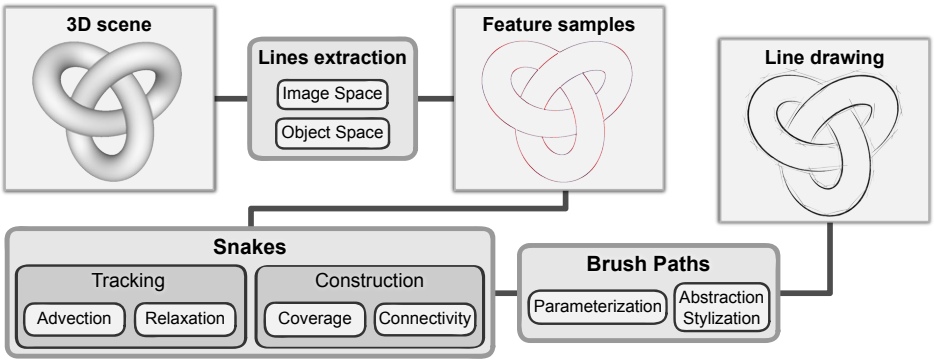


Figure 5.2: Stages of our line drawing pipeline, an overview of which is given in Sec. 1.

Goals. In order to produce clean and temporally coherent lines suitable for stylization, we set out six goals for these snakes:

- *accuracy*: Snakes should faithfully represent the shape of the underlying linear features extracted from the scene.
- *coverage*: Each feature line should be covered by one snake and each snake should cover a feature.
- *length*: Snakes should be as long as possible.
- *simplicity*: Snakes should maintain simple, clean topology.
- *smoothness*: Snakes should depict smooth paths.
- *coherence*: Snakes should evolve continuously from frame to frame as they track the features of the model.

Fig. 5.2 illustrates the pipeline we use to achieve these goals. The input is a set of feature lines samples containing 2D position, 2D velocity, and local 2D tangent. The samples may or may not be connected. For the first frame only, we construct a set of snakes conforming to these feature samples via a stochastic process described in Sec. 3.

■ *We combine vectorization, advection and relaxation to track the feature lines in 2D.*

For successive frames, snakes from the previous frame are advected to follow the motion field in the scene (Sec. 2.2), in support of *coherence*. Next they are relaxed to conform to the features in the current frame (Sec. 2.3), supporting a balance between *accuracy* and *smoothness* – a hallmark property of the snakes formulation.

Active contours

Introduced by Kass et al. [KWT88], snakes are widely used in computer vision and medical imaging applications because of their robustness to noise. They have also been applied in several NPR contexts: painterly rendering [Her01] and video stylization [Aga02, AHSS04]. The systems of Agarwala et al. segment and track color regions in videos with B-spline snakes for coherent stylization. The user manually draws the initial outlines of the contours and the outlines are propagated to track features in the video. These systems were designed for processing video offline and can therefore rely on heavy-weight optimization and manual correction.

In contrast, our approach is designed to track motion field in animated 3D scenes where the camera path is not known in advance, for example games. Therefore it must be interactive, fully automatic, and cannot rely on "future" frames. On the other hand, we benefit from the quality of our input motion – the motion field extracted from 3D is generally less noisy than optical flow from video.

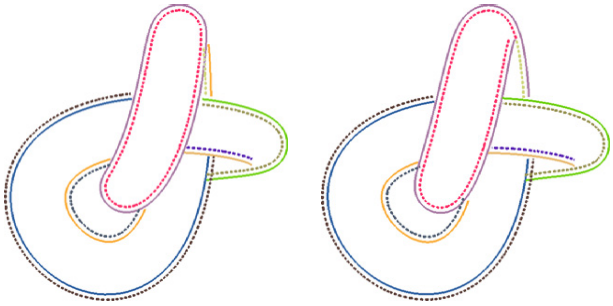


Figure 5.3: Each brush path (dotted) is defined relative to a snake (solid), but can deviate substantially to support abstraction effects. As the knot turns, the snake topology evolves; for example the orange snake merges with the purple snake (top). Coherent arc-length parameterization is preserved by maintaining the beige brush path over the purple snake.

After advection and relaxation, the goals of *length, coverage* and *simplicity* will generally have been violated. The ideal solution would involve a complex non-linear optimization over both discrete and continuous variables, unlikely in a system designed for interactive frame rates. Instead, we adopt a greedy approach and update each snake individually, using heuristics to move towards a better solution (Sec. 3). We think of this process as "vectorizing" the features in this frame; but rather than starting from scratch we begin with a good guess adapted from the previous frame. This approach effectively produces long, continuous paths from unorganized input points, and also provides a form of level of detail control.

Brush paths. We build on top of our snakes a set of brush paths (Fig. 5.3) that provide a coherent parameterization across frames (Sec. 4). Freed from the need to track, they can deviate largely from the initial feature line position to satisfy other stylization and abstraction goals.

2. Tracking Feature Lines

Unlike previous approaches such as Kalnins et al. [KDMF03], our method does not directly stylize the lines extracted from the 3D model. Instead, the lines are approximated and tracked by snakes [KWT88]. A snake is a 2D polyline that acts as an elastic strand, simultaneously attempting to satisfy external attraction forces and internal smoothing forces. In our approach the snakes are attracted to feature lines and track them from frame to frame. At each frame, we move the snakes according to the motion field in the scene (Sec. 2.2) and then relax them to conform to the feature lines in the current frame (Sec. 2.3).

2.1. Input

Our method takes as input a set of feature samples with 2D position and 2D tangent, and for animated scenes, 2D velocity. These samples can be either computed from any object space line definition (coupled with a visibility computation method such as the "segment atlas" approach of Cole and Finkelstein [CF10]), or extracted by any image space method which provides an orientation at each pixel (e. g., [LMLH07, VVC*11]).

One image filtering method that provides very smooth 2D orientations along with intuitive lines is contour extraction using steerable filters [FA91]. We apply the contour extraction to a shaded image, allowing us to capture lighting as well as shape effects (similar to [LMLH07]). The image is optionally augmented with depth information to ensure that depth discontinuities are captured by lines. The

input to this extraction method is a scalar image $I(x,y) = L(x,y) + \beta z$ where x and y are the screen space pixel coordinates, $L(x,y)$ is any shaded image, and β is a parameter that controls the influence of depth z. We apply to I the second-order quadrature filter pair contour detector of Freeman and Adelson [FA91]. This method makes a good estimate of dominant orientation in screen space, which we use for the tangent at each contour pixel. Depth is estimated by sampling three points at 1 pixel spacing along the perpendicular to the tangent and taking the closest value. The velocity is extracted at the closest sample.

2.2. Advection

At initialization, each snake vertex is associated with a 3D feature sample (Sec. 3). During advection, the snake vertex is moved according to the projected 3D motion field of the scene. Advecting in 3D ensures accurate tracking when two contours that are close in 2D are tracking feature lines that are far in 3D, as is common at T-junctions and silhouette cusps.

The 3D motion field is computed by reprojecting positions from the scene at frame $i-1$ into the camera at frame i and computing the difference, similar to the approach of Kalnins et al. [KDMF03]. For static scenes, the 3D position of the sample is the same in the two frames, but for animated scenes the position+velocity is reprojected for proper correspondence. This computation is exact for lines that remain fixed on the model (ridges, valleys and creases), but is approximate for view-dependent lines that slide on the surface of the object as the camera moves. Unlike the explicit search for nearby contours performed by Kalnins et al., we use relaxation to attach the advected snakes to features in frame i.

2.3. Relaxation

As described by Kass et al. [KWT88], snakes are parametric 2D curves $\mathbf{v}(s) = (x(s), y(s))$, $s \in [0,1]$ that minimize the energy E, defined as the sum of smoothing and data terms E_{int} and E_{ext}:

$$E = \int_0^1 E_{int}(\mathbf{v}(s)) + E_{ext}(\mathbf{v}(s))\, ds$$

■ *Snakes attempt to match areas of interest while preserving their continuity and smoothness.*

The internal energy E_{int} ensures *continuity* (first-order membrane term weighted by α) and *smoothness*, expressed as second-order thin-plate term weighted by β):

$$E_{int}(v(s)) = \frac{1}{2}(\alpha(s)|\mathbf{v}'(s)|^2 + \beta(s)|\mathbf{v}''(s)|^2)$$

Addressing the *accuracy* goal, the external energy E_{ext} attracts the snake to areas of interest – in our case, feature lines that are projected and rasterized into a binary image I. This image is filtered by a Gaussian kernel whose gradient is the attraction field:

$$E_{ext}(v(s)) = -|\nabla [G_\sigma(\mathbf{v}(s)) \star I(\mathbf{v}(s))]|^2$$

The size σ of the filter kernel controls the width of the attraction field around the contour. A larger filter makes relaxation more robust to errors in advection, at

the price of accuracy in tracking. In our experiments we use σ = 8px for screen-resolution images.

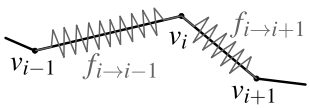
Following the approach of Kass et al. we optimize the energy E by gradient descent. The continuous snake $\mathbf{v}(s)$ is discretized as n vertices v_i and the energy is iteratively minimized by a semi-implicit Euler scheme. At each frame, a matrix of internal forces is constructed and applied iteratively, updating the contours positions and their associated external forces at each iteration. This scheme is simple and fast so long as the advection step provides a good starting condition. But when advection fails to produce a good initial guess it is likely due to extreme motion of the model. In this case, temporal coherence may not be necessary or even expected.

To maintain good sampling, especially while zooming, the snake resolution is updated after several iterations of relaxation. Via the approach of Delingette et al. [DM00] we ensure an almost uniform sampling of the curves, with a maximum distance of δ_{max} = 6px.

Snakes have a natural tendency to shrink like a rubber-band due to their internal forces. This shrinking tends to produce inaccurate results where snakes round off corners or fail to reach line endpoints. To overcome this behavior, we apply spring forces along each snake edge so that they maintain almost constant length if no tangential external force is applied. The spring forces is implemented as an extra term in the external energy, similar to the approach of McInerney et al. [MT95]:

$$f_{i\to i+1} = k\left(L_i - \|v_{i+1} - v_i\|\right)\frac{v_{i+1} - v_i}{\|v_{i+1} - v_i\|}$$

$$f_{i\to i-1} = k\left(L_{i-1} - \|v_{i-1} - v_i\|\right)\frac{v_{i-1} - v_i}{\|v_{i-1} - v_i\|}$$



where k is the stiffness parameter of the spring (k = 1 by default) and L_i is the rest-length equal to $\|v_{i+1} - v_i\|$ at the previous frame.

3. Vectorizing Feature Lines

During animation snakes must be created, deleted, or modified to preserve a good approximation of the underlying feature lines. Constructing snakes from the feature samples is equivalent to vectorizing line art. This is a difficult problem even for still images, and our scenario also requires that the vectorization be temporally coherent. Fortunately, we can exploit the smooth tracking behavior of the snakes to ease the vectorization problem. Snakes from the previous frame that have been advected and relaxed provide an excellent starting point for vectorizing the samples for the current frame, because the features they track are coherent in image space.

■ *Snakes update and construction can be seen as line art vectorization throughout time.*

However, starting with these snakes, we still need to address two challenges. The first challenge is that after advection and relaxation we generally have not met our unit coverage goal. Several snakes may now overlap in image space where a single snake would suffice (see insert p.46). Also, new lines may have appeared in regions where no snake existed before, or conversely a snake may remain where feature lines no longer exist.

Line drawing LOD

When using object space line extraction techniques, level of detail (LOD) is mandatory to create abstraction, because most feature lines are extracted independently of their final projected location on screen. LOD prevents line clutter by controlling the local coverage of the image, i. e. the number of lines that live on an given area of the image in a given direction.

Solutions to simplify line drawings have been proposed for *static* images. They can occur both during the stylization [GDS04] and as post-process [BTS05, SC08]. These three methods are based on a directional coverage measure, which is used by the first to omit some lines, and by the other two to replace a set of lines by a representative line. In contrast, Cole et al. [CDF*06] propose a *dynamic* solution based on a "priority buffer" (inspired by the "item buffer" of Kalnins et al. [KMM*02]). This method controls the local density by omitting lines of low priority in areas of high density. However, it does not support modification of their positions so as to replace two or more nearby lines by their "average."

All image-based line methods impose an intrinsic simplification behavior based on screen space proximity. Nevertheless, vectorizing this raster input necessitates a decision as to whether or not two unconnected feature samples belong to the same line. Taking inspiration from the work of Grabli et al. [GDS04] and Barla et al. [BTS05], we propose a simple definition of the coverage that allows us to solve the problem efficiently with explicit controls over spatial and directional proximity. Finally, by generating snakes and therefore brush paths that are independent of the feature lines, we can circumvent the limitation of Cole et al. [CDF*06] by generating a stroke that is the "average" of two or more input features.

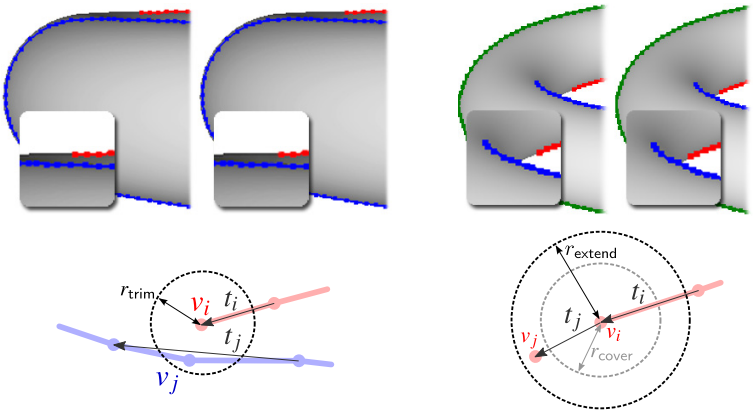
The second challenge is snake topology. Unlike our scenario, typical formulations for parametric snakes need not address the issue of automatically inferring their connectivity, usually by relying on user input. Starting from this initial connectivity, some approaches have been proposed to modify topology of the contours during segmentation tasks [McI00, DM00, Ji02]. We take inspiration from these methods to define a set of heuristic rules that allow the snakes to change their topology and overall arrangement during animation so as to match the underlying feature lines.

■ *Six operators modify the topology and coverage of the snakes to match the feature lines.*

We organize these heuristics as six operators – delete, trim, extend, split, merge, insert – which are applied sequentially in a greedy fashion on the vertices of the snakes, until they can no longer modify the snakes.

The order in which the operators are applied is important. Deletion and splitting occur first, so as to allow subsequent trimming to produce clean junctions. Extension also benefits from following splitting, and must precede merging to allow snakes to join across a gap. Insertion of new snakes occurs to address features without coverage after all the other operations have had the opportunity to adjust the snakes (especially extension). So in summary, we delete as much as possible, trim as much as possible, and so on; and finally insert new snakes where extra coverage is needed.

On the other hand, the order in which the snakes are processed for each operation has relatively less influence on the quality of the final result. Nevertheless, we process snakes in order from longest to shortest, except for the trim operator for which we use the opposite order, with the goal of maximizing length of long snakes.



(a) Trim (b) Extend

Figure 5.4: Modifying snakes to improve coverage. (a) The endpoint of a snake is trimmed if it is too close and parallel to another snake. When multiple snakes overlap in screen space this operation moves towards unit coverage. (b) The end of a snake is extended when doing so adds coverage to otherwise uncovered features. In concert, the trim and extend operations tend to clean up T-junctions.

3.1. Operations

Coverage. Feature samples extracted from 3D models generally do not provide uniform density when projected into the image plane; they are typically sparse, but also often overlap in image space. Our goal is to cover each feature line with exactly one snake. Here "coverage" means that a feature sample f_i with tangent $\vec{T_i}$ has a snake vertex v_j nearby whose local orientation $\vec{t_j}$ is similar:

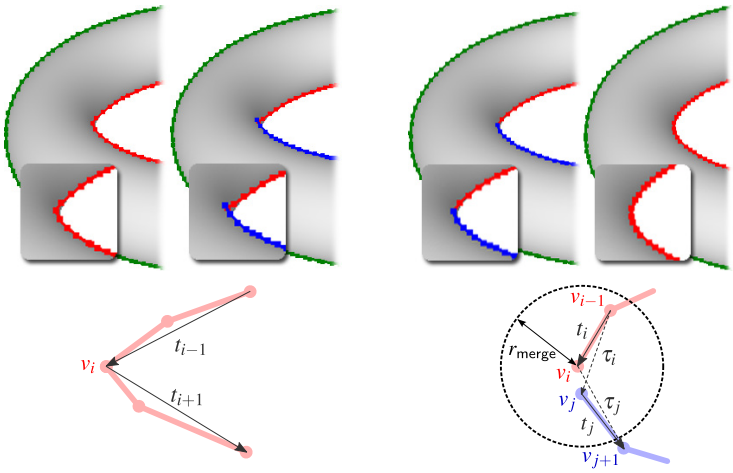
$$\text{covered}(f_i) \equiv \exists v_j : \begin{cases} \|f_i - v_j\| & < & r_{\text{cover}} \\ |\vec{T_i} \cdot \vec{t_j}| & > & \theta_c \end{cases}$$

The constants r_{cover} and θ_c control, respectively, the apparent line density and the tendency for snakes to match the orientation of the underlying features. A related notion is $\text{footprint}(v_j)$ which describes the set of feature points covered by v_j.

Initialization. For the first frame of an animation we start by creating new snakes as follows. We randomly pick an uncovered feature sample from which we grow a snake as far as possible using the extension operator described below. This process is repeated until it can no longer improve coverage. Though not providing optimality guarantees, this approach is simple and fast, and it offers a reasonable first guess for subsequent processing.

Delete. Snake vertices that are not covering any feature sample are deleted. This mechanism may remove an entire snake where a feature has become occluded, or may cut away portions of a snake in case of partial occlusion.

Trim. (Fig. 5.4a) Having just deleted portions of snakes that do not cover features, we now trim snakes in regions where parallel lines provide more coverage than needed. Several strategies might be possible, for example attempting to merge lines where they overlap. However, we have found a simple policy that trims lines at their endpoints to be effective for achieving unit coverage. Specifically, the three criteria for removing the endpoint v_i from a snake are that it is too close to a vertex



(a) Split (b) Merge

Figure 5.5: Split and merge operations modify snake connectivity. (a) When a snake bends too sharply around a corner it is split. (b) When two snakes abut smoothly they are joined.

v_j of a different snake, they are close to parallel, and that by removing v_i we will not leave a feature uncovered that would otherwise be covered by v_i:

$$c_{\text{trim}} \equiv \begin{cases} \|v_i - v_j\| & < & r_{\text{trim}} \\ |\vec{t_i} \cdot \vec{t_j}| & > & \theta_{//} \\ \text{footprint}(v_i) & \subseteq & \cup_{k \neq i} \text{footprint}(v_k) \end{cases}$$

where $\vec{t_i}$ and $\vec{t_j}$ are their respective tangents, and r_{trim} and $\theta_{//}$ are threshold constants for how close and parallel these parts of the snakes may become before trimming occurs.

Extend. (Fig. 5.4b) Next we extend snakes at every endpoint where doing so will offer coverage for otherwise uncovered feature points whose tangents align locally with the snake. Specifically, beyond the current endpoint v_i we will consider appending a new endpoint v_j at the location of every uncovered feature sample, based on four criteria:

$$c_{\text{extend}} \equiv \begin{cases} \neg\ \text{covered}(v_j) & & \\ \|v_j - v_i\| & < & r_{\text{extend}} \\ \vec{t_i} \cdot \vec{t_j} & > & \theta_{//} \\ |\vec{t_j} \cdot \vec{T_j}| & > & \theta_{//} \end{cases}$$

where $\vec{t_i}$ and $\vec{t_j}$ are the tangents at the snake vertices, $\vec{T_j}$ is the tangent of the feature sample, and r_{extend} is a search radius.

In general there may be many candidate feature locations for which the extension criteria above are satisfied. Therefore, we compute a goodness score g_j that describes how well each candidate extends the snake and then choose the candidate with the highest score. The score favors aligning the tangents and a relatively short extension, as follows:

$$g_j = \alpha \left(\frac{\min(\vec{t_i} \cdot \vec{t_j}, |\vec{t_j} \cdot \vec{T_j}|) - \theta_{//}}{1 - \theta_{//}} \right) + (1 - \alpha) \left(\frac{r_{\text{extend}} - \|v_j - v_i\|}{r_{\text{extend}} - r_{\text{cover}}} \right)$$

where the two terms in parentheses have been normalized to the range $[0, 1]$ and α is a parameter that balances between them.

Split. (Fig. 5.5a) To accurately track sharp features of a 3D model, a snake is split at vertex v_i if a sharp angle appears there, found by:

$$c_{\text{split}} \equiv \overrightarrow{t_{i-1}} \cdot \overrightarrow{t_{i+1}} < \theta_{\perp}$$

where $\overrightarrow{t_{i-1}}$ and $\overrightarrow{t_{i+1}}$ are the tangents at two neighboring vertices and $\theta_{\perp}$ is a threshold constant for how much the snake is allowed to bend at a vertex.

Merge. (Fig. 5.5b) To simplify the snake topology, two snakes are merged if they are locally parallel and their endpoints are close:

$$c_{\text{merge}} \equiv \begin{cases} \vec{\tau}_i \cdot \vec{t}_j & > & \theta_{//} \\ \vec{\tau}_j \cdot \vec{t}_i & > & \theta_{//} \\ \|v_j - v_i\| & < & r_{\text{merge}} \end{cases}$$

These criteria may lead to multiple conflicting candidate mergers, for instance where three endpoints meet in a Y shape. Therefore, in a fashion similar to the extension process described above, we assign a goodness score that favors straightness and closeness:

$$g_{ij} = \alpha \left(\frac{\min(\vec{\tau}_i \cdot \vec{t}_j, \vec{\tau}_j \cdot \vec{t}_i) - \theta_{//}}{1 - \theta_{//}} \right) + (1 - \alpha) \left(\frac{r_{\text{merge}} - \|v_j - v_i\|}{r_{\text{merge}} - r_{\text{cover}}} \right)$$

where v_i and v_j are the endpoints of two distinct contours, $\vec{t}_i$ and $\vec{t}_j$ their tangents, and $\vec{\tau}_i$ and $\vec{\tau}_j$ are tangents of hypothetical merger segments, shown in Fig. 5.6b. We perform as many non-conflicting mergers as possible, in order of descending score.

Insert. This operation works like the initialization step described above, except that it occurs after all the other operations. Its role is to insert new snakes where new features have appeared that have not been addressed by, for example, extending existing snakes.

3.2. Practical Considerations

Threshold constants. The scope of action of the four operations that trim, extend, split and merge the snakes are given by threshold constants for various radii and angles. Several relationships between them are important. In order for the snakes to establish proper coverage between samples: $r_{\text{cover}} > \delta_{\text{max}}/2$. In order to permit extension to occur: $r_{\text{extend}} > r_{\text{cover}}$. In order to enforce hysteresis between split and merge events: $\theta_{\perp} < \theta_{//}$. Other constants trade off between accuracy and performance. In our experiments we have found the values effective for the full range of results shown: $r_{\text{cover}} = 10\text{px}$; $r_{\text{trim}} = 5\text{px}$; $r_{\text{extend}} = 20\text{px}$; $r_{\text{merge}} = 20\text{px}$; $\theta_{\text{c}} = \theta_{//} = 0.85$; $\theta_{\perp} = 0.5$; $\alpha = 0.8$.

Acceleration structure. During advection and coverage updates, we often need to access the neighbors of snakes vertices or feature samples in the coverage radius. To expedite local search, we use the data structure of Delingette et al. [DM00]: a randomized sparse regular grid (implemented by a hash table) containing all the snake vertices and feature samples. With a grid cell size of $2\, r_{\text{cover}}$, we enumerate all points in a disc of radius r_{cover} by walking four cells.

4. Brush paths Shape and Parameterization

Snakes approximate and track the underlying feature lines from frame to frame. Based on this temporally coherent layer, we build a set of brush paths that will be responsible of providing the final shape and position of the strokes (Sec. 4.1) and computing a temporally coherent parameterization (Sec. 4.2).

4.1. Brush path Geometry

A brush path is a 2D polyline with vertices associated 1 – 1 to a sequence of vertices on a single snake (Figure 5.3). The position of brush path vertices are described relative to their associated snake vertices, to take advantage of the snake coherence during motion. The user controls a target length l_{max} (potentially infinite) for the brush path allowing for short brush paths if desired. In addition, brush paths can have "overshoot" that extends their end segments.

Smooth Fading. In circumstances such as rapid camera movements, feature lines may abruptly appear or disappear. Because snakes persist across frames, we can smooth out these events by adding a lifespan to each snake vertex, as follows. Each snake vertex has a lifespan κ, initially null. At each frame, if a vertex persists, its lifespan increases: $\kappa = \max(\kappa + 1, \kappa_{\max})$ with $\kappa_{\max}$ a constant defined by the user (we typically use 3 in our experiments). Similarly, if a vertex disappears its lifespan decreases ($\kappa = \kappa - 1$).

This lifespan is transmitted to the brush paths vertices and can be mapped to different style attributes, like the opacity or the thickness of the line. That way brush strokes are appearing and disappearing progressively during $\kappa_{\max}$ frames.

Shape Abstraction. The shape of the brush paths can deviate substantially from the features shape of its corresponding snake, allowing to produce very abstract styles. As a proof of concept we show examples of extreme abstraction by drawing with straight brush paths or circular arcs (Fig. 5.6).

Drawing with straight brush paths implies to partition each snake in a set of approximately linear parts. Posing this mathematically, we seek the lowest "cost" measured as the sum of squared distances between the snake vertices and their corresponding brush path, together with a constant extra cost per sequence. We solve this optimization problem using dynamic programming, following an approach similar to that of McCrae and Singh [MS09] for fitting sketched clothoid curves.

At each frame, a set of straight brush paths is propagated with the snakes from the previous frame. However, the underlying shape of the snakes may have evolved and therefore the we may need to consider breaking a straight path into two or more segments to better match the new shape. So we apply the same dynamic programming approach used initially for the snake path to each brush path in subsequent frames. In addition, wherever the policy described in Sec. 4.2 might consider merging brush paths, we add an additional test to verify that the dynamic programming solution would not immediately split the merged path.

Conveniently, the abstraction algorithm for straight brush paths is easily adapted for circular arcs (and other primitives, though we have only tried these two). The

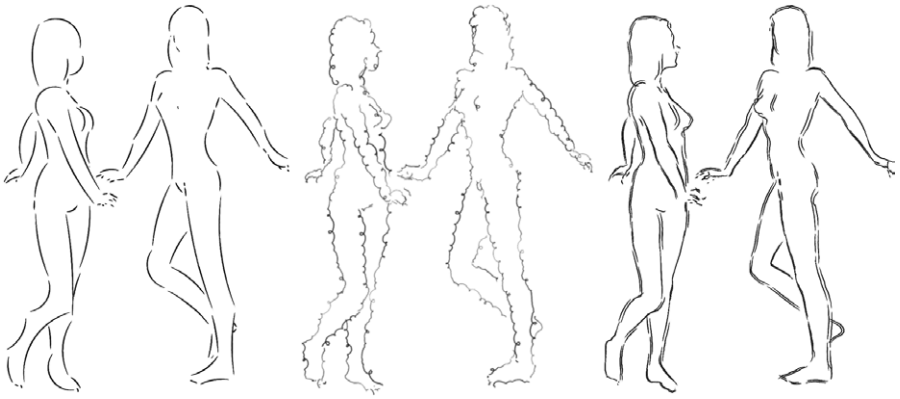


Figure 5.6: Stylized woman in two poses and three styles: arcs, loopy offsets, and overdrawn.

main difference is that in determining a best arc and fitting error for a sequence of vertices we use the "modified least squares" method of Umbach and Jones [UJ03].

Abstraction smoothing. When using strong abstraction there are sometimes annoying temporal artifacts, for example a circular arc suddenly switching from concave to convex, or single segment suddenly breaking into two.

Therefore, we optionally add a smoothing operation to ease such transitions, as follows. Suppose the smoothed position of a path vertex at frame i is s_i. In frame $i+1$, rather than immediately sending the vertex to its target position t_{i+1} we move it a step in that direction $\vec{d} = (t_{i+1} - s_i)$:

$$s_{i+1} = s_i + \min(1, d_{\max}/\|\vec{d}\|)\,\vec{d}$$

where $d_{\max}$ specifies a maximum step size. Note that s_i and t_i are represented as offsets relative to the corresponding snake vertex, so we smooth these abstraction offsets but not the underlying motion of the model. We smooth just the endpoints of segments, whereas for arcs we smooth all vertices (meaning that the latter shape will deviate from exact circles during smoothed transitions).

4.2. Brush path Parameterization

We need to parameterize these paths, and do so with temporal coherence during animation. Snake track features moving in 3D and can transmit their parameterization from one frame to the next, but it will no longer correspond to arc-length. Mapping a stroke texture using this parameterization would create sliding or compression artifacts. To avoid them, we linearize the parameterization at each frame based on previous frame, and we use *self-similar line artmap* (SLAM) textures (Chapter 3) to avoid compression when zooming. Moreover, to handle topological events – typically merging – we follow the approach of Kalnins et al. [KDMF03] and allow multiple brush paths per snake as long as their parameterization doesn't match.

Parameterization fitting. Similarly to the 2D coherence strategy of Kalnins et al. [KDMF03] and the parameterization scheme described in Chapter 4, our assumption is that a brush path represents the interaction between a drawing tool and 2D paper. Thus it should be uniform in screen-space, implying that its parameterization T is linear with respect to arc-length s: $T(s) = \rho s + \phi$. T should also evolve according to the motion of the underlying snake. To account for these two goals,

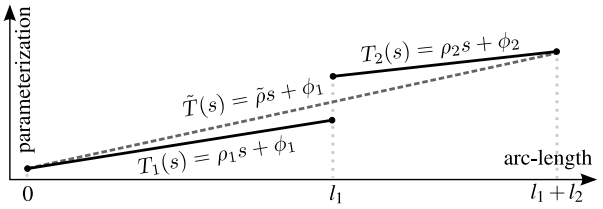


Figure 5.7: The mean parameterization $\tilde{T}$ of two brush strokes T_1 and T_2 is defined as the line joining their start and end point in parameterization versus arc-length space. The leveling process pushes the parameterization of the neighboring brush strokes toward this mean.

we follow the approach of Kalnins et al., a linear least squares fit of T, based on the parameter values from the previous frame propagated to the brush path of this frame (via the associated snake).

However, we have an extra degree of freedom that was not available in their approach. The use of SLAM textures allows us to handle gracefully changes in both phase and slope to prevent sliding artifacts, especially when zooming. Note that we do not need a fitting algorithm robust to outliers, like the use of *RANSAC*, because the parameterizations of multiple brush paths are not mixed.

Topological events. Brush paths rely on the connectivity of their associated snakes, so when a snake is updated so are its brush paths. For example, snake resampling (Sec. 2.3) causes resampling of brush paths on that snake. When a snake is split (Sec. 3.1), the brush paths covering the break are also split; and likewise for trimming, extension, and deletion. However, when two snakes merge, their brush paths are not merged automatically, as they may differ in position and parameterization. Instead, a new merging operator tests that the paths are continuous both spatially and in parameter value. If not, both paths remain (Fig. 5.3). Otherwise, the two brush paths are joined and their mutual parameterization becomes:

$$\tilde{T}(s) = \tilde{\rho} s + \phi_1 \quad \text{where} \quad \tilde{\rho} = \frac{\rho_2 l_2 + \phi_2 - \phi_1}{l_1 + l_2}$$

To avoid brush path fragmentation over time, we also propose a *leveling* mechanism, similar in spirit to the "healing" process of Kalnins et al. [KDMF03]. It progressively pushes the parameterization of two reasonably close brush paths to their common mean $\tilde{T}$, encouraging a merger in subsequent frames (Fig. 5.7). Note that we can explicitly deal with the periodicity of the parameterization during this process, which increases its efficiency.

5. Results

Snakes and their accompanying brush paths together provide a robust, temporally coherent, evenly parameterized set of paths in screen space. These paths allow us to explore stylization possibilities not available previously for animated 3D scenes of moderate complexity.

5.1. Style Examples

We can produce standard drawing styles by using SLAM textures together with stroke attributes such as color and thickness. Tapering [NM00] is provided and can be computed per brush path or per snake.

Offsets. We can also apply displacement mapping to the vertices of each brush path, exploiting the fact that brush path geometry is persistent across frames. This mechanism supports, for example, the loopy effect added to the knot (Fig. 5.8) or woman (Fig. 5.6). Kalnins et al. [KMM*02] used screen space offsets on brush path geometry, synthesized using Markov random fields. In contrast, we use a variation on the SLAM synthesis to provide similar details at varying scales.

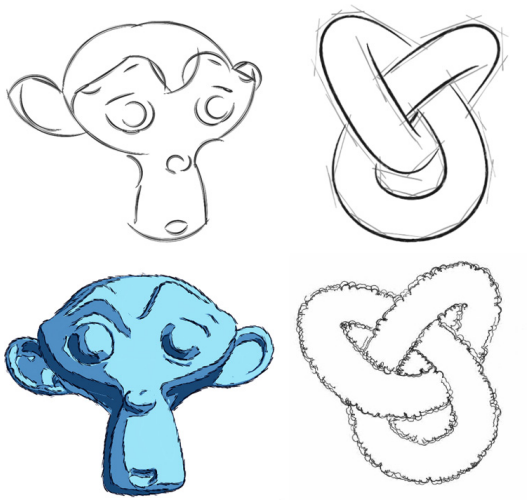


Figure 5.8: Stylization examples, from left to right: monkey with circular arcs (above) and fuzzy SLAM texture over toon shading (below); knot with construction lines (above) and overdrawn wiggles (below).

Overdraw. The unit coverage computation at the snakes not only prevents line clutter but also supports a form of "overdrawing," i. e. the overlap of a controlled number of brush paths on the same snake. The user specifies a target *overdraw* $\Omega_t \in \mathbb{R} > 1$. We measure the current overdraw Ω_c between two snake vertices as the number of brush paths overlapping this edge. For every brush path endpoint, we stochastically decide whether to extend, trim, or leave it unchanged, based on the probabilities:

$$\begin{aligned} P(\text{extend}) &= \max(0, \gamma_{\text{extend}}(1 - \Omega_c/\Omega_t)) \\ P(\text{trim}) &= \max(0, \gamma_{\text{trim}}(1 - \Omega_t/\Omega_c)) \end{aligned}$$

where γ_{extend} and γ_{trim} weight how rapidly the overdraw moves towards its target during animation.

We can create even sketchier styles by combining overdraw with a small target length and random offsets per brush paths.

Motion Effects. Brush paths are persistent across frames and have their own linear parameterization. We take advantage of these two properties to produce motion and speed lines (Fig. 5.10) inspired by lines appearing in hand drawn comics.

These styles reinforce the motion of the depicted scene by adding visual elements at locations undergoing noticeable motion. Similar effects have been described in the literature: motion lines added to static line drawings [MSS99] and stroboscopic and other 3D effects for animated 3D models [BZOP07, SSBG10], but this is the first opportunity to make motion lines appear over animated line drawings in an interactive setting.

Our goal is to apply these effects to portions of the drawing corresponding to the "trailing edge" – that is, regions of the brush paths where the velocity $\vec{v}$ has a strong component opposite the projected surface normal. So at every brush path

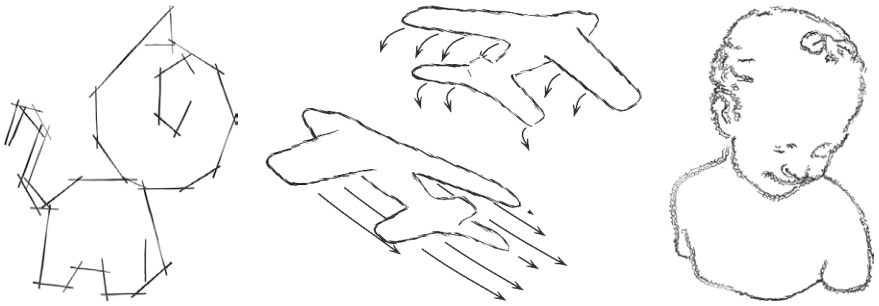


Figure 5.9: Additional examples with, from left to right, straight brush paths, speed lines and overdraw.

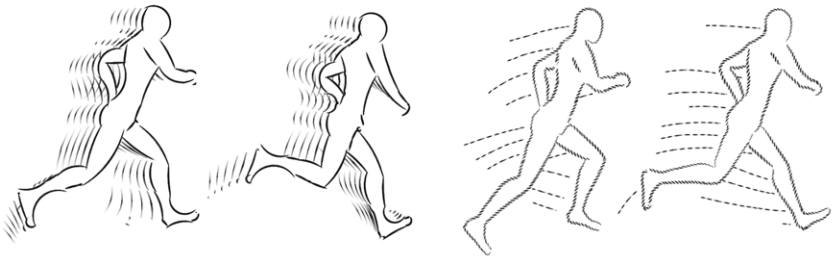


Figure 5.10: Motion lines (left) and speed lines (right).

vertex we compute $V = -\vec{v} \cdot \hat{n}$, and our goal is to find regions where this value is large. But how large? To answer this question we compute the mean μ_V and standard deviation σ_V of V over all the vertices. Then at each vertex we check to see if $V > t\ \sigma_V + \mu_V$ for a moderate threshold t that controls how much of the model receives these effects (we use $\frac{1}{2}$). Vertices that pass this test are considered potential "seed points" for motion lines or speed lines.

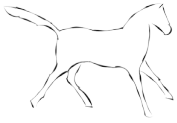
Motion lines are new brush paths created by copying portions of existing brush paths that contain consecutive sequences of seed points. On the other hand, speed lines are new brush paths formed by chaining together copies of seed vertices over a series of frames. In either case, the depth of the new paths is taken to be the depth at the seed points, which means that these effects can be occluded by the model itself. These paths can either remain fixed in the screen for a stroboscopic effect or have a velocity imparted by the object or scene. Finally, the new brush paths can be rendered using the effects described earlier. We generally find that it helps to fade out motion lines over a period of time, and to trim the ends from speed lines after a given length.

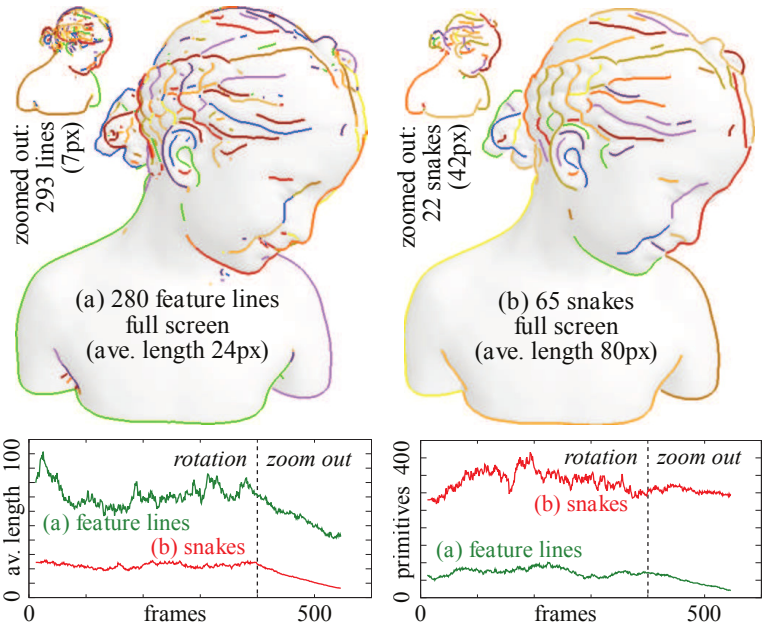
5.2. Performance

Our implementation is primarily CPU-based, except for the line visibility computation, blur for relaxation, and final rendering which rely on the GPU. The system performs at interactive rates on a recent machine (Intel Core 2 at 2.40 Ghz with 4 GB of RAM and a GeForce GTX 260) for models of moderate complexity (ranging from 10 to 25 FPS on our examples). A major bottleneck is the readback from the GPU to CPU, which is performed at several stages. Techniques such as stream compaction [Hor05] could be used to reduce this cost. The sparse matrix inversion required for the relaxation step is performed efficiently using the CHOLMOD library [CDHR08].

5.3. Comparison with Previous Approaches

The results produced by this method are similar to the methods of Kalnins et al. [KDMF03] and our robust votes propagation scheme (Chapter 4) for simple styles and models such as a torus. For moderately complex models, however, our approach produces longer and more stable brush paths. The difference is particularly noticeable on complex shapes like the Bimba, where contours are broken into small pieces (Fig. 5.11) and multiple line types (e. g., contours and suggestive contours) can be uniformly processed by the snakes.





(c) Evolution of the number of primitives and their mean length

Figure 5.11: Statistics of silhouette and suggestive contour lines, recorded during a full rotation followed by zooming out. The snakes are longer and less numerous than the input feature lines. These qualities support rendering long, connected strokes with improved coherence, and suppress line clutter. During the zooming out phase, both feature lines and snakes grow shorter (left), but the snakes also coalesce while the number of input lines remains high (right). To remove clutter it would be possible to omit short lines from (a) but significant features composed of short pieces would be lost.

The previous methods rely on visible stretches of object space feature lines as input, and their brush paths can not be longer than these features (Fig. 5.11a). In contrast, one snake may extend across multiple feature lines. This property significantly reduces the overall number of snakes while increasing their mean length (Fig. 5.11b). During a rotation around the model the number of snakes remains relatively stable, even though the number input features varies substantially (Fig. 5.11c). On average the snakes are three times longer than the feature lines, consistent with our *length* goal. When zooming out, the number of features lines remains relatively constant. The number of snakes, however, drops as each snake begins to straddle multiple feature lines projecting in the same region in 2D, maintaining our *coverage* goal.

The impact of brush path length on the final rendering can be seen in Fig. 5.12. The short, unpredictable paths of the silhouette segments are clearly visible, while the snakes provide long brush paths that correspond with the major shape features of the model.

An alternative solution to our snake-based approach could be to chain the small pieces of line together with a more involved chaining algorithm (e. g., [GTDS10]) before computing coherence using Kalnins et al. method [KDMF03]. However,

(a) Kalnins et al. [KDMF03] (b) our approach

Figure 5.12: The Jot system of Kalnins et al. (a) generates many short strokes because the visible silhouettes produce short brush paths. Our approach (b) recovers connectivity using snakes, leading to fewer, longer strokes.

nothing guaranties that interactive performances might be achievable with such an approach. Moreover, it won't be able to handle fully disconnected image space line samples.

6. Discussion

■ *Line coherence in image space is a key assumption of our approach.*

Our method relies strongly on an assumption that input feature lines are coherent in image space. However, this may not always be the case, as discussed by DeCarlo et al. [DFR04]. Methods that can "smooth" the input to our system would improve the temporal coherence of the output. Note that such smoothing methods are orthogonal to our contribution and have been addressed by previous work in various ways (e. g., [NM00, DFR04]).

Our method is robust in part because it only enforces a piece-wise linear parameterization, which is supported by the distinction between snakes and brush paths. Each brush path uses a linear arc-length parameterization to which we can apply SLAM textures. Thus, one limitation of our system is that it does not support the tradeoff described by Kalnins et al. [KDMF03] between two competing parameterization goals: linear in arc-length vs. tracking in 3D. This limitation is not fundamental to the snakes approach but rather to our implementation.

Another limitation of our approach is that for efficiency, decisions about snakes and brush paths are made locally. There is no explicit mid- or high-level understanding of the scene, which an artists would leverage in making such animations by hand. For example, tracking T-junctions might be a way to improve the behavior at occlusions. This work focuses on feature lines that are view-dependent and extracted from organic shapes, rather than fixed features such as those found in architectural scenes. Nevertheless, artists deliberately treat structural relationships like a repeating pattern of windows on a building, whereas our system has no mechanism to treat such features in concert.

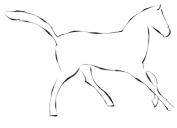
Our pipeline is generic enough to be applied on video, by using vision algorithms to extract an approximated version of the data that we need: optical flow, feature lines, and depth. This endeavor would be particularly timely with the advent of commodity cameras that supply depth and optical flow along with color information.

■ *Our local optimization would benefit from predictions of the future.*

Finally, while this system cannot optimize globally over an entire animation sequence (since we assume the frames are not known in advance) it might be possible to anticipate one or two frames in the future using something like a Kalman filter on the camera path and then use this information to broaden the scope of the optimization.

Part II

Region Stylization



Summary

NPR methods, which aim at depicting shaded regions of 3D scenes with 2D media and patterns such as pigments or strokes, are faced with temporal coherence issues when applied to dynamic scenes. These issues arise from the difficulty to satisfy at best the three goals of flatness, motion coherence and temporal continuity.

In this part, we describe two methods making a different compromise between these goals. Our first approach favors motion coherence, whereas our second technique better preserves flatness. Both of them targets interactive visualization and performances, and allows to produce a wide range of patterns.

Note that our goal is not to reproduce a specific style, which explains why our results do not precisely mimic their real-world counterpart (e.g., handmade hatching and stippling). Instead, we want to provide a base layer that is temporally coherent, such that the artist can focus on style creation rather than on the dynamic behavior of the stylization.

6

DYNAMIC SOLID TEXTURES

We present in this chapter *dynamic textures*, a method that facilitates the integration of temporally coherent stylization in real-time rendering pipelines. Our method uses textures as simple data structures to represent media with characteristic patterns (eg. watercolor, charcoal, stippling) while ensuring real-time performances due to the optimized texture management of modern graphic cards.

Central to our technique is an *object space infinite zoom* mechanism that guarantees a quasi-constant size and density of the pattern in screen space for any distance from the camera. This simple mechanism preserves most of the 2D appearance of the medium while maintaining a strong temporal coherence during animation. The infinite zoom illusion can be applied to both 2D and 3D textures. However, we present our approach for 3D textures (the dynamic solid textures) which alleviate the need for complex computation or manual definition of 2D parameterizations over the 3D surfaces.

In order to demonstrate the effectiveness of our approach, we integrated it into the OGRE[1] game rendering engine. The performances measured in this environment indicate the low impact of the proposed method on framerate, even for complex 3D scenes.Various applications of our approach for the real-time coherent stylization of 3D animations are described in Chapter 8.

1. Object Space Infinite Zoom Mechanism

Drawing inspiration from the procedural noise function of Perlin [Per85] and the infinite zoom mechanism of Dynamic Canvas [CTP*03], our method relies on the texture *fractalization*. We define a *dynamic solid texture* as the weighted sum of n octaves Ω_i of the original solid texture. Each 3D object is then carved in such a solid texture, which naturally ensures a convincing feeling of zooming in and out: the texture will appear twice bigger when the object is twice closer to the camera.

■ *Self-similar textures are obtained by linearly blending octaves (doubled frequency) of a texture in a process called fractalization.*

But care must be taken to keep the texture elements at a quasi-constant size in screen space. To this purpose, we introduce the notion of *zoom cycle* that occurs every time the apparent size of the texture doubles. In that case, each octave is replaced by the following one and a new high frequency octave is created (Fig. 6.1). Note that the fractalization process introduces new frequencies in the texture, along with a loss of contrast, as discussed in Sec. 5. While reducing the number of octaves limits these artifacts, it also makes the apparition and disparition of texture elements more visible. Empirically we observed that $n = 4$ octaves is enough to deceive human perception.

[1] http://www.ogre3d.org

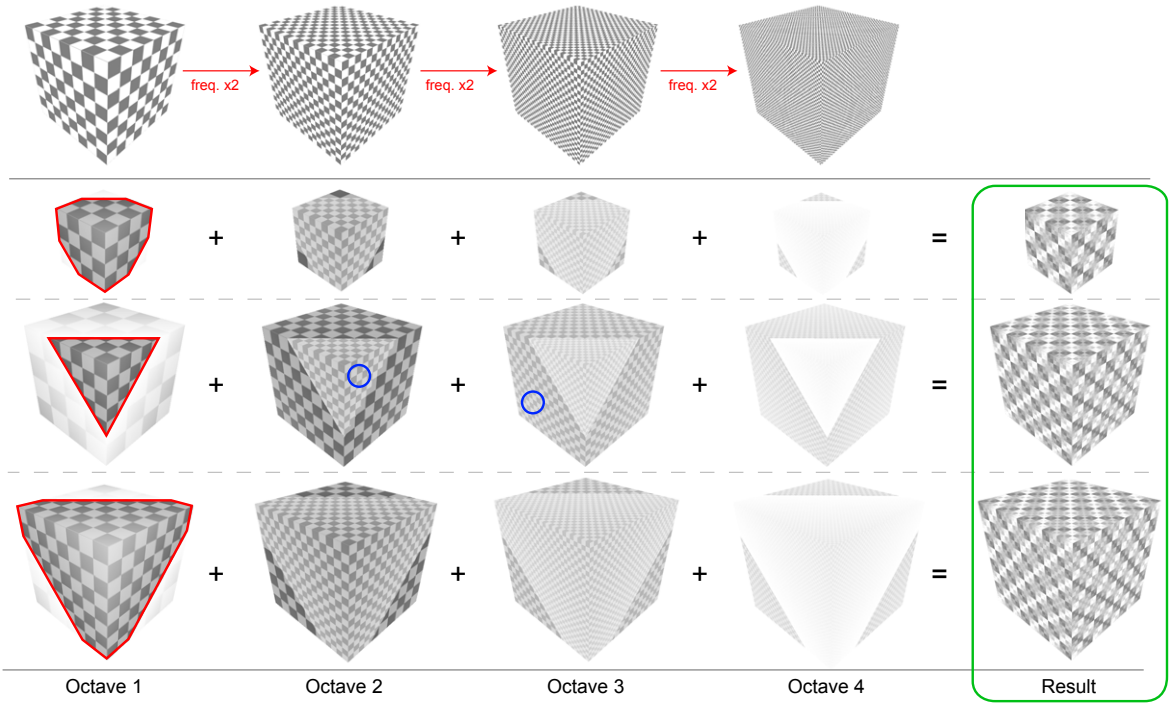


Figure 6.1: Tridimensional infinite zoom mechanism (using a solid checkerboard texture for illustration purpose). Observe the globally invariant frequency of the solid texture produced by our algorithm (last column). The red line delineates the boundary between two consecutive zoom cycles. Note the frequency similarity between two adjacent octaves for two consecutive zoom cycles (blue circles).

2. Fractalization Algorithm

In practice, an object is embedded in its dynamic solid texture by deriving texture coordinates from vertex coordinates in the local mesh frame. This object space texturing solves the Dynamic Canvas limitation of approximating an entire scene with a plane. For the sake of simplicity, we use only one cube of solid texture that we sample at different rates to retrieve all n octaves. For a vertex $p(x,y,z)$ at distance z_{cam} from the camera, the 3D texture coordinates (u,v,w) for the octave i are given by:

$$(u,v,w)_i = 2^{i-1}(x,y,z)/2^{\lfloor \log_2(z_{cam}) \rfloor}$$

In this equation, the 2^{i-1} term scales the sampling rate so that the texture retrieved for one octave is twice smaller than the texture for the previous octave. The $\lfloor \log_2(z_{cam}) \rfloor$ term accounts for the refreshing of the octaves at each zoom cycle: the distance z_{cam} can be decomposed in $log_2(z_{cam})$ zoom cycles, making $\lfloor \log_2(z_{cam}) \rfloor$ the indicator of how many times each octave has been doubled during the zoom.

During a zoom cycle, we modulate each octave with a weight $\alpha_{i=1...n}(s)$. These weights are dependent on s, the interpolation factor between the beginning and the end of the cycle:

$$s = \log_2(z_{cam}) - \lfloor \log_2(z_{cam}) \rfloor \in [0,1]$$

We must impose three constraints on the weights in order to ensure a smooth transition during the frequency shift (Fig. 6.2). First, to avoid the sharp appearance and

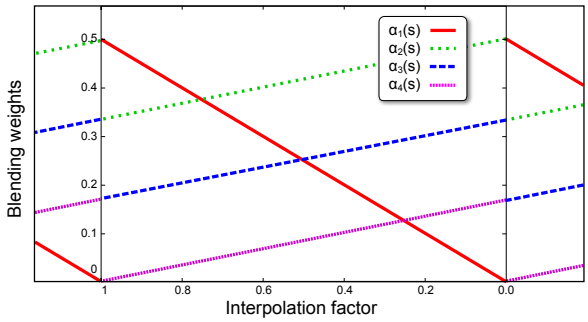


Figure 6.2: Evolution of the blending weights during a zoom cycle. Note the continuity at the beginning and the end of the zoom cycle, which ensures an infinite number of cycles.

disappearance of texture elements, the first octave should appear at the beginning of the cycle while the last should disappear at its end:

$$\alpha_1(0) = 0 \quad \text{and} \quad \alpha_n(1) = 0$$

Second, the weights of the intermediate octaves at the end of the cycle should be equal to the weight of the following octaves at the beginning of the next cycle:

$$\alpha_i(1) = \alpha_{i+1}(0) \quad \forall i \in \{1, \ldots, n-1\}$$

Finally, the weights should sum to 1 to preserve a constant intensity. In our implementation, we use a linear blending which is fast to compute and coherent with the linearity of the zoom. We choose the following weights:

$$\begin{array}{lllllll} \alpha_1(s) & = & s/2 & & \alpha_2(s) & = & 1/2 - s/6 \\ \alpha_3(s) & = & 1/3 - s/6 & & \alpha_4(s) & = & 1/6 - s/6 \end{array}$$

The full process is illustrated in Fig. 6.1. The red line highlights the frequency shift. It corresponds to the distance after which the zoom cycle restarts. Note the frequency similarity between the octaves i and $i+1$ for two consecutive zoom cycles. This correspondence ensures the texture continuity after the blending (last column).

3. Implementation Details

We have implemented this infinite zoom algorithm in GLSL[2] and integrated it in the rendering engine OGRE (see shaders source code in Appendix A). The *vertex shader* assigns the 3D texture coordinates of the vertex which can be the position of the vertices in the local object coordinate system (for deformable objects, this position in an undeformed pose of the object). The user can also specify an additional scaling factor to define the global size of the texture with regards to the depicted object. We compute the distance between a vertex and the camera as its z coordinate in the camera frame. Then the *fragment shader* blends linearly the four octaves of the solid texture for each pixel, according to the formula detailed in the previous section.

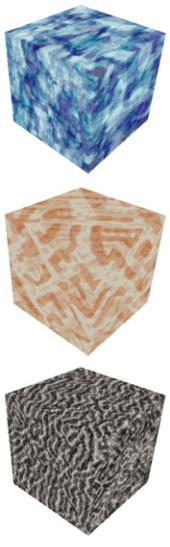
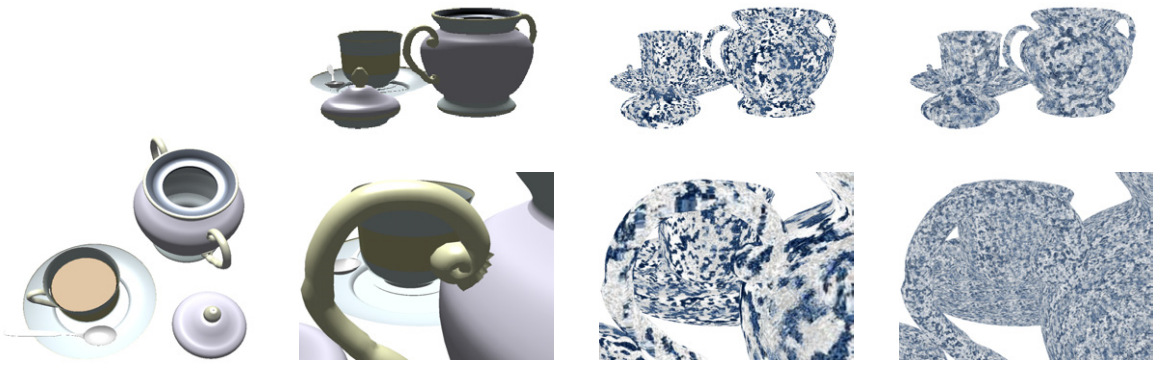


Figure 6.3: Raster dynamic solid textures

Raster solid textures (Fig. 6.3) are stored in the graphic card memory (typically $128 \times 128 \times 128$ RGB pixels, *i.e.* 6 MB without compression in *DirectDraw Surface* format). Procedural textures (Perlin noise [Per85, Ola05], for example) are evaluated on the fly in the fragement shader.

[2] OpenGL Shading Language: http://www.opengl.org/documentation/glsl



(a) Top view (b) Phong shading (c) 3D texture mapping (d) Dynamic solid texture

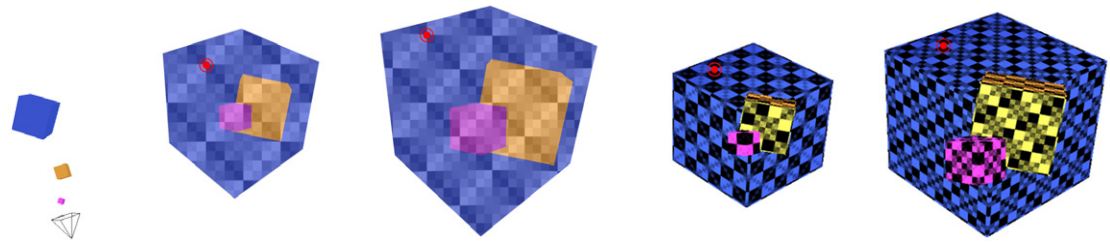
Figure 6.4: Comparison of our dynamic solid textures (d) with standard texture mapping (c). Regardless of the distance from the camera (top and bottom row), our method provides quasi-constant size of texture elements.

4. Comparisons

We compare in Fig. 6.4 our dynamic solid textures with traditional texture mapping. With a standard (meaning fixed scale) mapping, the size of texture elements varies with depth due to perspective projection. On the contrary, with our infinite zoom approach, texture elements keep a globally constant size in image space independent of the zoom factor.

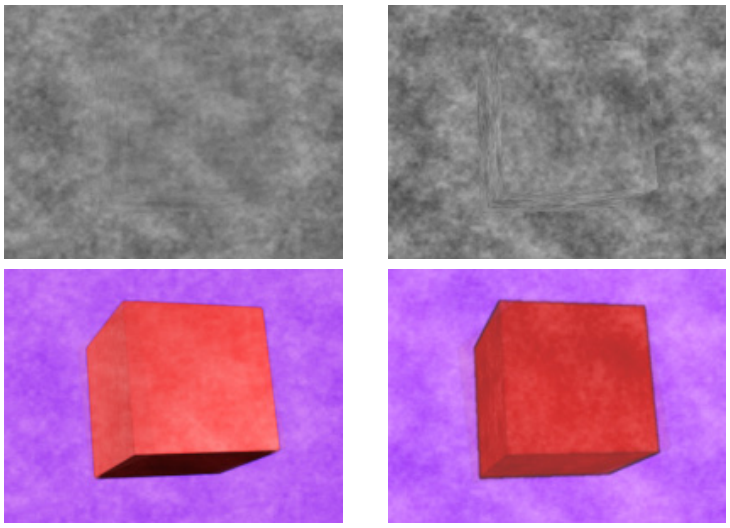
Compared with Dynamic canvas in Fig. 6.5, our approach suffers from perspective deformations when the surface is almost tangential to the viewing direction and from discontinuities at occlusion boundaries. However, these artifacts are limited by the infinite zoom mechanism that effectively reduces the apparent depth of the scene.

On the other hand, as highlighted by the red dot on Fig. 6.5, sliding effects occur with the Dynamic canvas approach. In our case, the texture elements follow perfectly the 3D motion of the scene. We refer the reader to the accompanying video of [BBT09] for a better illustration of these sliding effects compared to the accurate motion and convincing infinite zoom produced by our approach. Note that the approaches of Coconu et al. [CDH06] and Breslav et al. [BSM*07] would suffer from the same sliding problem.



(a) Top view (b) Dynamic canvas (c) Dynamic solid textures

Figure 6.5: Comparison with dynamic canvas: note the sliding of the texture with the dynamic canvas method, highlighted by the red dot.



(a) Texture advection (b) Dynamic solid textures

Figure 6.6: Comparison with texture advection: grayscale pigments (top) and watercolor stylization (bottom). Texture advection better preserves the 2D appearance of the pigments, but after stylization and during motion the difference with dynamic solid textures is much less noticeable.

We also compared our method with the *texture advection* approach of Bousseau et al. [BNTS07] (Fig. 6.6). Note that the bidirectional advection requires the knowledge of the entire animation sequence, which makes it unsuitable for real-time applications. However it produces animated watercolor with almost no noticeable sliding or deformations and thus gives a very strong 2D appearance on still frames.

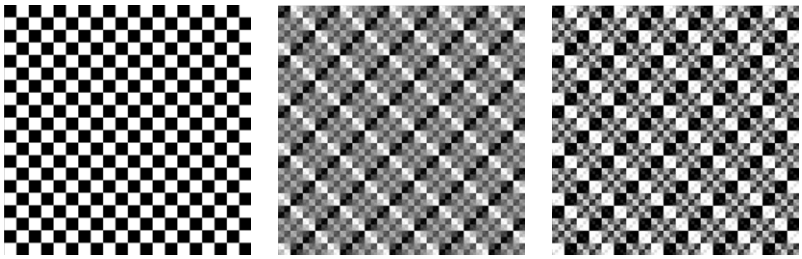
As can be seen on the video of [BBT09], the motion of the texture elements – perceived and interpreted as tridimensional by the observer – tends to conceal their 2D characteristics during an animation. Consequently we believe that the additional perspective deformation produced by our approach, although noticeable on still frames, is a small degradation compared to the vivid perception of perspective induced by the motion cues, particularly considering the performance gain.

The main advantages of our method are its simplicity of integration in existing rendering engines and its real-time performance. Our implementation in OGRE induces a small additional cost on the order of 10% in comparison with a traditional gouraud shading (computed in a shader). For a complex scene (135k tris) rendered at a resolution of 1280×1024 pixels, the framerate decrease from 70 fps to 65 fps with a 2.4GHz Core 2 Duo 6600, 4Go memory and a Geforce 8800 GT. This makes dynamic solid textures perfectly suited for real-time applications such as video games.

5. Discussion

Infinite zoom mechanism. The main limitation of our method, shared with Dynamic canvas [CTP*03], texture advection [BNTS07] and to some extent with mipmaps-based approaches [KLK*00, PHWF01, FMS01], is the linear blending of multiple octaves that creates new frequencies and induces a global contrast loss compared with the original texture. Self-similar textures, such as Perlin fractal noise, paper or watercolor textures, are not severly affected by this blending. However, more structured textures can be visually altered, as individual features tend to overlap (the checkerboard texture being an extreme case). The perceptual

■ *The linear blending of multiple octaves introduces new frequencies and induces a global contrast loss.*



(a) Original texture (b) Dynamic solid texture (c) Yu et al. blending

Figure 6.7: Comparison of our linear alpha-blending (b) with Yu et al. [YNBH10] blending (c) on a checkerboard texture. Their approach significantly reduces the contrast loss observed with dynamic solid textures.

evaluation described in Chapter 10 targets a more precise characterization and measurement of these effects.

As noted by Yu et al. [YNBH10], linear blending only preserves the mean $\hat{R}$ of the original texture but not its variance σ_R^2. Assuming that the variations of the blended textures R_i are independant variables and of a much higher frequency than the blending weight w_i, they demonstrate that the following formula (equation 4 of [YNBH10]) keeps both the mean and variance of the reference texture:

$$R' = \frac{\sum w_i(x)(R_i - \hat{R})}{\sqrt{\sum w_i^2(x)}} + \hat{R} \tag{6.1}$$

The loss in contrast is significantly reduced when using this equation to blend between the octaves of our infinite zoom mechanism (Fig. 6.7).

One possible solution to reduce these artifacts further would be to replace the linear blending by a salience aware blending in the spirit of the work of Grundland et al. [GVWD06]. This blending should better preserve the relevant features of the texture. The required salience data could be easily encoded in solid *feature maps* [WY04].

Alternatively, more advanced interpolation, blending or morphing techniques [MZD05, RLW*09, RSK10] could be extended to handle dynamically fractalized textures. The main issue is to ensure the visual quality and temporal continuity of the blending throughout the zoom cycle.

Texture mapping. We chose to develop the infinite zoom mechanism for solid textures because it avoids the need for an adequate 2D parameterization. However, the counterpart of this choice is that textures are decorrelated from the 3D surfaces. This can be seen as a limitation for styles that benefit from the orientation of texture elements along the surface. An example of such styles is pen-and-ink, for which it has been shown that orienting strokes along the principal directions of the surface emphasizes the shape of the objects [HJO*01, PHWF01].

■ *Solid textures are decorrelated from the surface and cannot be oriented along principal directions of curvature.*

Nevertheless, the infinite zoom mechanism described in this paper is independent of the texture dimension and can also be applied on 2D textures if the required parameterization is available.

A limitation shared by all texture-based approaches concerns the texturing of highly deformable objects. In this case, defining the texture coordinates as the vertices positions of the undeformed object creates additional dilatation/shrinking of the simulated medium, decreasing the perception of a two-dimensional look. Mark-based approaches – including the method we propose in the next chapter – does not suffer from this limitation.

7

NPR GABOR NOISE

In this chapter, we define a noise primitive that provides a trade-off for the temporally coherent stylization of any kind of 3D animation including deformable objects. By merging a large number of such primitives we produce a temporally coherent noise that exhibits the benefit of both mark-based and texture-based approaches.

Our new noise primitive is based on Gabor noise [LLDD09, LLD10]. We chose Gabor noise because it features precise local spectral control, which allows us to maintain the 2D aspect of the noise, and to create a wide variety of different patterns. By defining the noise on the 3D surface of the object, but evaluating it in 2D screen space, we ensure motion coherence without compromising flatness. In addition, we propose a new level of detail mechanism which guarantees temporal continuity.

NPR Gabor noise is then used in the same way as traditional procedural texture approaches to produce styles ranging from continuous (watercolor, ink) to discrete (hatching, painterly) as demonstrated in Chapter 8.

1. Gabor Noise

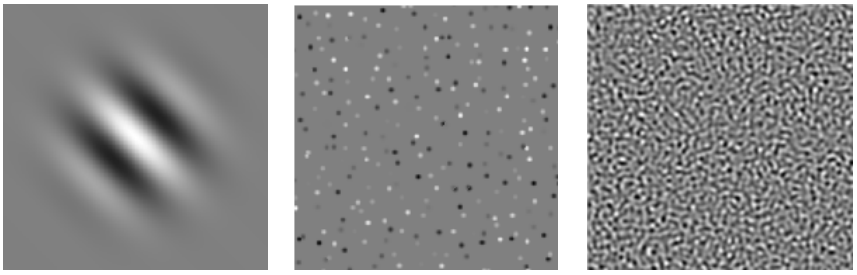
Gabor noise n is a sum of randomly weighted and positioned Gabor kernels (Fig. 7.1),

$$n(x,y) = \sum_i w_i g(a_i, F_{0i}, \omega_{0i};\ x - x_i, y - y_i), \tag{7.1}$$

where $\{w_i\}$ are the random weights, g is the Gabor kernel, and $\{(x_i, y_i)\}$ are the random positions. The Gabor kernel is the multiplication of a Gaussian envelope and a harmonic:

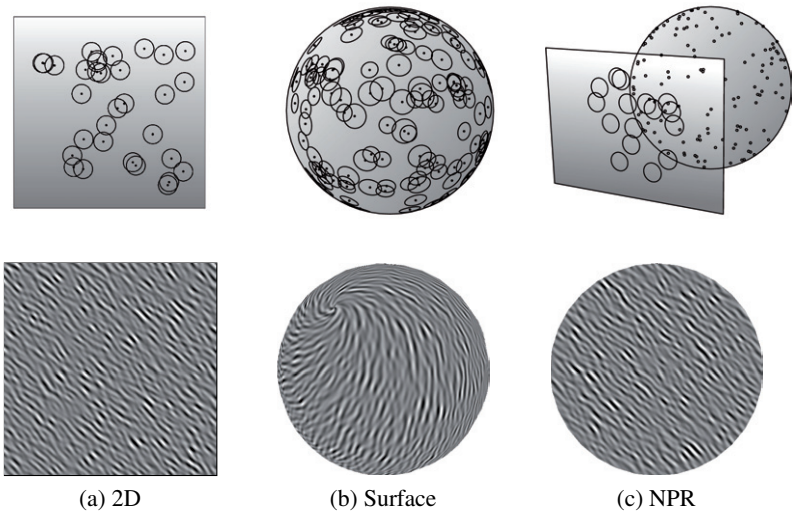
$$g(x,y) = Ke^{-\pi a^2(x^2+y^2)} \cos[2\pi F_0(x\cos w_0 + y\sin w_0)]. \tag{7.2}$$

The Gaussian envelope is parametrized by a bandwidth a, which determines the kernel radius r, and a magnitude K. The harmonic is defined in polar coordinates by a frequency F_0 and an orientation ω_0.



(a) Gabor kernel (b) Random positions and weights (c) 2D isotropic Gabor noise

Figure 7.1: Gabor noise is a sum of randomly weighted and positioned Gabor kernels featuring precise local spectral control. Taken from [LLDD09].



(a) 2D (b) Surface (c) NPR

Figure 7.2: (c) NPR Gabor noise has a 2D aspect, similar to (a) 2D Gabor noise, and moves coherently with the object (see video of [BLV*10b]), similar to (b) surface Gabor noise.

The random positions $\{(x_i, y_i)\}$ are distributed according to a Poisson distribution with mean λ. We call the random positions the point distribution, and λ the point density. Depending on how a_i, F_{0i} and ω_{0i} vary for different kernels, anisotropic, isotropic, or even more general kinds of noise are obtained.

The number of overlapping kernels is relatively large. The mean λ is expressed as $N'/\pi r^2$, where N' is the average number of overlapping kernels. The choice of N' is a speed versus quality trade-off, since both the evaluation time and the quality of the noise are proportional to N'. of N' range from 16 to 128.

2. NPR Gabor Noise

None of the existing kinds of Gabor noise addresses the problem of temporal coherence for NPR. 2D Gabor noise (Fig. 7.2a) is inherently 2D, and therefore behaves like the *shower door* effect, while surface Gabor noise (Fig. 7.2b) and solid Gabor noise are inherently 3D, and therefore behave like regular texture mapping. We present a new kind of Gabor noise, which we call NPR Gabor noise (Fig. 7.2c), that takes into account the problem of temporal coherence.

Gabor noise can be seen as a mark-based method, where the kernel is the mark. As for other mark-based methods in NPR, the basic principles of NPR Gabor noise follow from the goals formulated in Chapter 1, Sec. 2:

- To obtain a noise with a 2D aspect, we define the noise parameters F_0, a, ω_0 and λ in 2D screen space, and also evaluate the noise in 2D screen space.
- To ensure a coherent motion of the noise, we define the point distribution used by the noise on the surface of the 3D model.

Mark-based methods in NPR typically use splatting [KC05, BKTS06]. Noise based on kernels can be generated using splatting, as in [vW91, FW07], or procedurally, as in [Lew89, LLDD09], two conceptually opposite approaches. Procedural approaches are usually more general, while splatting is usually faster and maps better to the GPU. Since we target interactive applications, we use splatting.

3. Dynamic Point Distribution

Assuming a triangle model and a perspective camera, we first need to generate a Poisson distribution for our Gabor noise. We then need a way to guarantee temporal continuity of the noise. To achieve this while minimizing popping, we introduce an LOD mechanism which ensures a continuous and constant density of point distributions in 2D screen space throughout time. A key aspect of this LOD is the blending weight, which we determine based on the statistical properties of the noise.

3.1. Poisson Distribution for NPR Gabor Noise

In contrast to most few-mark methods in NPR, which require a Poisson-disk distribution, Gabor noise only requires a Poisson distribution. We obtain this distribution per triangle using a method inspired by [KC05] and [LLDD09].

■ *Gabor noise only requires a Poisson distribution, not a Poisson-disk distribution.*

We generate the points in a triangle using a pseudo-random number generator (PRNG), with a different seed for each triangle to ensure a random point distribution. We use the PRNG to generate uniformly distributed barycentric coordinates that correspond to the random kernel positions $\{(x_i, y_i)\}$. Since points in a Poisson distribution are independent, these positions are guaranteed to follow a Poisson distribution over edges, even if they are generated per-triangle. We always use the same seed for the same triangle, which ensures temporal continuity during the animation.

If the expected number of points N on the triangles is less than 1, both generating a point or not will result in an incorrect point density. We solve this problem by generating a point and weighting it by N to account for this discretization error.

3.2. Dynamic Point Distribution using LOD

Since the density of the point distribution is constant in 2D screen space, the number of points on a triangle in 3D object space should vary. More specifically, the required number of points N on a triangle is λA, where A is the 2D screen space area of the triangle. Note that N changes when A changes, for example, when the triangle moves away from the viewer. This means that NPR Gabor noise requires a dynamic point distribution, i.e., a distribution that changes over time.

One way to achieve this would be to add and remove individual points in a temporally consistent manner, to maintain the required point density λ in 2D screen space, similar to [KC05, BKTS06]. Unfortunately, we have experienced that, with our noise, this fine-grained level of control can still lead to popping, since a single kernel can still appear relatively fast.

Instead, we halve or double the number of points, and weight the second half of the points to account for this operation. More specifically, rather than using N points, we use $2M$ points, where:

$$M = 2^{\lfloor \log_2 N \rfloor} \tag{7.3}$$

and we weigh the first M points with a weight of 1, and the second M points with a weight of α. Note that M and $2M$ are the powers of 2 that bracket N. This coarse-

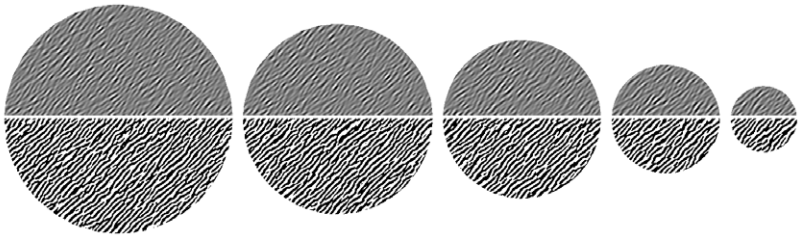


Figure 7.3: Zoom sequence on a sphere textured with our NPR Gabor noise before (top) and after binary threshold (bottom). At each level of the zoom, the statistical properties of the noise are well preserved thanks to our LOD mechanism.

grained level of control further reduces popping, since kernels now appear at a slower rate.

3.3. Choice of Blending Weight using Statistical Properties

Since Gabor noise is a summation, we can express the noise n_{2M} based on $2M$ points as a linear interpolation between two noises n_M and n'_M based on M points:

$$n_{2M}(x,y) = n_M(x,y) + \alpha n'_M(x,y). \tag{7.4}$$

Therefore, it might seem as if we just trade popping artifacts for blending artifacts, in a manner similar to texture-based methods in NPR. However, this is not the case, since we do not affect the appearance of the noise (Fig. 7.3). Our approach preserves the statistical properties of Gabor noise because we blend between two noises with the same parameters. In contrast, most existing approaches in NPR that blend between two noises blend noises with different parameters, for example two octaves of Perlin noise, which have a different principal frequency.

We derive a weighting function α that preserves the statistical properties of the noise n_M, based on the power spectrum and the variance. We need to determine the power spectrum and variance of n_{2M} in terms of n_N.

Power spectrum. First, we take the magnitude of the Fourier transform of both sides of Eqn. 7.4. Then we simplify, noting that the Fourier transform is a linear operator, and that n_M and n'_M have the same expected Fourier transform,

$$|\mathcal{F}(n_{2M})|^2 = (1+\alpha^2)|\mathcal{F}(n_M)|^2. \tag{7.5}$$

Note that with expected Fourier transform we mean the Fourier transform of the stochastic processes corresponding to n_M and n'_M rather than the Fourier transform of specific instances of n_M and n'_M. Finally, we apply equation 10 of [LLDD09], noting that $\lambda_M = M/A = M/N\,\lambda_N$,

$$|\mathcal{F}(n_{2M})|^2 = (1+\alpha^2)\frac{M}{N}|\mathcal{F}(n_N)|^2. \tag{7.6}$$

This establishes a relation between the power spectrum of the noises n_N and n_{2M}.

Variance. First, we take the variance of both sides of Eqn. 7.4. Then we simplify, noting that n_M and n'_M are uncorrelated and have the same expected variance,

$$\sigma^2_{2M} = (1+\alpha^2)\sigma^2_M. \tag{7.7}$$

Finally, we apply Eqn. 10 of [LLDD09],

$$\sigma_{2M}^2 = (1+\alpha^2)\frac{M}{N}\sigma_n^2. \tag{7.8}$$

This establishes a relation between the variance of the noises n_N and n_{2M}.

From Eqn. 7.6 and Eqn. 7.8 we see that n_{2M} preserves the variance and the power spectrum of n_N when $(1+\alpha^2)M/N$ equals 1. We obtain the weighting function α by solving this equation for α and substituting Eqn. 7.3:

$$\alpha(N) = \sqrt{\frac{N}{2^{\lfloor \log_2 N \rfloor}} - 1}. \tag{7.9}$$

4. Implementation

Since we target interactive applications, we implement NPR Gabor noise efficiently on the GPU, using geometry and fragment shaders.

We use a geometry shader to generate point sprites on-the-fly from incoming triangles. It is important for performance to discard invisible point primitives before rasterization. Therefore, we do not generate primitives for back-facing and invisible triangles. We resolve visibility using a depth test, or using a per-object ID buffer and a per-triangle ID buffer. In order to avoid popping, we use a *fuzzy* depth test, such as a blurred depth test [LD06] or a partial visibility test [CF08] (see Sec. 5). The number of point primitives that can be emitted by the geometry shader for each triangle is limited (in our setup to 64). We work around this limitation by subdividing large triangles, currently in a manual preprocess. Future GPU's with relaxed hardware limitations and the programmable tessellation unit will most likely remove this limitation.

We use a fragment shader to perform per-fragment visibility tests and to evaluate the Gabor kernel. The visibility tests include an object ID test, a triangle ID test, and a potentially blurred depth test. Each test affects the appearance of the noise, and can be enabled or disabled by the user. The object ID test, triangle ID test and depth test reject fragments based on object ID, triangle ID and depth, trimming the noise to object boundaries, triangle boundaries and depth edges.

Our GPU implementation runs at interactive rates, ranging from 8 to 50 fps for models of moderate complexity (4k to 50k tris), using off-the-shelf hardware (Intel

Figure 7.4: Rotation around a pitcher textured with our NPR Gabor noise before (top) and after binary threshold (bottom). The 2D appearance of the noise is well preserved. See the accompanying video of [BLV*10b] to better assess the temporal continuity and motion coherence.

Core 2 Duo 2.4GHz CPU and GeForce GTX 260 graphics card). Our GPU implementation scales well with scene complexity. Although every triangle is processed by the geometry processor, the total number of point sprites is approximately constant. Our GPU implementation supports anti-aliasing using super-sampling for higher image quality at the expense of a lower frame rate.

5. Discussion

Popping. The GPU implementation of our noise primitive based on splatting is subject to residual popping artifacts, which may be reinforced by the stylization. This popping is caused by aliasing in the per-point-sprite visibility tests, due to the mismatch between the continuous point sprite position and the discretized visibility. This problem is typically alleviated using *fuzzy* visibility tests, such as the blurred depth test [LD06]. We use a partial visibility test based on the work of Cole and Finkelstein [CF08]. It works relatively well in practice, although some popping may remain according to the mesh and style. Moreover, partial visibility avoids binary visibility decisions at (dis)occlusion boundaries, which reduces the residual motion of the pattern near silhouettes.

■ *Remaining popping artifacts should be removed using a procedural computation of the noise at the price of interactivity.*

We believe it might be possible to completely remove the remaining popping artifacts by using a procedural approach, as in [LLDD09]. However, initial experiments were not encouraging in terms of performance. Because we target interactive rates, which is especially important for the artistic design process, we believe that a splatting approach is currently more suited for NPR Gabor noise. However, it would be interesting to develop such a high quality offline approach and use the current method as a previsualization tool. Appropriate guarantees on similarity of visual results must of course be provided.

Procedural noises in NPR. Our noise primitive for NPR makes the connection between mark-based methods in NPR and methods for sparse convolution noise in procedural texturing more explicit. This places existing work in a different perspective. For example, the watercolor of Bousseau et al. [BKTS06] is actually a sparse convolution noise [Lew89] with a Gaussian-like kernel, and the dynamic canvas of Kaplan and Cohen [KC05] is actually a spot noise [vW91] with a fiber as the spot shape. This connection is likely to be productive. For example, to further extend the range of patterns, it would be interesting to construct a version of spot noise [vW91] that takes into account temporal coherency, using the ideas of NPR Gabor noise.

8

STYLES

BOTH dynamic solid texture (Chapter 6) and NPR Gabor Noise (Chapter 7) can be directly used with many NPR methods by only substituting standard textures for them. We illustrate this principle on a variety of binary (Sec. 1) and color (Sec. 2) styles. Some of them are inspired by existing methods and others are original style that benefits from the variety of patterns we can produce.

Input solid textures can be produced following two different approaches. We generate some of them procedurally using Perlin noise [Per85, Ola05] shaders. The more natural and complex textures are synthesised from 2D exemplars using the algorithm of Kopf et al. [KFCO*07].

Starting from NPR Gabor Noise, we use standard techniques from procedural texturing and modeling (e.g., [EMP*02]). See the "cookbook" in Appendix B for a detailed description of the style shaders, noise parameters and blending functions we designed.

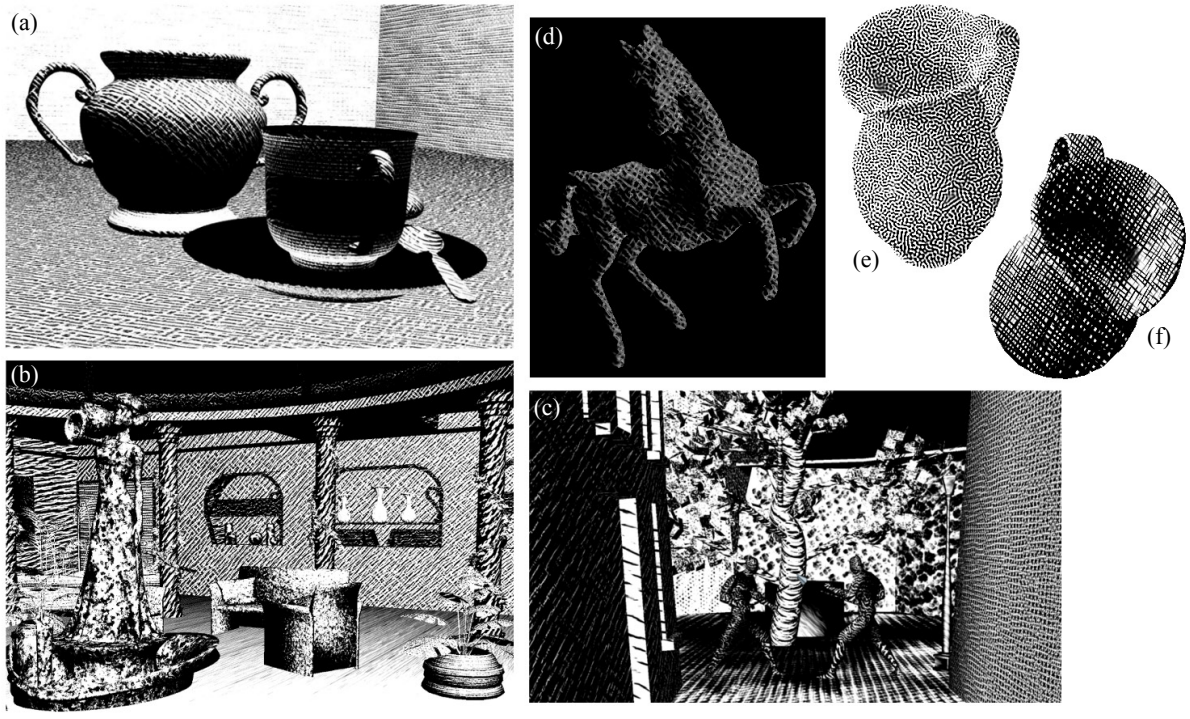


Figure 8.1: Binary styles. (a-c) scenes rendered with dynamic solid texture; (d-f) patterns produced with NPR Gabor noise. See the accompagnying videos of [BBT09, BLV*10a] to appreciate the coherence of these styles during motion.

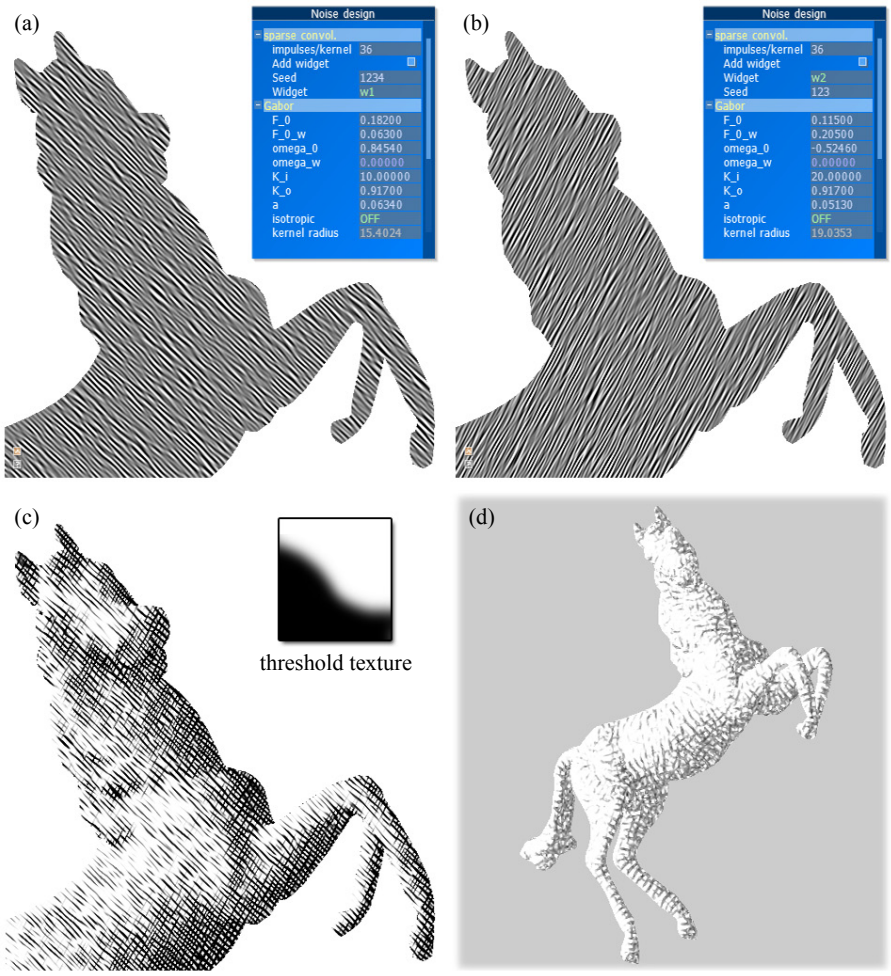


Figure 8.2: Style design. (a, b) Two anisotropic noises are designed. (c) The thresholded results are multiplied to produce a cross-hatching. The inset shows the x-toon texture used for thresholding. (d) Adding a third high-frequency noise layer allows to produce a graphite style.

1. Binary styles

We obtain black-and-white shading styles (pen-and-ink, stippling, charcoal...) using a rendering pipeline very similar to the one of Durand et al. [DOM*01]. In this method, strokes are simulated by truncating a *threshold structure* – a grayscale texture seen as a heightfield – at different height with respect to the target tone. Similarly we threshold dynamic solid textures or NPR Gabor noise layers using sigmoid-shaped functions or x-toon textures [BTM06].

Fig. 8.1a-c illustrate the diversity of binary styles that we can combine in one image by simply using a different dynamic solid texture for each object. During the animation, binary strokes appear and disappear progressively thanks to the infinite zoom mechanism. Note that contrary to existing methods for pen-and-ink stylization [HZ00, PHWF01], our solid texture based approach does not orient the binary strokes along the principal directions of the surface.

Fig. 8.1e,f show the same kind of patterns produced using anisotropic (hatches) and isotropic (stipples) NPR Gabor noises. We use a smooth step function to threshold the noise. The position of the step controls the overall intensity of the binary pattern, and the smoothness of the step controls the smoothness of the edges of the hatches and stipples. The frequency of the noise controls the average distance between stipples or hatches, and the bandwidth of the noise controls the regularity of the pattern.

Figure 8.3: Watercolor stylization. (a,b) scenes rendered with dynamic solid texture, (c) with NPR Gabor noise.

We achieve local control by linking these parameters to scene attributes: examples include noise orientation linked to geometric curvature (Fig. 8.1d) or noise frequency linked to shading, allowing for various hatches or stipple sizes (Fig. 8.1 e,f). We obtain cross-hatching by multiplying two thresholded anisotropic noises, and graphite by modulating hatching with a high-frequency noise layer (Fig. 8.2d).

■ *Local control is achieved with linking NPR Gabor noise parameters to scene attributes (curvature, shading).*

2. Color styles

We produce color styles by compositing the pattern with the scene colors in various ways (e. g., alpha-blending, overlay).

Watercolor. We draw inspiration from the pipeline of Bousseau et al. [BKTS06] to create our animated watercolor style. Their approach makes use of 2D gray level textures to mimic the variation of watercolor pigments density on paper.

We simply replace these textures by dynamic solid textures (Fig. 8.3a-c) or NPR Gabor noise layers (Fig. 8.3d and Fig. 8.6). As a result, each of the watercolor effects related to the texture (pigmentation pattern, wobbling) remains totally coherent during the animation. The accompanying videos of [BBT09, BLV*10a] illustrate the richness of the medium and the temporal coherence obtained with this approach.

Brush effects. Using NPR Gabor Noise, we also use scene color in an overlay blending mode to produce felt-tip patterns (Fig. 8.4a). A painterly style is produced using large hatches overlaid with an high frequency anisotropic noise that simulates brush fibers (Fig. 8.4b-d). We add a bump mapping effect to emphasize the fibers [Her02].

Collage. We finally propose a new stylization process that we call *collage*. This new style takes advantage of the diversity of the texture gallery that one can synthesize. Traditional collage consists in creating an image as a composition of sev-

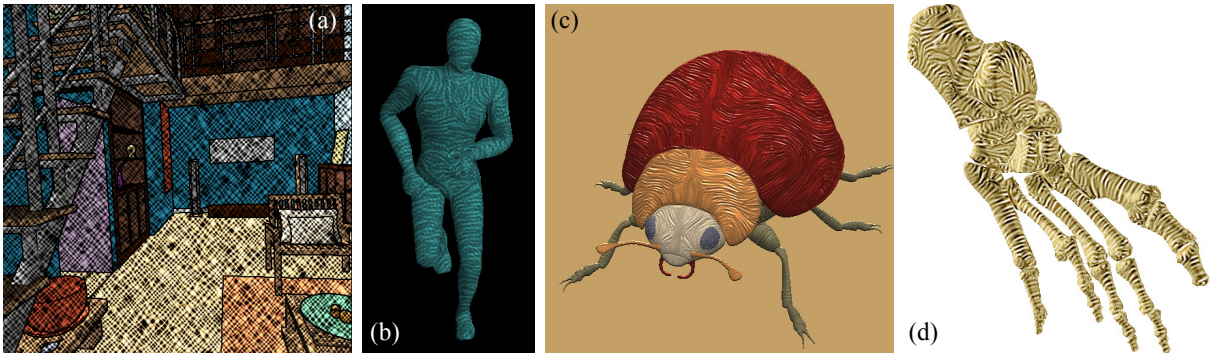


Figure 8.4: Brush effects allowed by NPR Gabor noise: (a) felt-tip pattern, (c-e) brush fibers guided by curvature.

eral small strips of paper with various colors and textures. We mimic this style by assigning different dynamic solid textures to each tone of the image (obtained by a discretized shading model similar to toon shading [LMHB00]). In order to reinforce the paper aspect of the collage, a white border and a wobbling effect is added between the tone strips (Fig. 8.5). The accompanying video of [BBT09] illustrates the accurate temporal coherence of the paper strips, which would be difficult to obtain without our dynamic solid textures.

3. Discussion

Dynamic solid textures and NPR Gabor noise are making a different tradeoff and offer different advantages. The first one targets real-time applications, such as video games, and fits seamlessly into such rendering pipelines. However, despite the infinite zoom mechanism, perspective distortions remain, especially on deep planes tangential to the screen. On the other hand, NPR Gabor noise fully preserves the 2D appearance of the pattern, but it is more computationally demanding and may suffer from temporal discontinuities.

In terms of styles, neither dynamic solid textures nor NPR Gabor noise offer direct control at the stroke level, contrary to few-mark approaches. Consequently, we cannot use them to create styles with clearly independent marks. This limitation is shared with other texture-based and many-mark methods.

However, in contrast to most other texture-based approaches, NPR Gabor noise does offer local control, for example by varying the noise parameters according to scene attributes.

Figure 8.5: Collage style using three dynamic solid textures per object.

Our current method for style design is not entirely trivial. Nevertheless, we were able to create a wide variety of styles, with reasonable effort, and without an advanced user interface (see Fig. 8.2 and the video of [BLV*10a] for a style-creation sequence). Of course, more intuitive methods for artistic control would significantly facilitate this process.

The manual creation of solid textures by artists is not simple. However, recent solid texture synthesis by exemplar algorithms [JDR04, KFCO*07, QY07] allow to generate automatically a wide range of patterns. Moreover the object space infinite zoom mechanism of dynamic solid texture also works for 2D textures as long as texture coordinates are available.

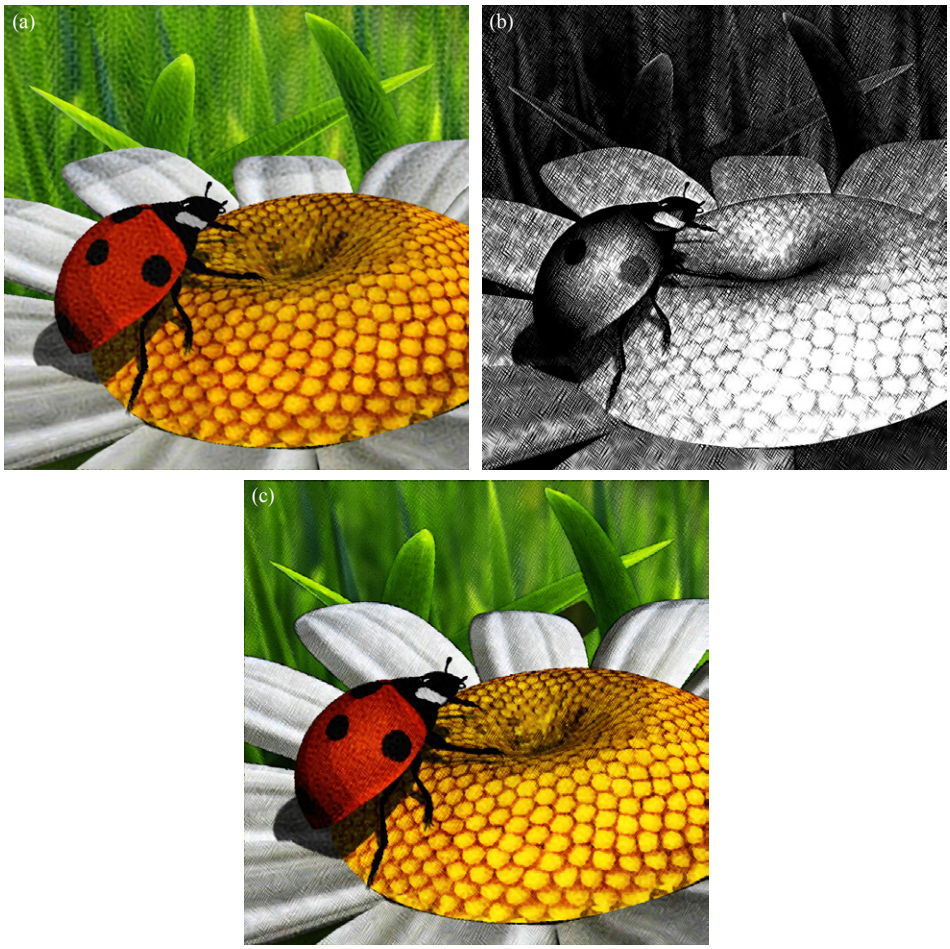
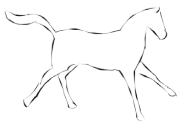
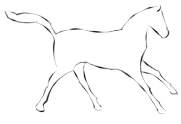


Figure 8.6: Complex scene combining (a) watercolor and (b) cross-hatching styles using NPR Gabor noise.



Part III

Evaluation



Summary

Evaluation is a longstanding question in non-photorealistic rendering. It is particularly difficult for animations, as no absolute ground truth is generally available. Starting from purely speculative visual inspection, progresses have been made in computer graphics in general and NPR in particular to develop more objective and perceptually grounded evaluation methodologies.

Focusing on the temporal coherence problem when stylizing shaded regions, we propose in this part two user-studies contributing to this trend. We first investigate the effect of fractalization on various medium textures and derive an objective quality metric from users judgments.

In a second study, we compare six stylization methods – including our Dynamic Solid Textures and NPR Gabor Noise – with regards to the three goals defining the temporal coherence problem. The results of this study tends to show that motion coherence is a key goal to preserve in the overall temporal coherence compromise.

9

PERCEPTUAL EVALUATION IN GRAPHICS

ADVANCES in Computer Graphics are usually demonstrated through images or videos. Visual inspection is consequently paramount in the evaluation of these techniques. In image processing or photorealistic rendering, reference solutions are often available and can serve as basis for this evaluation. For example, an approximate real-time rendering method can be compared to a more precise off-line approach to evaluate how similar their outputs are perceived. Various approaches have been proposed to assess the fidelity of an image or a video sequence to a reference (Sec. 1).

However, most of the time, no such ground truth exists for non-photorealistic techniques, in particular when dealing with animations. Nevertheless objective evaluation methodologies have been developed to go beyond purely speculative visual judgment (Sec. 2).

Among them, psychophysical methods (Sec. 3) are well-grounded and have been used in many other fields. As stated by Ferwerda [Fer08], psychophysics is *"concerned with developing experimental methods for objectively measuring our subjective perceptual experiences"*. The tools developed in psychophysics seem perfectly suited for our needs and we will use some of them for our user-studies (Chapter 10 and 11).

1. Image and Video Quality Assessment

Image and Video Processing. The image and video processing community has used for a long time mean squared error (MSE) and peak signal-to-noise ratio (PSNR) as fidelity metrics. They are simple and fast to compute, but only rely on byte-by-byte comparison of the data without relationship with the quality perceived by human observers. The only alternative was to perform formal user-studies in order to compute a *mean opinion score* from many responses. This approach is costly and time-consuming, thus not practical on a regular time basis.

Consequently much effort has been spent on designing better predictors that take into account the effects of distortions and content on perceived quality. Their concordance with users judgments are empirically verified. Two kinds of methods have been proposed: a *vision modeling* approach and an *engineering* approach [Win05].

The first one tries to model with filters certain parts of the human visual system (HSV) – e. g., color perception, contrast sensitivity – using data from psychophysical experiments. Well-known metrics in this category are the Visual Difference Predictor (VDP) by Daly [Dal92] and the Sarnoff's just noticeable difference (JND) metric described by Lubin and Fibush [JF97]. Frame-by-frame application

of these metrics on videos is insufficient. Temporal models of visual perception [vdBV96, WM02, Win05] have been developed to overcome this limitation.

On the other hand, the engineering approach is based on the extraction of certain feature of the data, either structural element like contours, or specific distortions (e. g., blur, ringing, masking, color bleeding). For example, the Structural Similarity index (SSIM) [WBS*04] computes the mean, variance and covariance of small patches inside an image and combines the measurements into a distortion map. The Visual Information Fidelity (VIF) [SB06] proposes to conciliate vision modeling and engineering approaches in a novel information framework based on natural scene statistics. Extensions to videos using optical flow to evaluate motion have been recently proposed [SB10]. They allow to estimate temporal artifacts (e. g., jerkiness, blockiness, flickering).

Applications in Computer Graphics. Human perception takes a growing place in computer graphics. I refer the reader to the state-of-the-art report of O'Sullivan et al. [OHM*04] for an in-depth overview of existing techniques.

Image quality assessment has been studied in the area of high quality rendering for two main purposes: to improve the speed of global illumination systems by establishing stopping criteria, and to optimize the allocation of resources using perceptual error maps. Previously described metrics, such as the VDP [DPF03, FP04], as well as specially designed ones (e. g., [RFWB07]) have been used for static images. To handle temporal artifacts, such as flickering, extensions to dynamic scenes have been developed using the image flow [MTAS01], attention models [YPG01] or spatiotemporal models of the HSV [AvMS10].

In another sub-field, user studies have compared the performances of tone mapping operators in terms of overall quality [LCTS05] and also local image attributes [vWNA08] (brightness, contrast, reproduction of colors and details). Automatic quality metrics to compare high dynamic range images with their tone-mapped counterparts have been proposed and validated by experimentation [AMMS08].

2. Evaluation in Non-Photorealistic Rendering

Perceptual evaluation has been used in NPR for specific styles or applications [SD04]. The goal of this evaluation is either to quantify to which extend stylization modifies (and hopefully improves) the perception of the depicted scene or the information it conveys, or to assert the visual quality of the stylization process itself. Designing meaningful evaluation with unbiased questions is hard in general and especially difficult in the case of non-photorealistic rendering for two main reasons: (1) we often want to evaluate complex images that involve high level of cognition; (2) purely aesthetic preferences can easily influence user assessment.

Schumann et al. [SSLR96] compare pencil-like drawings with standard output of CAD systems by polling architects. Similarly, Agrawala and Stolte [AS01] assess the effectiveness of their route maps using feedback from a large number of users. Though informative, these approaches do not provide an objective measure of the methods evaluated.

Isenberg et al. [INC*06] conduct an observational study to evaluate how people perceive both hand-made stippling illustrations and computer-generated ones. Going a step further, they perform a statistical analysis [MIA*08] using gray-level

co-occurrence matrices (GLCM) to compare the correlation between the different techniques. They finally demonstrate [KMI*09] that GLCM can be used to synthesize new distributions of stipples.

An orthogonal approach to evaluate non-photorealistic rendering consists in measuring users' performance when doing a specific task. Gooch and Willemsen [GW02] measure the perception of space for users immersed in a line drawing environment seen through a head-mounted display. Gooch et al. [GRG04] and Wallraven et al. [WBC*07] investigate the effect of stylization on learning and recognition tasks of faces and facial expressions. The main limitation of these approaches is that the goal of the stylization is not always task-related (e. g., aesthetic, abstraction).

Instead, the act of looking is very often, in itself, the central task. Santella and DeCarlo [SD04] use eye-tracking to measure the impact of stylized images on viewers' perception. This passive setup analyzes eye movements without impacting users' behavior to provide an objective measure of perception.

3. Psychophysical Methods

Our goal is to compare images or animations produced either with a given medium by different stylization algorithms, or with various media by the same method. To choose the best suited experimental setup to achieve this goal, we review here the main methodologies proposed in psychophysics.

Two main psychophysical quantities can be measured: *thresholds* and *scales* [Fer08]. The former tests the limits of perception: the absolute detection of a stimuli, or the discrimination between two stimuli. The later tries to quantify how the appearance of a stimuli changes when its parameters change. Our work falls in this category. Three main methodologies exist to measure psychophysical scales: *rating*, *paired comparison* and *ranking*. We briefly present their advantages and drawbacks as well as previous applications in graphics.

Rating consists in assigning a numerical score to each stimuli based on examples of endpoints of scale. This is the usual setup in image and video quality assessment [SSB06, Win05]. It is often used to estimate a statistical metric (root mean square, Pearson correlation and outlier ratio) between the output of an objective quality metric and the results from viewer subjective rating of the test images. However, the validity and reliability of rating data can only be asserted considering very large number of trials, and participants have to be trained before the trial [Ken75].

Paired Comparison. In this setup, the whole dataset is presented pair by pair to the user who has to make straightforward forced choices. This approach has been used to compare the performances of tone mapping operators [LCTS05, vWNA08]. Its main limitation is its quadratic complexity. In our studies it would have meant to compare hundreds of pairs, making the experiment tiresome for users.

Ranking uses images organized according to a certain varying degree of quality [Gui54]. Hence, this setup is the least time consuming one while it still provides for relevant data. Although ranking is more complex than paired comparisons, it is more appreciated by participants in general. Stokes et al. [SFWG04]

run a series of ranking experiments to define the perceptual importance of different illumination components for efficient high quality global illumination rendering.

We opted for a ranking setup in our user studies, since ranking tasks are generally considered more stimulating and enjoyable than alternatives such as pairwise comparisons. Moreover a ranking task can be regarded as multiple simultaneous pairwise comparisons at the observer's own pace. It is therefore less time consuming than the alternatives while producing comparable results.

10

QUALITY ASSESSMENT OF FRACTALIZED TEXTURES

As observed in Sec. 5, *texture fractalization* modifies the patterns in the texture: new features and new frequencies may appear, global contrast may be degraded and deformations may occur. As a result the fractalized texture is likely to be visually dissimilar to the original pattern targeted by the artist. We believe that the automatic evaluation of this similarity loss can be a valuable tool for comparing existing alpha-blending approaches and may allow the development of new fractalization techniques.

In this context, we define the texture *distortion* as the dissimilarity between the original and transformed 2D texture. Our goal is to define a *quantitative metric* for this distortion. For that, we perform a study in which users are asked to rank pairs of original/distorted textures from the least distorted pair to the most distorted pair. We provide a statistical analysis of the results to derive perceived quality scales for ten classes of NPR media. In this study, we deliberately avoid giving an explanation of the nature of the transformation. We ask the participants themselves to identify the criteria they have used to assess the distortion.

In a second step, we study the correlation of these subjective results with several objective metrics that are coming from image quality assessment and texture analysis, based on global and local image statistics or spectrum analysis. We suggest that the *average co-occurrence error* (ACE) is a good predictor of the distortion.

1. Experimental Framework

Stimuli We choose twenty gray-scale 2D textures sufficiently representative of the main traditional media used in NPR. To create a redundancy in the results,

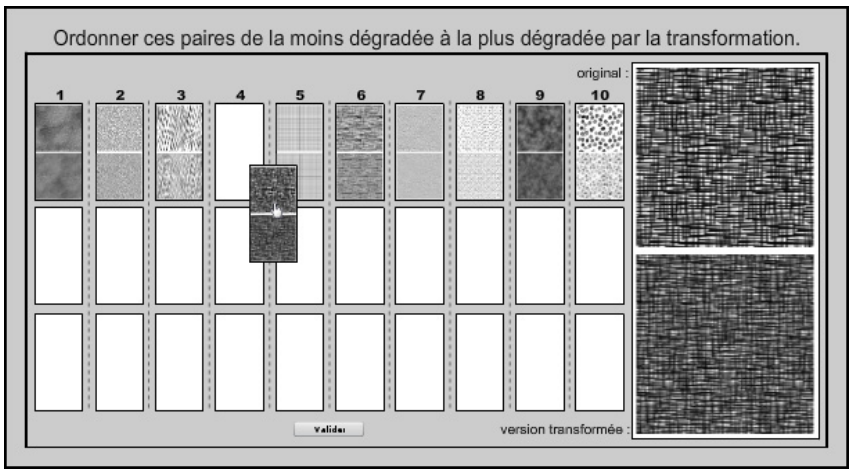


Figure 10.1: Web interface for ranking the two series of ten texture pairs. Up to three pairs can be ranked equally (vertical columns). A close-up view is provided to zoom on the currenlty flyed over texture (right).

we design two sets of ten texture pairs (S_1 and S_2). For each set, we choose one representative texture per class (pigments on canvas, paint, paper, hatching, cross-hatching, dots, near-regular or irregular patterns, noise and grid).

To construct S_1 and S_2, each of the original textures is transformed using a 2D static version of our fractalization algorithm (Chapter 6). Here, three scales of the texture are alpha-blended (with the coefficients $1/3, 1/2$ and $1/6$). This number is a minimum to ensure temporal continuity in most dynamic scenes. Fig. 10.2 shows the twenty pairs – original and transformed version – obtained for the two series.

Procedure We realize the ranking experiment via a dynamic web interface[1] (Fig. 10.1) to enrich and diversify our panel of subjects. Nevertheless, in order to keep a certain amount of control on the participants' origin, we have restricted the diffusion of this web site to known people. We are aware that our interface prevents us from having a precise control on the experimental conditions (screen resolution, calibration, viewing angle and ambient lightning). Consequently, we paid special attention to assessing the statistical validity of the resulting data as detailed in Sec. 2. We consider this trade-off admissible with regard to the number of participants (103) and the diversity of their skills in computer graphics (naive: 58%, amateur or professional digital artists: 8.5%, researchers: 22.4% and unknown: 11.1%).

On each trial, after presenting the general context of this study, the subject is asked to place the pairs in order from the lowest to highest by perceived distortion. This ordering can be partial if texture pairs seem identically distorted. To ease the comparison, a close-up view of the pair flown over by the mouse allows the user to compare the original and transformed textures precisely without aliasing.

After the ranking of the second set, the subject is asked to specify for each of these last ten pairs the criteria he thought to have used. We provide three basic criteria: *contrast*, *sharpness* and *scale*; but we also allow the user to freely type personal criteria. The two image sets are shown randomly at each user trial to avoid learning effects. The duration of each task (first ranking, second ranking and criteria specification) are recorded.

2. Statistical Analysis

103 people took part in this study. Among them, 45 started with the first image set (S_1^1 then S_2^2) and 58 with the second (S_2^1 then S_1^2).The statistical analysis of the resulting two datasets are carried out identically in three steps. After studying the potential learning effect, we propose to derive an interval scale of relative perceived distortion intensity. Finally we run an analysis of the criteria used during the ranking.

2.1. Ranking Duration and Concordance

First, we estimate whether the presentation order of the two image sets has an influence on the final ranking to verify that the subjects are not biased by the first ranking while doing the second.

[1] http://artis.inrialpes.fr/~Pierre.Benard/TextureQualityMetric/UserStudy

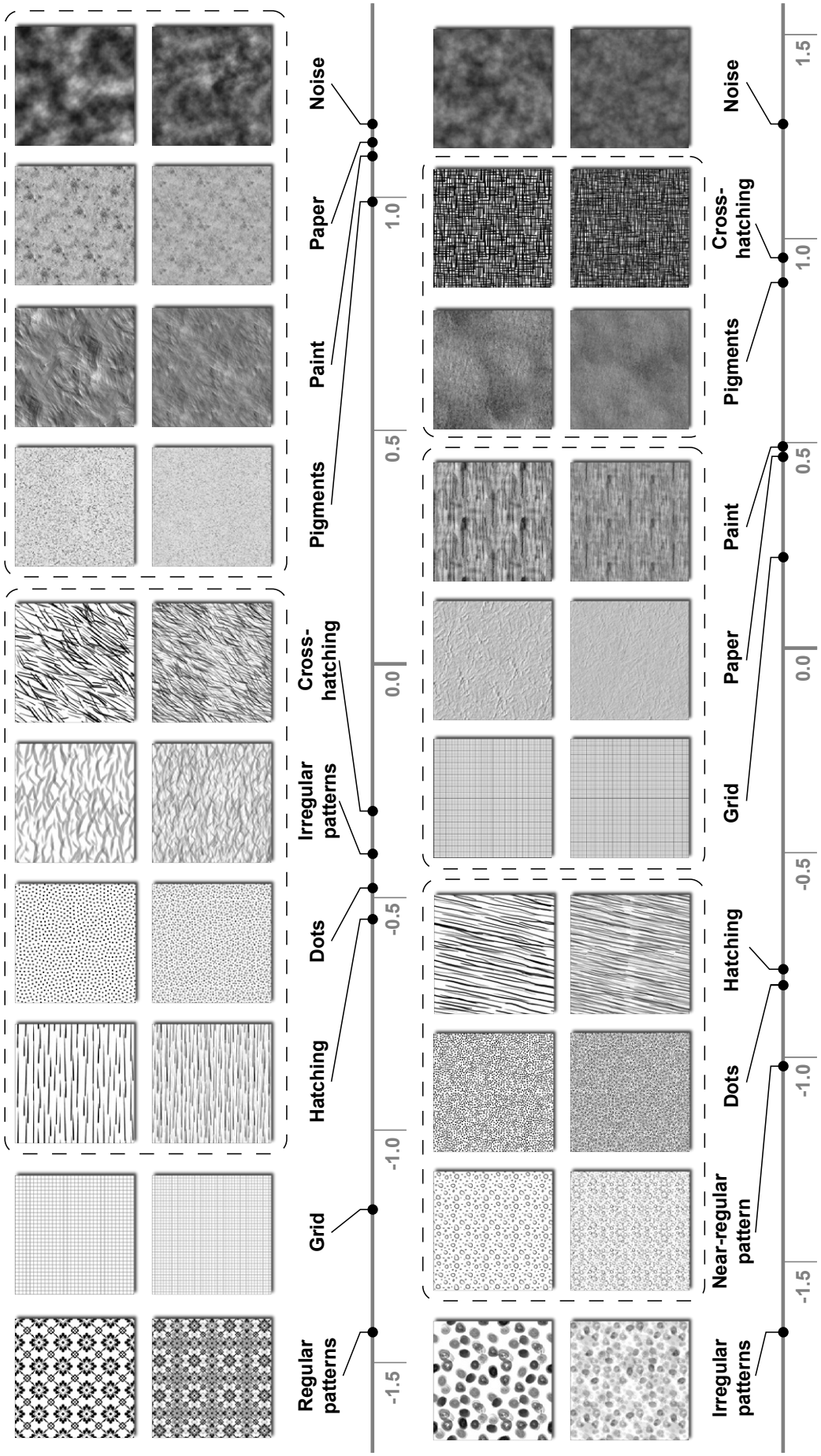


Figure 10.2: The two series of medium texture pairs in decreasing order of perceived distortion. First (top) and second (bottom) texture pairs sets. For each series, the first row corresponds to the undistorted original textures, while the second depicts their fractalized versions. The intervals scales are derived from the z-Scores shown in Tab. 10.2. Texture pairs surrounded by the same dashed line may be considered as perceptually equivalently distorted.

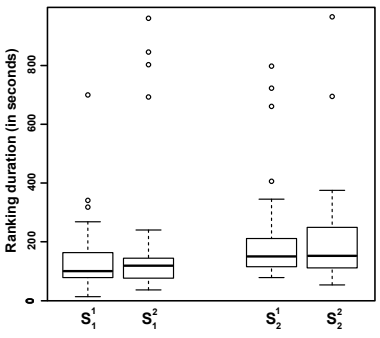


Figure 10.3: Boxplot of the ranking duration for the two series.

Fig. 10.3 shows the boxplot of the experiments duration for the two series depending on presentation order. Notice that for each series the distribution of durations is similar whatever the presentation order. Thus, we can conclude that the presentation order has no significant influence on the subjects' attention. Moreover the mean duration of both series is comparable (211s and 147s respectively).

In order to quantify the consistency of rankings provided by the users panel, we compute the Kendall's coefficient of concordance (Kendall's W) [Ken75] over the four results sets S_i^j and the S_i obtained by merging the previous series. This non-parametric measurement is commonly used to evaluate the agreement among raters (from 0 for no-agreement to 1 for full-agreement) without the need to assume any specific distribution. It is computed as:

$$W = \frac{s}{\frac{1}{12}k^2(n^3-n)} \quad \text{with} \quad s = \sum_j \left(R_i - \sum \frac{R_i}{n}\right)^2 \tag{10.1}$$

where $i \in \{1 \ldots n\}$ are the pairs to rank, j the judges and R_i the sum of the ranks assigned to i among the different rankings.

	S_i^1	S_i^2	S_i
$i = 1$	0.598	0.55	0.574
$i = 2$	0.576	0.657	0.599

Table 10.1: Kendall's coefficiants of concordance

As shown in Tab. 10.1, the coefficients of concordance of the merged results sets S_i remain comparable with the un-merged ones. Regarding these statistics the analysis of merged data seems relevant. To validate the statistical significance of Kendall's W, we perform a χ^2 test with the null hypothesis H_0 that there is no agreement among the subjects. In the six cases, this hypothesis may be rejected at $\alpha = 0.001$ level for 9 degrees of freedom. Hence, we can conclude that there is some agreement amongst the participants, that is, the rankings are not effectively random.

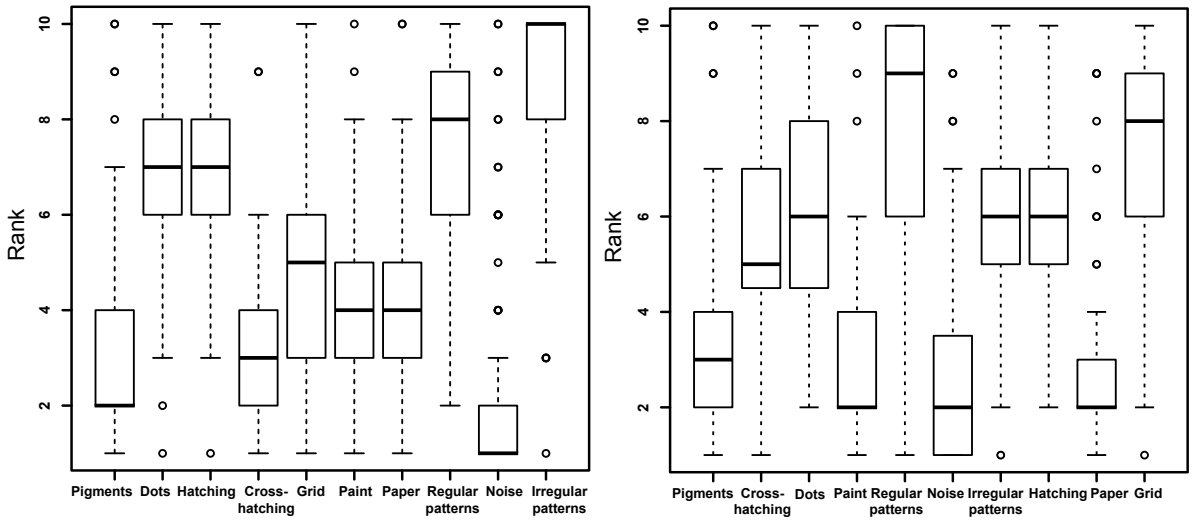


Figure 10.4: Boxplots of the ranking for the first (left) and second (right) texture pairs sets.

Fig. 10.4 shows the boxplots of the two ranking sets, i. e., the smallest observation, lower quartile, median, upper quartile, and largest observation. We can notice that the concordance among the raters varies largely according to the textures pair, which indiquates the difficulty of the task for certain patterns.

2.2. Interval Scale of Relative Perceived Distortion

An ordinal scale can be directly derived by tabulating the raw data to show how often each pair was placed in each rank position (frequency) and calculating their mean ranks. However, it gives no quantification of perceived differences between texture pairs: it doesn't inform on *how much* higher one pair is distorted compared to another.

By assuming a normal distribution of the data and using the Thurstone's *law of comparative judgment* [Tor58], we can convert the proportion of each pair:

$$p_i = \frac{\#\text{ranks} - \text{mean}(\text{rank}_i)}{\#\text{ranks} - 1} \tag{10.2}$$

into z-Score (Tab. 10.2):

$$z(p_i) = \frac{p_i - \mu}{\sigma} \tag{10.3}$$

with μ the mean and σ the standard deviation of these proportions.

These z-Scores correspond to relative differences in perceived distortion between texture pairs on a perceptually-linear scale. We represent them on a 1D plot of relative perceived distortion intensity as illustrated in Fig. 10.2 below the pairs of textures. For both image sets, note that the unstructured textures (noise, pigment, paper) seem to be more robust to the fractalization process. On the contrary, textures exhibiting more distinctive features are in both cases judged as the most severely distorted.

The normal distribution assumption used to applied the Thurstone's law of comparative judgment needs to be verified. We use a normal probability plot (Q-Q plot against a normal distribution) of the proportions (Fig. 10.5). The further the points vary from a straight line, the greater the indication of departures from normality; thus the normal distribution hypothesis is less confirmed for the first series than for the second.

We validate these results with a a Shapiro-Wilk Normality test. The null hypothesis of this test H_0 is that the population is normally distributed. If the p-value is less than the chosen alpha level, then the null hypothesis is rejected (i.e. one concludes that the data are not from a normally distributed population). If the p-value is greater than the chosen alpha level, then one does not reject the null hypothesis that the data came from a normally distributed population. The p-value of this test for S1 and S2 are respectively 0.06998 and 0.37. This confirms our previous observation: the null hypothesis can be rejected for S1 assuming a quite low alpha level whereas for S2 it cannot be reasonably rejected.

	Pigments	Dots	Hatching	Cross-hatch.	Grid	Paint	Paper	Reg. patt.	Noise	Irr. patt.
S_1	0.9936	-0.4810	-0.5487	-0.3167	-1.1725	1.0903	1.1193	-1.4336	1.1580	-0.4085
S_2	0.8921	-0.82544	-0.7848	0.9552	0.2204	0.4909	0.4683	-1.0238	1.2798	-1.6729

Table 10.2: z-Scores of the two series.

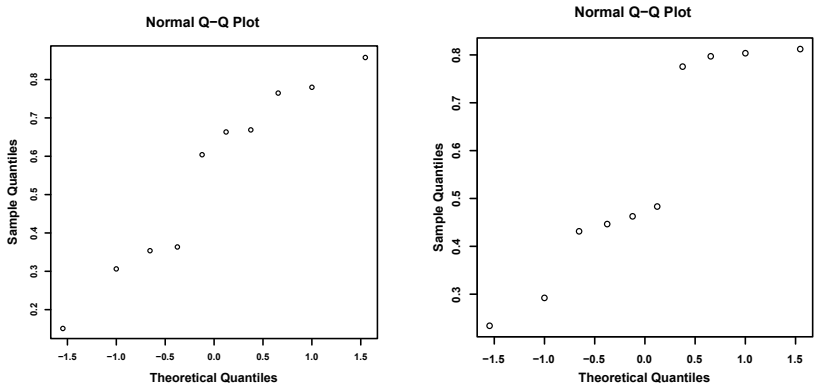


Figure 10.5: Normal probability plots of the proportions of the first (left) and second (right) texture pairs sets.

To go a step further, we perform statistical hypothesis tests to ensure that the perceived intensity is significant. We use the non-parametric *Wilcoxon rank sum test* (also called *Mann-Whitney U test*) for assessing whether two independent samples of observations come from the same distribution. The *U test* has both the advantages of working on ordinal data and of being more robust towards outliers than the *Student t-test*. The null hypothesis H_0 is that the two considered samples are drawn from a single population, and therefore that their distribution is equal. In our case, this test has to be run for each pair of samples: $(\text{tex}_{i,m}, \text{tex}_{i,n})$ where $1 \leq m \leq n \leq 10$ (i.e. ninety times).

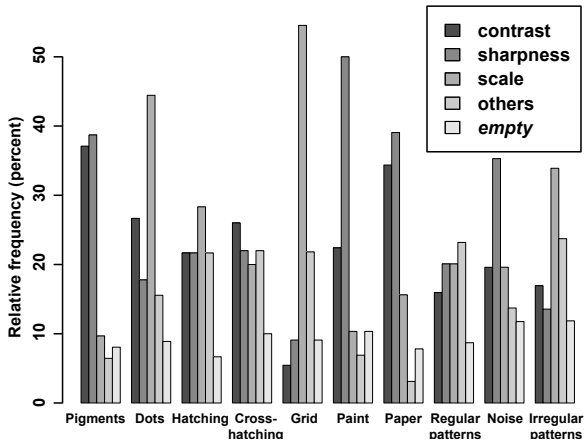
In Fig. 10.2, we frame groups of pairs for which the null hypothesis cannot be rejected. By looking at the corresponding texture pairs, we observe that the overall contrast of patterns seems to be the most significant criterion. In comparison, feature shapes seem to play a less important role in this grouping. Note that the interval scales and groups of each image set are not fully identical. It indicates that the classes of media we defined *a priori* only partially fit the classification based on the distortion. In particular, the two representative textures of the "grid" and "cross-hatching" classes are judged distorted differently. We think that this is due to their strong difference in terms of contrast, pattern density or feature shapes.

2.3. Ranking Criteria

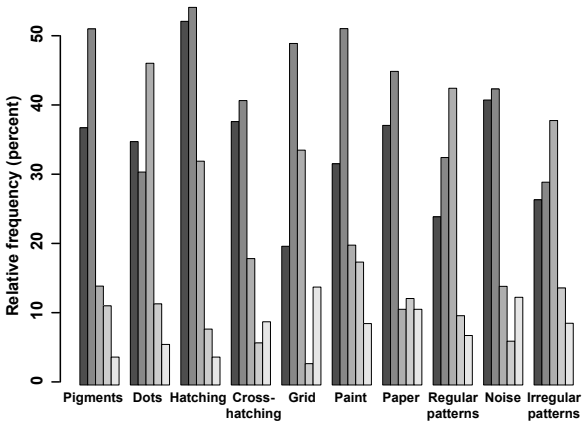
We finally analyze the criteria that each subject considered he used during the ranking. Fig. 10.6c shows the relative frequency at which the three proposed criteria have been used per series. Overall, they are quite similar, with a little preference for *sharpness* and then *contrast* in S_2.

However when we consider this distribution of criteria for each texture pair (Fig. 10.6a and Fig. 10.6b), we observe irregular preferences for different criteria. We thus conjecture that the content of each texture class triggers different criteria when assessing the similarity between original and transformed textures.

The analysis of the additional criteria proposed by the participants also shows that the notions of *density* (15% of these criteria), *shape* (10%) and *pattern coherence* (21%) and to a lesser extent *frequency* (4%) and *relief* (2%) are relevant as well.



(a) First texture pairs set.



(b) Second texture pairs set.

	contrast	sharpness	scale	other	*empty*
S_1	21.96%	26.86%	24.83%	17.40%	8.95%
S_2	28.27%	35.21%	21.72%	7.12%	7.07%

(c) Frequencies for each series.

Figure 10.6: Relative frequency at which the criteria have been used. For each texture pair (a,b) and for each series (c).

3. Correlation with Objective Metrics

In a second step, we review a large range of image metrics and statistics, with the hope of correlating them with the previously derived subjective scales. We use the *Pearson product-moment correlation coefficient r* and linear regression to evaluate this correlation.

3.1. Image Quality Metrics

We first examine eleven well-known objective quality assessment metrics:

- peak signal-to-noise ratio (PSNR),
- signal-to-noise ratio (SNR),
- structural similarity (SSIM) index,
- multi-scale SSIM index (MSSIM)
- visual signal-to-noise ratio (VSNR),
- visual information fidelity (VIF),
- pixel-based VIF (VIFP)
- information fidelity criterion (IFC),
- universal quality index (UQI),
- noise quality measure (NQM)
- weighted SNR (WSNR).

A free implementation is provided by Matthew Gaubatz in the MeTriX MuX[2] MATLAB© package.

None of these metrics shows a significant correlation with the subjective interval scales (Tab. 10.3). This conclusion was predictable as these metrics have been designed to assess the quality of images suffering limited amount of distortion (noise, blur, compressions artifacts, etc). In comparison, the fractalization process may strongly modify the appearance of the distorted texture.

[2] Available at: http://foulard.ece.cornell.edu/gaubatz/metrix_mux

	PSNR	SNR	SSIM	MSSIM	VSNR	VIF	VIFP	IFC	UIQ	NQM	WSNR
S_1	0.2817	0.0257	0.2877	0.1311	0.1788	0.2658	0.0012	0.6116	0.2408	0.1253	0.0272
S_2	0.0417	0.1366	0.0507	0.0185	0.2464	0.3826	0.2612	0.3261	0.2632	0.1962	0.1145

Table 10.3: Pearson product-moment correlation coefficients for eleven quality metrics.

3.2. Global and Local Image Statistics

Because the three criteria – *contrast*, *sharpness* and *scale* – have to be considered simultaneously, we cannot expect global image statistics to give significant results, especially when one considers that our textures are not "natural images". Our experiments on histograms, power spectra and distribution of contrast measurements [BG93] are, as expected, inconclusive.

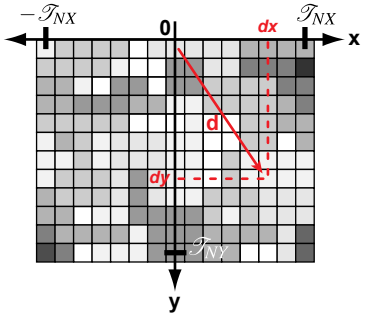


Figure 10.7: Computation of a gray level co-ocurrence matrix.

Taking inspiration from the field of texture analysis [TJ93], we choose a statistical method – the *gray level co-occurrence* (GLC) model – which estimates local image properties related to second-order statistics (like variance, standard deviation and correlation). Moreover, psychophysical experiments have shown that the GLC model matches certain levels of human perception [JGSF73] and it has been successfully used and perceptually validated for texture synthesis [CRT01]. Finally, this model might be related to the *density* and *pattern coherence* criteria freely proposed by the subjects.

This model (Fig. 10.7) consists in a set of n matrices P_d of size $G \times G$, where n is the number of displacement vectors, G is the gray-level quantization step and each entry (i,j) of P_d is the number of occurrences of the pair of gray levels i and j which are a distance d apart:

$$P_d(i,j) = |\{((r,s),(t,v) : I(r,s) = i, I(t,v) = j\}|$$

with $(r,s),(t,v) \in N \times N, (t,v) = (r+dx, s+dy)$ where $dx \in \{-\mathcal{T}_{NX} \ldots \mathcal{T}_{NX}\}$ and $dy \in \{1 \ldots \mathcal{T}_{NY}\}$.

We measure the distortion between original and distorted textures by computing their GLC matrices and then the distance between these two sets of matrices with the *average co-occurrence error* (ACE) of Copeland et al.:

$$\text{ACE} = \frac{1}{\mathcal{T}_{NGLC}} \sum_{d \epsilon \mathcal{D}} \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} |P_d^1(i,j) - P_d^2(i,j)|$$

with $\mathcal{T}_{NGLC} = 2\mathcal{T}_{NX}\mathcal{T}_{NY} + \mathcal{T}_{NX} + \mathcal{T}_{Ny}$.

This error metric is highly correlated with the perceptual interval scales for both series. We obtain the maximum absolute Pearson's correlations of 0.953 for S_1 and 0.836 for S_2 (with, respectively, a *p-value* < 0.0001 and 0.003 for 8 degrees of freedom) considering the GLC matrices for all displacements up to $\mathcal{T}_{NX} = \mathcal{T}_{NY} = 4$ pixels with a $G = 32$ gray levels quantization (with the notation of Copeland et al.).

Fig. 10.8 shows the corresponding linear regression of the ACE against z-Scores for both series ($r^2 = 0.9075$ and 0.6992 respectively, with the same *p-values* as for Pearson's correlations). This high correlation for both image sets confirms that the ranking differences observed in Sec. 2 are coherent and that an *a priori* classification is not a suitable predictor of the distortion.

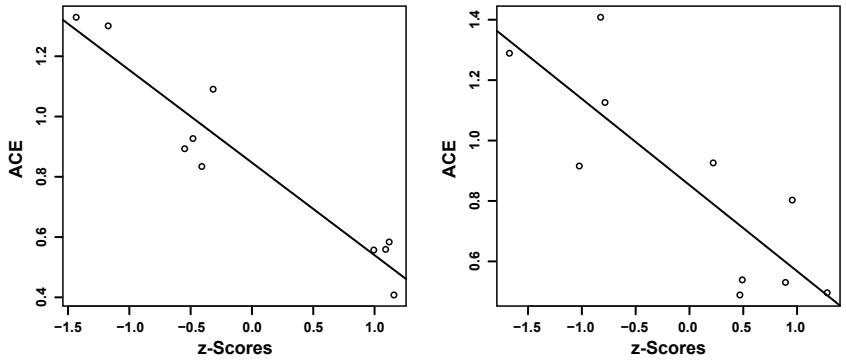


Figure 10.8: Linear regression of ACE against z-Scores for S_1 (left) and S_2 (right).

4. Discussion

We proposed the *average co-occurrence error* as a meaningful quality assessment metric for *fractalized* NPR textures. We validated the relevance of this predictor by showing its strong correlation with the results of a user-based ranking experiment. Nevertheless we would like to investigate potentially better suited texture and vision descriptors to derive an improved objective quality metric. Image retrieval approaches, based on extracted texture features, seem a promising field of inspiration.

Longer term future work will consider the dynamic version of the fractalization process. In this case, the methodology we developed for the current study will also need to handle the trade-off between temporal continuity and texture dissimilarity to the original medium.

11

EVALUATION OF REGION STYLIZATION METHODS

We present a user study which evaluates the success of various methods for stylizing shaded regions, including the two new ones that we proposed.

The goals of flatness, motion coherence and temporal continuity we set out in the introduction are contradictory, and therefore no perfect solution to temporal coherence problem exists. As a consequence, evaluating the success of any given trade-off is complex. However, each goal has specific artifacts which a user can observe. A user study can thus provide significant insight into how well each solution performs for each goal. It also provides an indication of overall success as well as relative importance of the different criteria involved.

The results of our study indicate that coherent motion is perceived as the most important criteria in the overall trade-off. They also shows that both dynamic solid textures and NPR Gabor noise, while making a different compromise, behave well for the goal they are targeting.

1. Procedure

Similarly to our previous user study (Chapter 10), we opt for a ranking experiment which allows to keep the duration of the test reasonable with no loss of information. In the following, we provide a detailed description of the techniques we chose for comparison and the setup we designed.

1.1. Evaluated Techniques

Among the stylization techniques discussed in Sec. 2 of Chapter 2, we need to choose the most representative ones. Among them, the naïve extreme cases of Shower Door (SD) and Texture Mapping (TM) are logical choices for reference comparisons. Then, as summarized in Tab. 11.1, we classify the other approaches into five groups: few-mark, many-mark, object space, local and global image space texture-based approaches.

We do not include few-mark approaches in our comparisons for several reasons: (i) there is no way to produce the same pattern in a texture-based and a few-mark approach; (ii) the severe popping produced by these methods is a highly disturbing, and thus obvious artifact which probably does not require a user test.

Finally, we select the most recent methods in each remaining group:

- NPR Gabor Noise (nprGN) [BLV*10a] and Chapter 7,
- Dynamic Solid Textures (DST) [BBT09] and Chapter 6,

		Flatness	Coherent motion	Temporal continuity	
Naive	Shower door	++	−−	++	
	Texture mapping	−−	++	++	
Few-mark	[Mei96]	−/+	+	++	
	[VBTS07]	++	+	−−	*popping*
Many-mark	[KC05, BKTS06]	++	+	−	*flickering, oscillations*
	[BLV*10a]	++	+	−/+	*secondary motion, popping*
Object space texture	[PFH00]	−	++	−/+	*mipmap*
	[BBT09]	−	++	+	*fractalization*
Local image space texture	[BNTS07]	++	++	−/+	*regeneration*
Global image space texture	[CDH06, BSM*07]	+	−	−/+	*mipmap*
	[CTP*03]	++	−−	+	*fractalization*

Table 11.1: Summary of the trade-offs made by various solutions for the temporal coherence problem.

- Bidirectional texture Advection (Adv) [BNTS07],
- Dynamic 2D Patterns (D2D) [BSM*07],

1.2. Setup

We created a setup with two rows of 6 slots as shown in Fig. 11.1. Initially the top row of slots contains the 6 stimuli (still images or video loops) of the different methods. The accompanying video of [BLV*10a] show them in motion[1]. The users are asked to rank the stimuli from left to right according to a criterion displayed above, by dragging and dropping them to the bottom row.

We created a local setup to ensure smooth 25fps video playback using a modified version of the *MPlayer* software and the multi-threaded *FFmpeg-mt* codec library. The local setup limits the number of participants, but on the other hand it also gives us more control over the stimulus presentation conditions. It consequently decreases the variance of the acquired data to make up for the smaller number of participants.

Figure 11.1: A participant performing the user study.

We used dual 24" LCD monitors at 1920×1200 resolution to provide a large enough work surface for the ranking task. The displays were calibrated to match brightness, contrast, and color reproduction. The experiment was performed in normal office lighting conditions. Participants were unpaid volunteers and had normal or corrected-to-normal vision. They were given written instructions in French or in English (see Appendix C), and were otherwise naive as to the aims of the experiment. Participants were asked to report their overall confidence in their rankings and any difficulties they might have had in a post-study questionnaire.

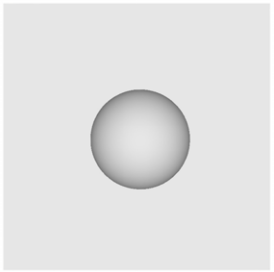
To keep the duration of the sessions below 30 minutes per participant, we decided to split the study into a part involving simple stimuli and a part involving complex stimuli. A total number of 15 volunteers (11 male, 4 female, ages 25–59) participated in both parts of the study. Participants took on average 90 seconds to complete a ranking task, and rarely exceeded 5 minutes even for ranking tasks they reported as difficult.

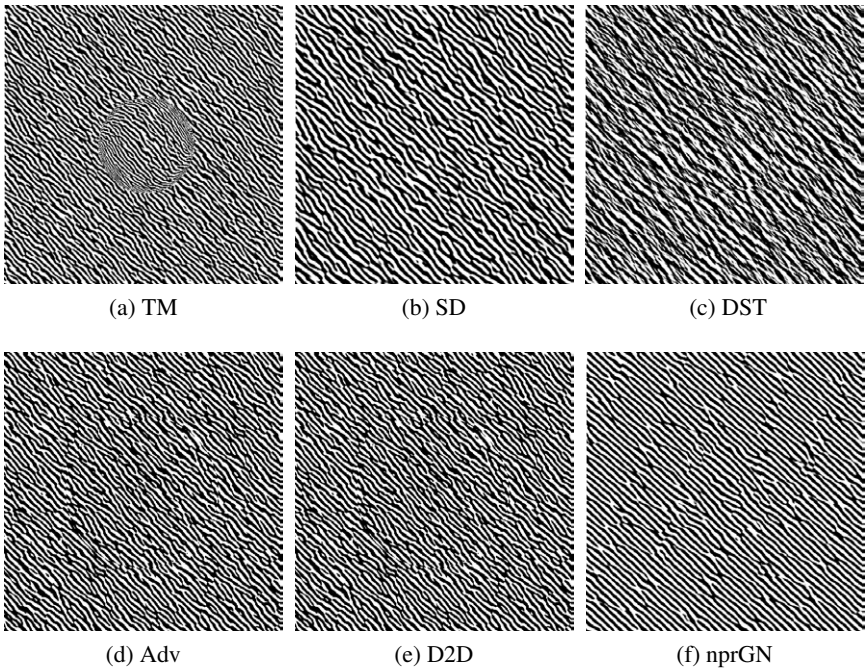
1.3. Analysis Methodology

We first determine appropriate assessment criteria for each goal defined in Chapter 1. We use them as guidelines to choose the stimuli and to formulate the questions we pose to the participants of the study. In the following sections we describe the details and results of the separate ranking tasks, in the order they were presented to the participants.

Similarly to Chapter 10, the results of the ranking tasks were analyzed using Thurstonian scaling [Tor58] to derive interval scales. The statistical significance of observed trends is confirmed by the Wilcoxon rank-sum hypothesis test [Wil45] and is indicated by the p-value. Fig. 11.4 provides the ordinal scales and the similarity groups at the 95% significance level.

[1] Raw stimuli available at: http://artis.inrialpes.fr/Publications/2010/BLVLDT10/stimuli.zip





(a) TM (b) SD (c) DST

(d) Adv (e) D2D (f) nprGN

Figure 11.2: Flatness, simple stimuli: sphere in front of a plane rendered with the same binary pattern.

2. Simple Stimuli

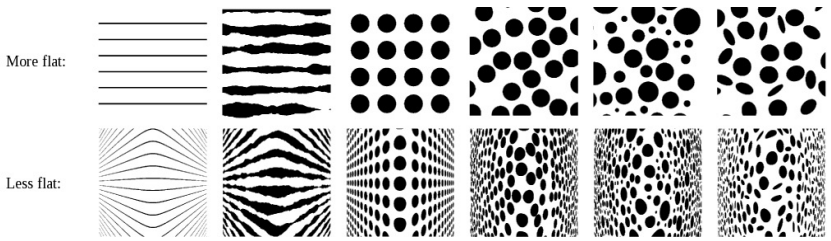
Simple stimuli were created to selectively test the separate goals in controlled conditions. Each stimulus consisted of a single object rendered with a black-and-white hatching pattern (Fig. 11.2 and 11.3), chosen to best reveal the differences and possible artifacts of each selected method.

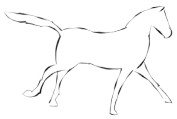
This style was reproduced to the best of our abilities for all the techniques we compared. In particular, a Gabor noise texture with the same parameters was used as input to the previous techniques, and subsequently deformed, advected, or transformed. The color map, in this case a binary threshold, was always applied at the end of the rendering pipeline

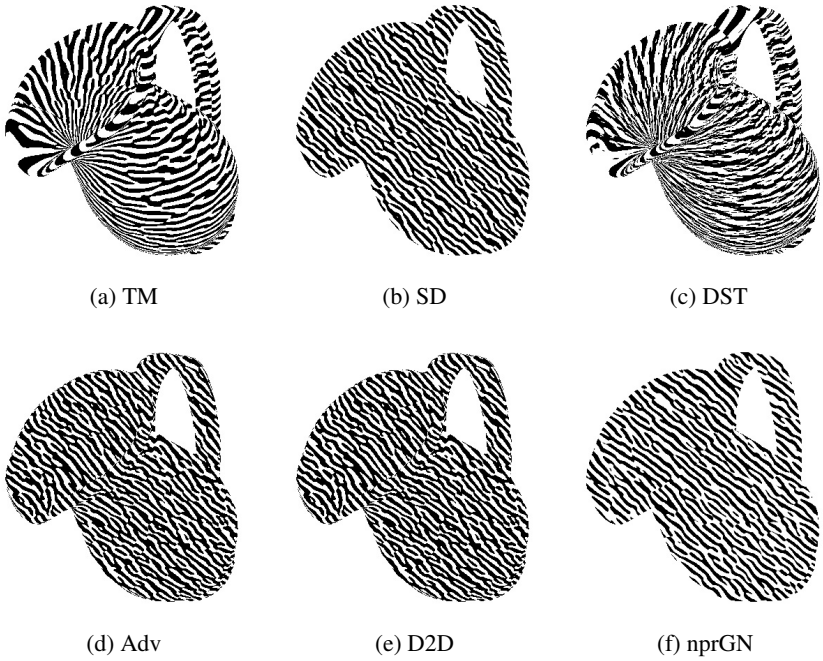
This black-and-white hatching corresponds to an extreme case; we thus believe that our conclusions generalize more readily to other styles. Lighting is disabled to avoid additional shape or motion cues [LMJY95, WFGS07].

2.1. Flatness

The stimuli consist of still images depicting a sphere in front of a planar background, both rendered with the same pattern (Fig. 11.2). Participants are asked to *rank the images according to how flat they appear*. We provide the following images (taken from [TTD*07, Fig. 7]) to clarify this task:







(a) TM (b) SD (c) DST

(d) Adv (e) D2D (f) nprGN

Figure 11.3: Coherent motion and temporal continuity, simple stimuli: pitcher rendered with the same black-and-white hatching pattern.

Since lighting is disabled, the only remaining shape cues are the silhouette of the sphere and the perspective distortion of the pattern [Pal99, Ch. 5]. In practice, the sphere completely blends in with the flat background for methods that do not provide these shape cues in the pattern.

The results show a significant amount of variance, indicating the complexity of this question. People are not accustomed to rely on shape-from-texture cues in isolation. Additionally, several methods produce similar flatness by design: Shower Door and Dynamic 2D Patterns both produce perfectly flat images without perspective distortion, while Dynamic Solid Textures and Texture Mapping both produce the correct perspective. Of the hybrid 2D/3D methods, NPR Gabor noise seems slightly closer to the 2D methods than Advection.

■ *The evaluation of flatness based on texture cues is complex.*

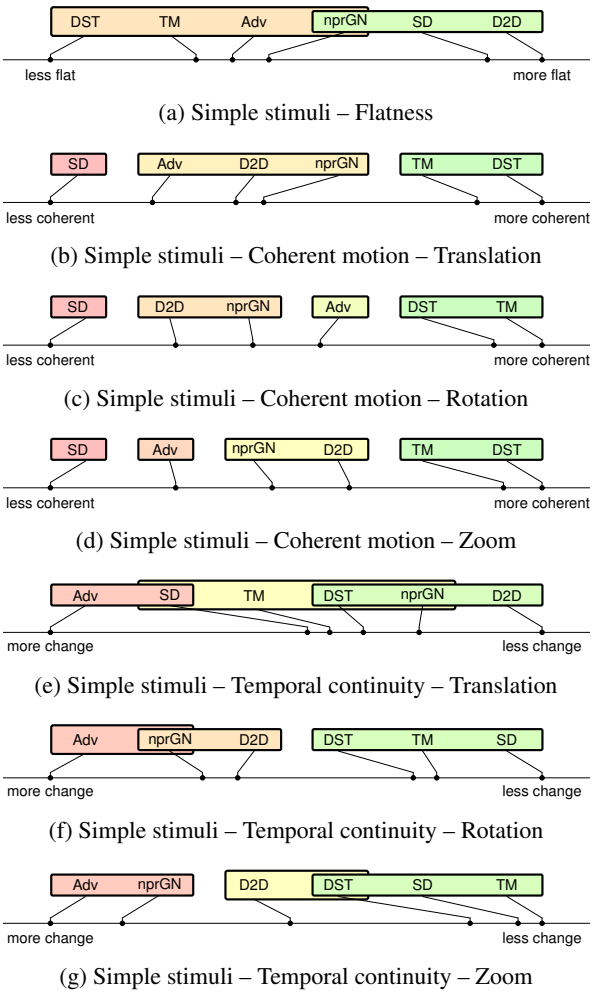
2.2. Coherent Motion

The stimulus videos show a pitcher on a white background (Fig. 11.3). We chose a pitcher as a simple, recognizable, everyday object, to make it easier for the observer to concentrate on the pattern. Nevertheless, the pitcher is sufficiently complex to exhibit some self-occlusion, which is handled differently by the compared methods.

Three separate basic motions were ranked, in random order: translation parallel to the image plane, rotation, and zoom. We need to evaluate the correlation of the motion of the 3D scene and the motion of the pattern after stylization. Insufficient correlation creates the impression that the pattern is sliding over the depicted scene. Participants were asked to *rank the videos according to how coherently the pattern moves with the object.*

As expected, object space methods such as Texture Mapping and Dynamic Solid Textures are significantly more coherent than the other techniques ($p < 0.05$), while Shower Door is consistently the least coherent ($p < 0.05$). The three other approaches which compromise flatness for motion coherence are perceived almost

■ *Object space methods ensure the best coherence of the motion.*



(a) Simple stimuli – Flatness

(b) Simple stimuli – Coherent motion – Translation

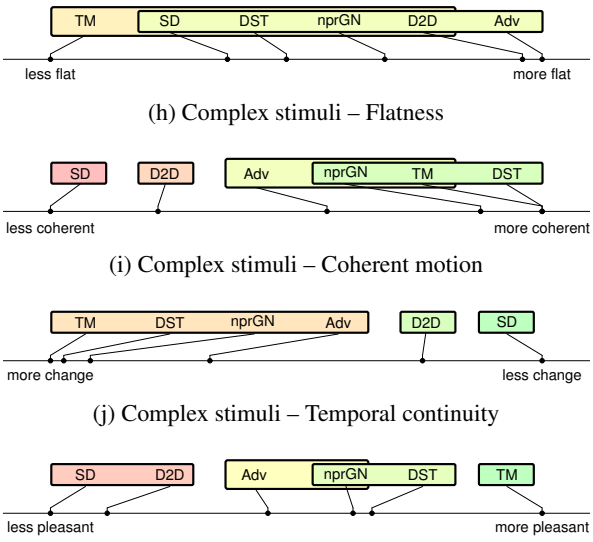
(c) Simple stimuli – Coherent motion – Rotation

(d) Simple stimuli – Coherent motion – Zoom

(e) Simple stimuli – Temporal continuity – Translation

(f) Simple stimuli – Temporal continuity – Rotation

(g) Simple stimuli – Temporal continuity – Zoom



(h) Complex stimuli – Flatness

(i) Complex stimuli – Coherent motion

(j) Complex stimuli – Temporal continuity

(k) Complex stimuli – Pleasantness

Figure 11.4: Results for complex stimuli. Ordinal scales indicating the relative merits of each method. The methods are classified into similarity groups based on the Wilcoxon rank-sum hypothesis test at 95% significance level. Within a group all methods perform similarly. Conversely, methods that are not grouped together are significantly different. A method can belong to multiple groups, as shown by overlapping rectangles.

equally. Depending on the motion, the ranking varies slightly: Advection performs better for rotation ($p < 0.05$), whereas Dynamic 2D Patterns and NPR Gabor Noise perform better for the zoom ($p < 0.05$).

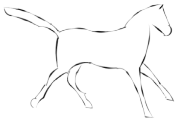
2.3. Temporal Continuity

■ *Temporal continuity in the absence of discrete events (popping, flickering) is hard to assess.*

Given a stylized version of an animation, we want to estimate the fluidity or continuity of the animation. Temporal discontinuities such as popping strongly distract the attention of human observers [YJ84, SS09], and thus are easily identified. Temporally continuous artifacts, which are more subtle and consequently harder to notice, include contrast or intensity oscillations, as well as local residual motion of the patterns. For the same pitcher stimuli, participants were asked to *rank the videos according to how much the pattern changes over time, regardless of the coherence of the motion of the object and the pattern, which they had already evaluated* in the coherent motion task.

The results are not as clear-cut as for the previous question. This may be due to the complexity of the question. For example Shower Door, for which the pattern is constant over time, clearly shows that many participants had difficulty to both ignore the principal object translation and to concentrate on residual motions.

Overall, Advection introduces the largest changes to the pattern in the form of a residual motion called “swimming” in [BNTS07, Sec. 5], which are made more



apparent by subsequently applying the binary threshold. For this kind of pattern, NPR Gabor Noise approach exhibits similar effects during rotations and zoom.

3. Complex Stimuli

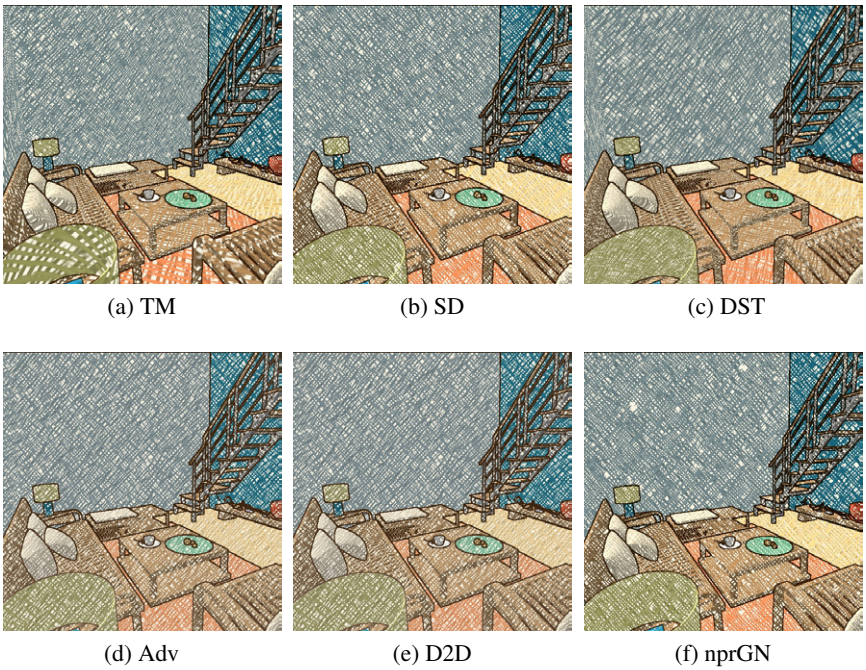
We created complex stimuli to study the behavior of the different approaches for a more aesthetically pleasing experience in a complex walkthrough. The viewer moves inside an indoor scene which contains a mix of large planar surfaces, small details, grazing angles, and depth discontinuities. To make the different objects and surfaces more apparent, lighting is enabled, black outlines are drawn, and the scene is rendered with colored cross hatching patterns on a white paper background (Fig. 11.5).

We kept the same questions as in the first stage. Flatness is evaluated on still images; coherent motion and temporal continuity are evaluated on a complex motion path. To assess the overall aesthetic appreciation of the different approaches, participants were additionally asked to *rank the videos according to how pleasant they find them in the context of cartoon animation.*

Once again, the variance of the answers while estimating the flatness of the scene shows the complexity of this question. This is more pronounced in the complex scene, due to the additional shape and depth cues. Even the extreme 2D case of Shower Door does not rank as completely flat in this test.

The responses concerning motion coherence are consistently in favor of object space approaches. Our noise ranks with the best of the image space methods for this goal. This is probably because other methods exhibit quite pronounced artifacts (e.g., swimming for Advection, static pattern for Shower Door, high amount of sliding for D2D).

The results for motion coherence and pleasantness (Fig. 11.4i and k) are strongly correlated (Kendall rank correlation $\tau = 0.58$ [Ken38]). This indicates



(a) TM (b) SD (c) DST

(d) Adv (e) D2D (f) nprGN

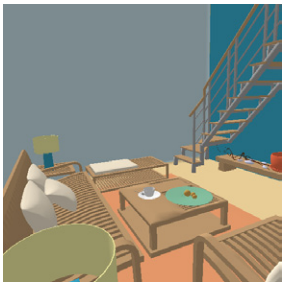


Figure 11.5: Complex stimuli: indoor scene rendered with colored cross hatching, shading and black outline.

■ *Pleasantness and motion coherence are correlated, showing the prime importance of this goal.*

that motion coherence is probably the most important quality to preserve in the overall temporal coherence compromise. Both Dynamic Solid Textures and NPR Gabor Noise perform well on the motion coherence scale; the first one trades off better temporal continuity whereas the second one preserves better flatness.

4. Discussion

Our user study is a first attempt to better understand the trade-offs involved in temporal coherence, and a first step towards a more formal evaluation procedure for this problem. This study is necessarily limited by the choices we have made.

We only considered hatching and cross-hatching patterns. Our previous study (Chapter 10) has shown that fractalization artifacts are clearly perceived for these kinds of structured and contrasted patterns. Consequently we expect that the conclusions drawn for them in this study will more likely extend to other media. It would still be of interest not only to investigate other patterns, but also to measure the effect of the post-processing (e. g., threshold, color map, overlay) on the perceived quality.

■ *The influence of the stimuli (pattern, scene, camera path) on the results needs to be evaluated.*

The 3D test scenes we used as stimuli necessarily have some influence on the final results. Compared to shape-from-texture [Tod04] or shape-from-shading studies [FTA04], we deliberately chose real world objects to make the stimuli look like scenes expected in the applications we target (e. g., films, video games). However it may have introduced some bias in the study. The subjects may have pre-established expectations on the pattern behavior due to the underlying scene. Our complex scene, in particular, exhibits many sharp corners and depth variations which gives a strong 3D appearance to the rendered images. The path of the camera during the walkthrough reinforces this 3D impression. These two effects may explain why Texture Mapping performs especially well on this scene.

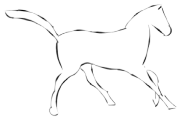
We chose naive users as subjects for this study, because we were targeting the audience of animated films and video games players. However, we would also like to experiment with expert users, such as professional digital artists or technical directors. Thanks to their technical knowledge and experience in computer graphics, they could have a keenest perception of the artifacts.

■ *The goals and their relative importance can be questioned further in the light of the results.*

Finally, the fundamental choice of the goals we evaluated as well as their relative importance need to be questioned. The results of our study tends to indicate that motion coherence is a key goal to ensure in the overall compromise. Besides the previously discussed precautions, it should be note that this conclusion is reached in the absence of discrete temporal discontinuities (e. g., popping, flickering) which already proved to be highly distracting. Consequently flatness is probably the goal on which stylization techniques can compromise the most with the least impact on the perceived quality.

We would like to do additional user studies, taking into account what we have learned so far. But we also want to explore objective measures to quantify these trade-offs, for example, statistical texture measures to quantify flatness, in the spirit of the average co-occurrence error (Chapter 10), and optical flow to quantify motion coherence and temporal continuity. This is likely to lead to deeper insight in the evaluation of solutions for the temporal coherence problem.

Conclusion



12

CONCLUSIONS

THIS MANUSCRIPT defends that textures constitute the most appropriate base layer for coherent stylization of 3D animations. Not only is the static non-photorealistic rendering of lines and shaded regions deeply enriched by the media and patterns provided by textures, but also the dynamic behavior of the stylization benefits from them, as textures allow smooth – if not seamless – spatial and temporal modifications or transitions.

1. Summary of Contributions

The first contribution of this dissertation is the *formalization* of the temporal coherence problem in NPR and its *evaluation* in the case of shaded regions. By decomposing this problem into three goals – *flatness*, *motion coherence* and *temporal continuity* –, previous work as well as the new solutions I proposed with my collaborators can be compared on a well defined basis. Moreover this decomposition allows more precise and relevant perceptual experiments than speculative visual inspection or aesthetic considerations.

The second contribution is two *parameterization schemes* for view-dependent lines. The first easy-to-implement propagation mechanism targets smooth and simple 3D models. The second approach handles robustly complex geometry as well as lines extracted in image space by decoupling tracking and stylization.

The last contribution of this work consists in three solutions to deal with the huge depth disparities and zoom variations encountered in most 3D scenes.

- The first approach focuses on stroke textures and uses parametric texture synthesis to automatically generate *self-similar texture pyramids* allowing for infinite zoom on textured lines.
- The second solution *fractalizes* solid textures on the fly to ensure a quasi-constant size of the pattern in screen space at any zoom level.

Figure 12.1: Monkey whose shaded regions are rendered using NPR Gabor noise and lines using snakes.

- The last method relies on procedural noise and introduces a new level of detail mechanism which perfectly preserves the 2D characteristics of the texture without introducing temporal artifacts.

The combination of these two last contributions allows users to create a wide range of styles running at interactive framerate (Fig. 12.1). The results produced that way – in particular their animated version – demonstrate the validity of the thesis supported in this work.

2. Perspectives

While the methods I presented in this manuscript already have potential practical applications, I hope that they will also pave the way for additional research. Besides the particular future work that is directly related to our contributions and has already been discussed in the according chapters, I see four main directions for improvement and extension.

Stylization by optimization. During my PhD, I tried to explore as much as possible the triangle delimited by the three goals defining the temporal coherence problem. Nevertheless, all the methods proposed so far – including mine – are fixed trade-offs in this triangle. Consequently the space of potential compromises is restricted to a relatively sparse set of techniques.

It would be useful to travel continuously in this space. First, it would give to the artist a precise control on the temporal coherence compromise. Then, it would allow smooth transitions between different trade-offs inside the same image or during temporal sequences. For example, the artist may choose to enforce motion coherence for foreground objects while favoring flat backgrounds.

■ *A versatile system would allow to explore continuously the space of compromises.*

To allow such effects, we need to design a versatile system with direct and continuous controls on the three axes of the compromise. It requires the objective, quantitative and automatic measurement of the fulfillment of each goal. Then, based on these measures, the system needs to *optimize* locally (in space and time) the stylization process in order to tend toward the artist's intent. It raises numerous questions: Which are the "right" – meaning perceptually grounded – metrics? How to turn the stylization process into a general optimization problem? What is the most appropriate representation to solve it?

Similarly to most visual attention models [Itt00, RTAK07] and quality assessment frameworks [FP04, AvMS10], this system is bottom-up, i. e., low-level measurements are integrated to build a complex metric. The integration step raises an additional difficulty: How to combine the three metrics meaningfully? Simple linear combination does not seem the right approach. Not only are the three goals unequally perceived, but also temporal effects may change the perception of spatial ones (motion and flatness for instance). Moreover other perceptual phenomena, such as *change blindness* [SR05], indicate that visual attention is needed to see changes. If these effects complicate metrics integration, it could be possible to take advantage of them to deceive human perception during necessary changes.

Many-mark methods seem a promising approach to implement this versatile system. They share local control with few-mark techniques, while keeping the main advantages of texture-based approaches (i. e., temporal and spatial continuity). For example, it should be possible to create a seamless interpolation between

the two extreme cases of surface Gabor noise and NPR Gabor noise by progressively tilting the Gabor kernels from the surface tangent plane toward the camera plane.

Perceptual evaluation. The versatile system described above could not only allow to improve the stylization, but also the perceptual evaluation. With such a system, we would be able to sample more densely the space of solutions, producing more experimental stimuli. We then could design a user study in the spirit of Willis et al. [WAKB09] to evaluate the relevance of the goals we defined a priori.

Willis et al. [WAKB09] construct a spatial embedding for gloss starting from pairwise comparisons of rendered images with constant geometry and illumination but with varying BRDFs. The participants of the study are shown triplets of such images and asked to indicate the two most similar among them. Using a new multidimensional scaling algorithm, they derive a 2D embedding from these comparisons (without any initial a priori about its dimension), and they study its correlation with formal gloss measurements.

■ *A perceptual space for temporal coherence could validate our decomposition into three goals.*

In a similar way, it should be possible to create an embedding of the temporal coherence problem by pairwise comparison of stylized animations. The dimensionality of this embedding would indicate if our decomposition into three goals is perceptually grounded. The location of the stimuli along each axis would validate the objective metrics used to produce the stylized sequences.

Artistic control. The methods described in this dissertation are all targeting interactive applications. They have the benefit of giving an instantaneous feedback to the artist when he manipulates the viewpoint or certain style parameters in a WYSIWYG fashion. However it prevents fine grain controls and local corrections along an animation sequence, which are usually required for high quality productions. Conversely, in the case of an off-line pipeline, the full animation (camera and objects motions) is available, and this knowledge could be capitalized to allow such features.

Our snakes seem already a suitable representation to reach this goal. They are persistent in 2D and partially independent of the underlying 3D geometry. Consequently, an artist could correct the automatic behavior of the snakes by putting hard constraints on them (e. g., position, topology, existence) at specific key frames. Similar tools are provided by the assisted rotoscoping system of Agrawala et al. [AHSS04], except that their contours cannot change topology. The main challenge is then to propagate these constraints backward and forward in time while preserving the locality of these modifications. Neither our greedy algorithm nor a fully global optimization [AHSS04] seem appropriate to ensure these two properties, which implies to find an in-between solution.

■ *Local corrections need to be propagated in time with sufficient locality.*

The problem is even more complex for shaded regions as they are not as "easily" delineated as contours. Segmentation-based techniques applied on screen space deep buffers may provide an appropriate intermediate representation between the 3D geometry and final stylized image. Controls on the motion field (e. g., simplification, edition) and style parameters inside the regions extracted by these methods could allow local modifications. Once again, it raises the question of their temporal propagation. The work of Lin et al. [LZL*10], Kagaya et al. [KBD*10] and O'Donovan and Hertzmann [OH11] on painterly rendering from video are promising source of inspiration.

■ *Style transfer functions require more intuitive and controllable interfaces.*

Besides controls on the temporal behavior of the stylization, the definition of the style by artists – that is the transfer function from scene primitives to marks attributes – need further work. Although it was not the focus of this thesis, the stylization techniques we proposed use the three main approaches for style definition: direct control of the parameters (sliders, dials and knobs), programming (shaders and scripts) and "by example" (e. g., texture synthesis). Each of them could be improved independently:

- More intuitive interfaces in conjunction with direct interaction techniques, such as sketching, could be developed to design self-similar textures or guide texture synthesis output.
- The creation of procedural patterns with NPR Gabor noise would benefit from the automatic estimation of noise parameters matching a texture exemplar [LVLD10, GD10, JCW11].

But an even higher challenge is to combine different approaches for defining styles. Realistic textures design techniques – Allegorithmic© "smart textures" which are mixing raster and vector graphics with procedural elements, for instance – could be an interesting source of inspiration.

2D animation. With the exception of SLAMs, this thesis only focused on 3D animations and strove to give back a 2D appearance to stylized images rendered from them. This approach has undeniable advantages, described in the introduction, but it seems reasonable to consider other inputs, in particular 2D vector animations (videos have already been discussed in previous chapters). Current software (e. g., Flash©, ToonBoom©) provide minimal tools to enrich 2D vector animations with media or patterns. At most they allow to map a rigid 2D texture inside a closed shape, but this texture is not affected by the deformations that the shape may undergo during animation, leading to showerdoor or sliding artifacts.

Two recent approaches based on diffusion curves [OBW*08] develop more involved techniques to add texture on static 2D vector art. Jeschke et al. [JCW11] use spatially varying Gabor noise [LLD10] defined along the curves to generate stochastic and irregular texture details. Although not shown in that paper, keyframed animations can probably be produced with this method, and they should be relatively similar to the results obtained with NPR Gabor noise. Winnemöller et al. [WOBT09] propose tools to design and manipulate regular and near-regular vector patterns in an intuitive way. Adapting this approach to animation is not trivial as their texture "draping" needs to be tuned by hand for each image.

■ *Enrich 2D animations with media is a promising but challenging avenue for future work.*

Extending these methods to a broader class of patterns and even to general raster textures raises major challenges: How to link texture and shape together? How to modify the texture according to the deformation of the shape during the animation? How to measure this deformation – taking into account perceptual effects – and regenerate the texture to avoid too large distortions?

If previous work, such as texture advection, could be reused in this context, the vectorial nature of the input should allow to define better suited solutions. Recent image interpolation and morphing techniques [HF06, WBCG09, BBA09, WG10] are showing promising results for drawings. I believe that they could be used for transferring the deformation of the shape to the texture.

Appendix

A

INFINITE ZOOM SHADERS

```
uniform float obj_scale;
varying float dist;

void main(void)
{
  //distance to the camera
  vec4 posTransform = gl_ModelViewMatrix*gl_Vertex;
  dist = abs(posTransform.z);

  //volumetric texture coordinate
  gl_TexCoord[0] = gl_Vertex/obj_scale;

  //projected position
  gl_Position = ftransform();
}
```

Listing A.1: Vertex Shader

```
uniform sampler3D solidTex;
varying float dist;

vec4 main(void)
{
  //number of zoom cycles
  float z = log2(dist);
  float s = z-floor(z);

  //scale factor according to fragment distance to the camera
  float frag_scale = pow(2.0,floor(z));

  //octave weight according to the interpolation factor
  float alpha1 = s/2.0;
  float alpha2 = 1.0/2.0-s/6.0;
  float alpha3 = 1.0/3.0-s/6.0;
  float alpha4 = 1.0/6.0-s/6.0;

  //texture lookup
  vec4 oct1 = alpha1*texture3D(solidTex, gl_TexCoord[0].xyz/frag_scale)
      ;
  vec4 oct2 = alpha2*texture3D(solidTex, 2.0*gl_TexCoord[0].xyz/
      frag_scale);
  vec4 oct3 = alpha3*texture3D(solidTex, 4.0*gl_TexCoord[0].xyz/
      frag_scale);
  vec4 oct4 = alpha4*texture3D(solidTex, 8.0*gl_TexCoord[0].xyz/
      frag_scale);

  //blending
  vec4 n = oct1+oct2+oct3+oct4;

  gl_FragColor = n;
}
```

Listing A.2: Fragment Shader

B

NPR GABOR NOISE: STYLES COOKBOOK

OUR RENDERING algorithm for temporally coherent stylization is a two-step process (see Fig. B.1) based on deferred shading. In the first step, we generate one or more noise layers using NPR Gabor noise. This is done using a general noise shader. We can use scene attributes (such as shading, surface curvature [Rus04] and object ID) to locally control the parameters of the noise (frequency, bandwidth and orientation). In the second step, we composite the noise layers to produce the final result. This is done using a style-specific shader.

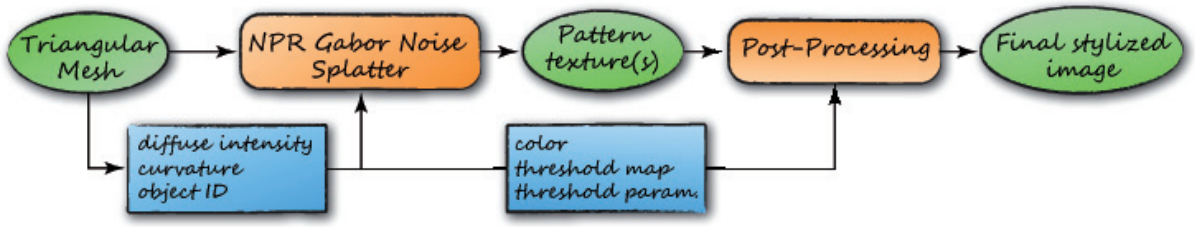


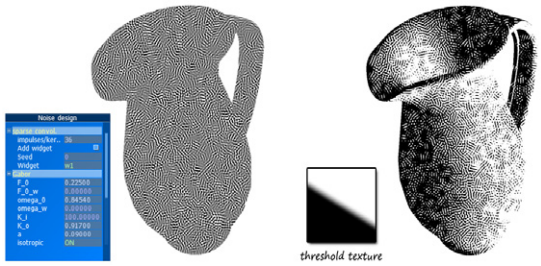
Figure B.1: Pipeline of our rendering algorithm for temporally coherent stylization using NPR Gabor Noise.

In the remainder of this appendix, we give a detailed description of the individual styles (Sec. 1), including the threshold function we used for several styles (Sec. 2), and the compositing functions we used (Sec. 3).

1. Styles

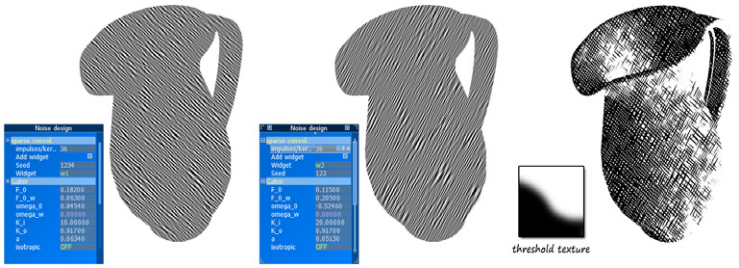
In this section we provide pseudo-code for the style-specific shaders as well as the parameters that we used to produce the styles shown in the paper.

Stippling



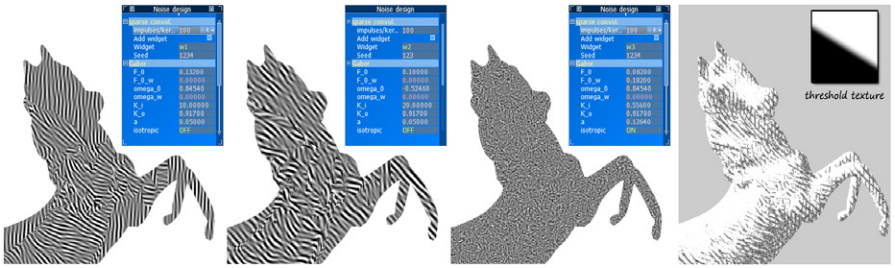

```
float pattern = texture(noise_texture0, texCoord).r;
vec3  stroke  = texture(thresholdMap,vec2(pattern1,diffuseIntensity)).
    rgb;

vec3  final_color = stroke;
```

Cross-Hatching



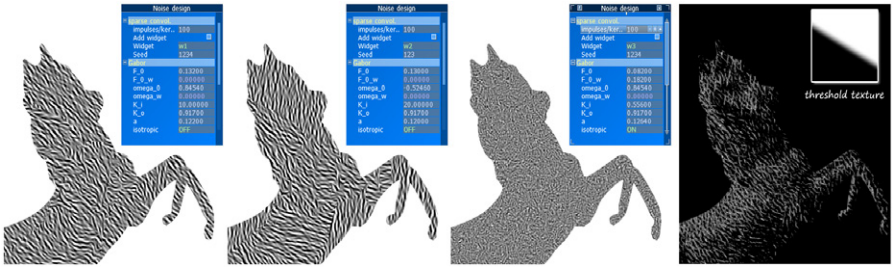

```
vec3 stroke1 = texture(thresholdMap,vec2(pattern1,diffuseIntensity)).
    rgb;
vec3 stroke2 = texture(thresholdMap,vec2(pattern2,diffuseIntensity)).
    rgb;

vec3 final_color = stroke1*stroke2;
```

Graphite



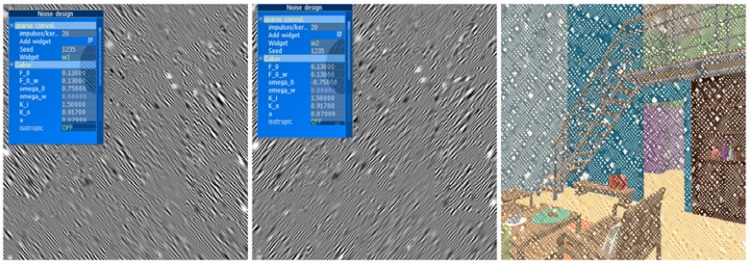

```
float pattern1 = texture(noise_texture0, texCoord).r;
float pattern2 = texture(noise_texture1, texCoord).r;
float pattern3 = texture(noise_texture2, texCoord).r;

vec3  stroke1 = texture(thresholdMap,vec2((pattern1+pattern3)/2.0,
    diffuseIntensity)).rgb;
vec3  stroke2 = texture(thresholdMap,vec2((pattern2+pattern3)/2.0,
    diffuseIntensity)).rgb;

vec3  final_color = (1.0-(1.0-stroke1*stroke2)*pattern3);

if(isBackground) final_color = vec3(0.8);
```

Chalk



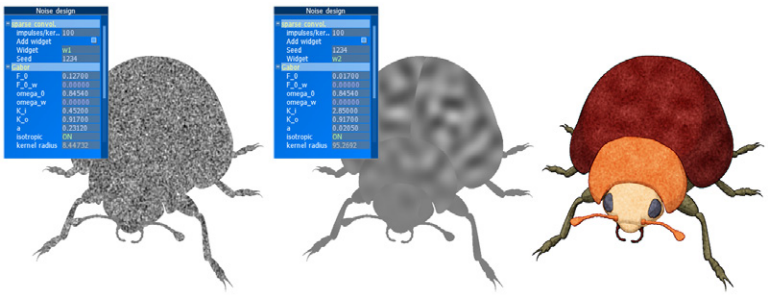
```
float pattern1 = texture(noise_texture0, texCoord).r;
float pattern2 = texture(noise_texture1, texCoord).r;
float pattern3 = texture(noise_texture2, texCoord).r;

vec3 stroke1 = texture(thresholdMap,vec2((pattern1+pattern3)/2.0,
    diffuseIntensity)).rgb;
vec3 stroke2 = texture(thresholdMap,vec2((pattern2+pattern3)/2.0,
    diffuseIntensity)).rgb;

vec3 final_color = 1.0-(1.0-(1.0-stroke1*stroke2)*pattern3);

if(isBackground) final_color = vec3(0.0);
```

Color Strokes

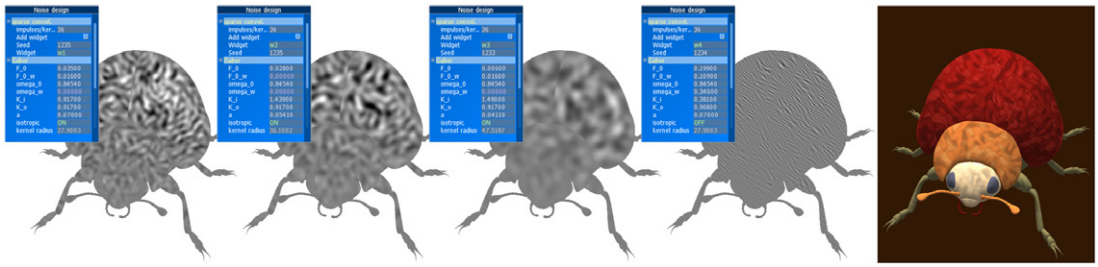


```
float pattern1 = texture(noise_texture0, texCoord).r;
float pattern2 = texture(noise_texture1, texCoord).r;

vec3 background = vec3(0.99,0.96,0.9);
vec3 final_color = background;

float alpha = threshold(pattern1, 25, 0.5);
final_color = alphaBlend(final_color, overlay(color*0.85,pattern1,1.2)
    , alpha);

alpha = threshold(pattern2, 25, 0.5);
final_color = alphaBlend(final_color, overlay(color,pattern2,1.0),
    alpha);

if(isBackground) final_color = background;
```

Watercolor



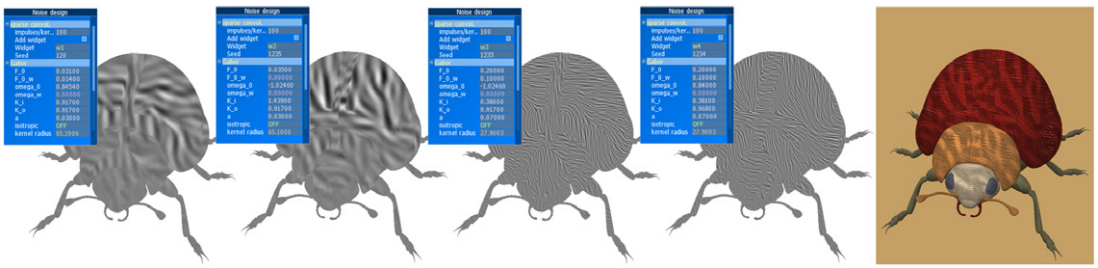
We use a watercolor stylization algorithm similar to the algorithm by Bousseau et al. [BKTS06], using one turbulence texture and one pigment texture.



Ink on Canvas

```
float pattern1 = texture(noise_texture0, texCoord).r;
float pattern2 = texture(noise_texture1, texCoord).r;
float pattern3 = texture(noise_texture2, texCoord).r;
float pattern4 = texture(noise_texture3, texCoord).r;

vec3  final_color = color;

float alpha = threshold(pattern3, 25, 0.5);
final_color = alphaBlend(final_color, overlay(color*0.9,pattern3,1.0),
     alpha);

alpha = threshold(pattern2, 25, 0.5);
final_color = alphaBlend(final_color, overlay(color*0.95,pattern2,1.0)
    , alpha);

alpha = threshold(pattern1, 25, 0.5);
final_color = alphaBlend(final_color, overlay(color*1.05,pattern1,1.0)
    , alpha);

final_color = overlay(final_color,pattern4,1.2);

if(isBackground) final_color = vec3(0.17,0.1,0.017);
```

Painterly



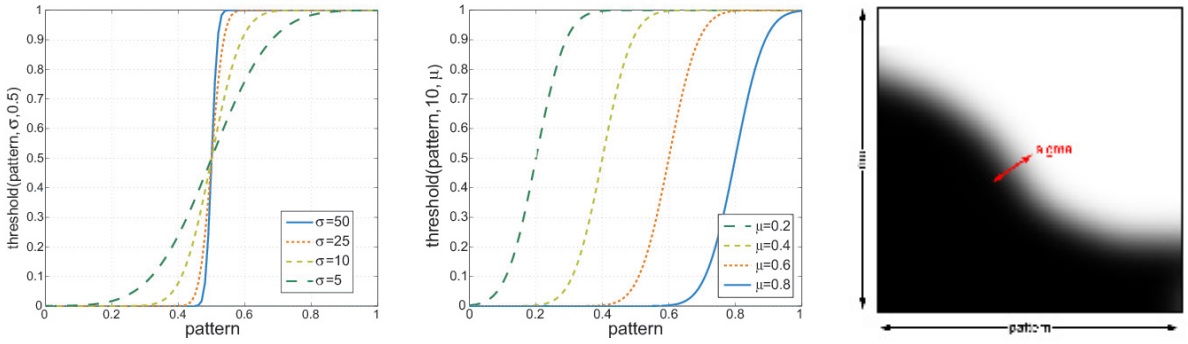
```
float pattern1 = texture(noise_texture0, texCoord).r;
float pattern2 = texture(noise_texture1, texCoord).r;
float pattern3 = texture(noise_texture2, texCoord).r;
vec3  pattern4 = texture(noise_texture3, texCoord).rgb;

vec3  final_color = overlay(color, pattern3, 0.7);

float alpha1 = threshold(pattern1, 25, 0.5);
vec3 stroke1_color = overlayAlpha(final_color, color*0.8, pattern4,
    1.2, alpha1);

float alpha2 = threshold(pattern2, 25, 0.5);
final_color = overlayAlpha(stroke1_color, color*0.9, pattern3, 1.2,
    alpha2);
```

```
if(isBackground) final_color = vec3(0.96,0.78,0.5);
```

We can add a bump mapping effect to emphasize the brush fiber layers, similar to the algorithm by Hertzmann [Her02].

2. Threshold Function

We use thresholding for various styles. The threshold function is a sigmoid-shaped function. We use a threshold function based on the error function, the integral of a Gaussian function, with parameters μ and σ, which control the position and the smoothness of the threshold respectively (see Fig B.2).



(a) Smoothness parameter (σ) (b) Position parameter (μ) (c) Threshold Map

Figure B.2: Threshold functions: (a,b) threshold curves; (c) threshold map

Instead of a threshold function, we can also use a *threshold map* (see Fig. B.2c), an X-toon texture [BTM06] that graphically defines a spatially varying relationship between the threshold function and another variable (for example, shading).

```
float threshold(float pattern, float sigma, float mu)
{
   return (erf((sigma * (pattern - mu))) + 1.0) / 2.0;
}
```

Listing B.1: Threshold Function

3. Compositing functions

We used the following functions for compositing the noise layers.

Alpha Blending

```
vec3 alphaBlend(vec3 initColor, vec3 colorToBlend, float alpha)
{
  return (1.0-alpha) * initColor + alpha * colorToBlend;
}
```

Overlay

```
vec3 overlay(vec3 initColor, vec3 colorToOverlay, float overlayFactor)
{
  return initColor*overlayFactor*(1.0-(1.0-initColor)*(1.0-
      colorToOverlay));
}
```

Overlay - Alpha

```
vec3 overlayAlpha(vec3 backColor, vec3 initColor, vec3
colorToOverlay, float overlayFactor, float alpha)
{
  return backColor*(1.0-alpha) + alpha*initColor*overlayFactor
      *(1.0-(1.0-initColor*alpha)*(1.0-colorToOverlay));
}
```

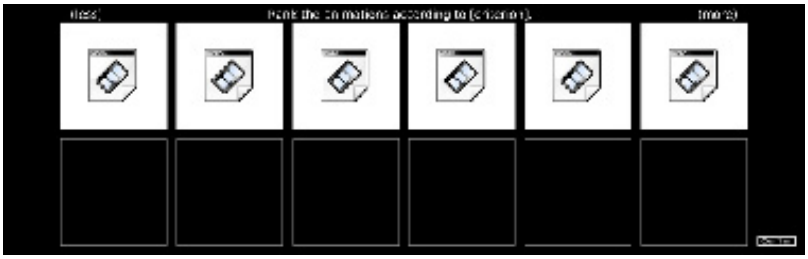
C

USER STUDY INSTRUCTIONS

THE FOLLOWING instructions were provided to the participants of our user study to evaluate and compare regions stylization methods (Chapter 11).

Cartoon Animation Perception Survey

The purpose of this survey is to understand how you perceive different variations of a cartoon-like animation style.



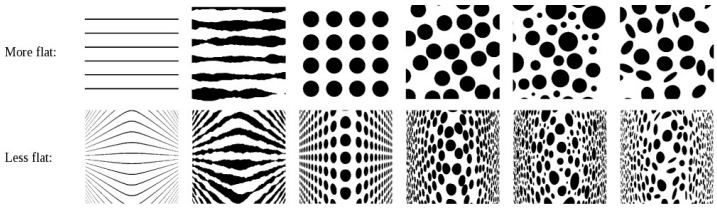
We will show 6 images or animation sequences simultaneously. You will be asked to rank them according to a number of very specific criteria, by drag-and-dropping them into the order you decide. A second row of slots is provided for shuffling items around. The items must form a single horizontal axis, i.e. no two items in a single column, before you can continue to the next criterion.

There are no right or wrong answers; the answer reflects only your opinion. If you are unsure about the ranking of some of the items, take your best guess or just put them in a random order.

There are 4 different tasks:

1. "Rank the images according to how flat they appear."

   We provide the following images (taken from [TTD*07, Fig. 7]) to clarify this task:

2. "Rank the animations according to how coherently the pattern moves with the object."
3. "Regardless of the coherence of the motion of the object and the pattern, which you have already evaluated in the previous task,

rank the animations according to how much the pattern changes otherwise over time."

4. "Rank the animations according to how pleasant you find them in the context of cartoon animation." (second part only)

You will do 7 rankings (first part) / 4 rankings (second part) in total. The complete survey will typically take less than 30 minutes (first part) / 10 minutes (second part).

Feel free to take a break at any time. You may quit the survey anytime, without having to give a reason and without detriment to you.

Thank you for taking part in this survey!

The post-study questionnaire consisted of the following questions:

Questionnaire

- How confident are you in your rankings for each criterion?
- Were there any criteria you didn't understand? If so, describe your interpretation.
- Were there any ambiguously formulated criteria? If so, describe the ambiguity.
- Were there any criteria for which it was very difficult to rank the video sequences? If so, describe the difficulty.
- Are there other relevant criteria for cartoon animation that weren't covered? If so, describe.
- Was the survey's duration too long, about right, or too short? Were the tasks fun or tiresome?
- Did the written instructions cover everything you needed to know about the survey? What else should have been included?
- Were the written instructions clear? How could they be improved?

D

RÉSUMÉ EN FRANÇAIS

1. Introduction

La synthèse d'image consiste à générer une image à partir d'un modèle géométrique 3D, ou une collection de modèles organisés en une scène, en utilisant des algorithmes informatiques. Ce processus met en oeuvre trois composants principaux : (1) le transport de la lumière, (2) son interaction avec la matière, (3) un dispositif de capture. Depuis la naissance de l'informatique graphique, l'objectif de la synthèse d'image a été le photoréalisme. La synthèse d'une image photoréaliste implique la simulation physique du transport de la lumière, une définition physique des fonctions de réflectance et un dispositif de capture photographique, généralement caractérisé par une projection perspective et une lentille. Cette approche est motivée par la nécessité de créer des images ressemblantes à une photographie, en particulier pour les effets spéciaux dans l'industrie cinématographique, mais aussi pour le défi intrinsèque de reproduire la réalité.

Néanmoins, la photographie n'est pas en contradiction avec l'expressivité. Comme l'a remarqué Durand [Dur02], les photographes prennent des décisions artistiques lors de la prise de vue (point de vue, cadrage, composition, distance focale, exposition) et lors du développement du négatif (modification de la densité) pour exprimer leur vision de la réalité (Figure D.1b). Si la photographie a pris une part importante au sein des arts visuels, de nombreuses autres techniques picturales (dessin au trait, peinture, gravure, etc.) ont été utilisées, et sont encore utilisées, par les artistes pour leur *expressivité* et leur *force d'abstraction*.

La stylisation permet en effet d'omettre certains détails superflus tout en mettant en avant l'information importante de façon à guider le regard du spectateur. Les illustrations scientifiques et techniques en sont les meilleurs exemples. La stylisation peut également suggérer l'incertitude : les architectes dessinent fréquemment leurs premières esquisses au crayon et à l'aquarelle pour exploiter cette propriété. Par conséquent, pourquoi la synthèse d'image 3D devrait-elle se réduire au photoréalisme quand l'image numérique pourrait ouvrir de nouveaux horizons ?

1.1. Le rendu non-photoréaliste

C'est guidé par cette question que le rendu non-photoréaliste est né dans les années 90 et s'est efforcé à proposer des alternatives à chaque étape du processus de synthèse d'image. Ces manipulations visent à donner aux utilisateurs une large palette de possibilités pour transmettre leur impression de la scène, réelle ou imaginaire, qu'ils souhaitent représenter (Figure D.1c). C'est ce que nous appellerons la *stylisation*.

Stylisation. En étendant le modèle de Willats [Wil97], Durand [Dur02] propose de décomposer le processus de création d'une image en quatre systèmes :

(a) “Grand Teton National Park”, Latham Jenkins© (b) “The Tetons and the Snake River”, Ansel Adams© (c) Rendu aquarelle utilisant la méthode de Bousseau et al. [BKTS06]

FIGURE D.1: Deux photographies d’un sujet réaliste (a,b) peuvent avoir une atmosphère radicalement différente grâce à des manipulations artistiques. Les techniques de rendu expressif (c) ouvrent un champ de possibilités encore plus vaste.

le système de *marques* (coup de pinceaux ou de crayon, pigments d’aquarelle...) qui représente des *primitives* de la scène (points, lignes ou régions) à une certaine position *spatiale* avec des *attributs* (couleur, épaisseur, texture...). Les choix effectués par l’artiste pour chacun de ces systèmes définissent le style final de l’image produite. Il peut ainsi décider de représenter les silhouettes d’un objet en projection perspective avec des pointillés noirs. Il peut tout aussi bien choisir de représenter les régions de couleur projetées orthogonalement avec des hachures dont l’épaisseur varie avec l’ombrage.

En rendu non-photoréaliste, le système de marque représente non seulement le *médium* de l’image au sein littéral (canevas, papier), mais aussi le *motif* produit par l’outil de dessin (coup de pinceau, hachure, etc.). Ces médiums et motifs – dans leur version numérique – constituent le principal sujet de cette thèse.

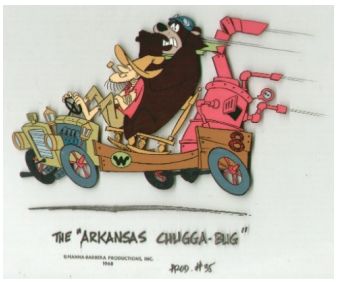
Bien que la stylisation soit inhérente au dessin ou à la peinture manuelle, sa version informatique a de nombreux avantages pour les utilisateurs débutants comme pour les infographistes professionnels. En effet, elle permet aux utilisateurs occasionnels de produire des images complexes en un temps limité et avec des compétences techniques restreintes – une partie ou l’intégralité du processus de stylisation étant automatisée. Elle permet également de répéter les mêmes opérations plusieurs fois sur différents modèles, ou de raffiner la définition d’un style en suivant une approche d’essais et erreurs. Enfin, elle rend possible la stylisation interactive d’animations en laissant un certain contrôle à l’utilisateur au cours du temps. Il est quasiment impossible d’assurer cette dernière propriété avec des médiums et techniques traditionnels.

L’animation 2D donne l’illusion d’un mouvement continu à partir d’une séquence d’images fixes. C’est une différence majeure avec la capture vidéo ou la synthèse d’image. En effet, ces deux techniques échantillonnent un processus continu à une fréquence suffisante pour qu’aucune discontinuité ne soit perceptible. À l’inverse, chaque image d’une animation manuelle est dessinée indépendemment. Par conséquent, contrôler l’évolution du médium tout au long de l’animation est complexe, voire impossible dans bon nombre de cas. Par exemple, les pigments d’aquarelle se déposent inévitablement à différentes positions entre deux images, produisant des scintillements. Les coups de pinceau apparaissent ou disparaissent brusquement en raison des occlusions et disocclusions. Ces effets sont généralement appelés *incohérences temporelles* en rendu non-photoréaliste.

Les artistes ont peu de solutions pour éviter ces incohérences. Ils peuvent utiliser plusieurs calques pour séparer les objets en mouvement au premier plan d'un arrière-plan fixe. Les dessins animés sont généralement réalisés sur des feuilles de celluloïd transparentes (Figure D.2a), et des médiums homogènes (gradients de couleurs et fines bordures noires) sont utilisées pour les parties animées de la scène afin de limiter les artefacts temporels. Les arrières-plans statiques peuvent être représentés avec des styles plus riches (Figure D.2b). De la même façon, Alexander Petrov utilise des plaques de verre pour réaliser ses animations à la peinture l'huile (Figure D.2c). Ainsi, il n'a pas besoin de redessiner l'intégralité de la peinture à chaque image et peut contrôler, dans une certaine mesure, les scintillements des coups de pinceau apparaissant et disparaissant entre deux images. Il lui a néanmoins fallu deux ans et demi pour réaliser les vingt minutes du « Vieil homme et la mer ».

La dernière solution est d'intégrer les incohérences temporelles dans le style. Par exemple, Georges Schwizgebel ajoute explicitement des oscillations à ses coups de pinceau pour masquer les incohérences temporelles et donner ainsi un aspect très vivant à ses animations (Figure 1.2d).

Animations 3D. Les approches 2D manuelles ne sont généralement pas envisageables pour les séries de dessins animés. Les contraintes sont telles que le processus de rendu doit être automatisé autant que possible. Cela explique le succès actuel de la modélisation, animation et synthèse 3D. Les personnages et objets ne sont modélisés qu'une fois, dans différentes poses ou avec un squelette d'animation, et peuvent ensuite être réutilisés à de nombreuses reprises. Une grande partie de l'animation et de la mise en lumière peut être scriptée pour simplifier et accélérer le travail de l'infographiste. En contrepartie, de nombreuses séries utilisent des techniques de rendu basiques ce qui leur donne un aspect 3D relativement similaire en dépit de leurs différences en termes de design ou d'animation (Figure D.3a,b).



(a) "The Wacky Racers", Hanna-Barbera©

(b) "The Secret of Kells", Tomm Moore©

(c) Peinture sur verre d'Alexander Petrov technique

(d) "L'homme sans ombre", Georges Schwizgebel©

FIGURE D.2: Exemples d'animations manuelles.

(a) “Oui-Oui 3D”©

(b) “Mickey Mouse Club House”, Disney©

(c) “Iron Man : Armored Adventures”, Marvel©

FIGURE D.3: Exemples de dessins animés générés par ordinateur.

Néanmoins, certaines séries utilisent des techniques de rendu alternatives, comme *l'ombrage de celluloïd* ou « toon shading » (Figure 1.3c). Cela leur donne assurément un aspect « cartoon », mais des médiums plus complexes sont encore peu utilisés. Cela s'explique par le fait que les deux solutions naïves pour styliser des animations 3D ne sont pas satisfaisantes. La première est le *placage de texture* qui consiste à coller le médium directement à la surface des modèles 3D. Le médium suit parfaitement le mouvement des objets – évitant ainsi toute discontinuité temporelle – mais sont apparence 2D n'est pas préservée. Généralement, les artistes ne dessinent ou ne peignent pas à la surface des objets 3D, mais représentent des primitives 3D à l'aide de marques 2D sur un papier ou canevas plat. La seconde approche naïve couvre donc l'image avec un médium statique qui ne suit pas le mouvement de la scène (comme un filtre photo sur une caméra). L'aspect 2D du médium est bien préservé, mais cette approche brise la connexion entre les marques et les primitives ce qui produit d'importants *glissements*. Les discontinuités temporelles, l'aspect 3D et les glissements sont les trois artefacts englobés par le *problème de la cohérence temporelle* en rendu non-photoréaliste.

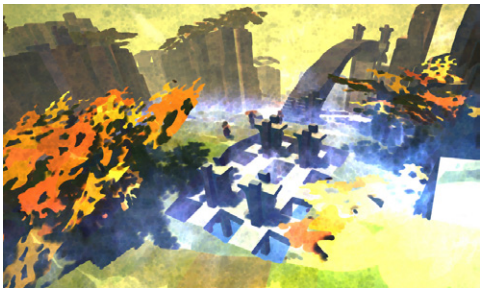
Jeux vidéo. L'industrie du jeu vidéo s'intéresse également au rendu non-photoréaliste. Outre le « toon shading » – abondamment utilisé depuis l'année 2000 – certains studios ont essayé de développer des styles plus complexes combinant un modèle d'illumination non-photoréaliste, le tracé des silhouettes et des textures peintes (Figure D.4a-c). La stylisation aide non seulement ces jeux à sortir de l'ordinaire, mais elle peut aussi améliorer l'immersion du joueur, en particulier sur des dispositifs aux capacités de calcul limitées (Wii, PSP). Un effet similaire à l'« unncany valley », introduit par Mashiro Mori pour les robots humanoïdes, peut en effet se produire : plus le rendu est réaliste, plus les attentes des joueurs sont grandes en termes de graphisme, mais aussi d'animation et de comportement des personnages virtuels. Avec un rendu stylisé, ces contraintes peuvent être relâchées.

Néanmoins, les styles utilisés par ces jeux restent relativement simples. Le jeu « Love » d'Eskil Steenberg fait exception avec son style pointilliste plus abstrait (Figure D.4d). Malheureusement, il souffre de nombreuses instabilités temporelles (scintillements) qui peuvent être très dérangeantes.

(a) “Prince of Persia”, Ubisoft©

(b) “Borderland”, Gearbox software©

(c) “Okami”, Clover Studios©



(d) “Love”, Eskil Steenberg©

FIGURE D.4: Exemples de jeux vidéo utilisant des techniques de rendu expressif.

1.2. Le problème de la cohérence temporelle

Notre définition du problème de la cohérence temporelle en rendu non-photoréaliste inclus aussi bien des aspects spatiaux que temporels. Plus précisément, il implique de satisfaire simultanément trois objectifs : le respect de l’*aspect 2D*, de la*cohérence du mouvement* et une suffisante *continuité temporelle*.

L’aspect 2D est la capacité à transmettre la nature 2D de la stylisation. Comme l’a montré Meier [Mei96], l’objectif est de donner l’impression que chaque image est produite sur un médium plat plutôt que peint à la surface d’un objet 3D. Les propriétés 2D du médium doivent être respectées : la taille du motif, sa distribution, l’homogénéité du contraste.

La cohérence du mouvement représente la forte corrélation entre le mouvement de la scène 3D et celui du médium. Comme l’a énoncé Cunzi et al. [CTP*03], l’objectif est de donner l’impression que le mouvement du médium en espace image est aussi proche que possible du déplacement en espace objet. A l’inverse, quand le médium est statique à l’écran, l’effet *rideau de douche* apparaît : la scène semble glisser sous le médium.

La continuité temporelle est la qualité de minimiser les changements d’une image à l’autre pour assurer une animation fluide. Des études perceptuelles [YJ84, SS09] ont montré que nous sommes très sensibles aux brusques discontinuités temporelles, comme le *popping*. L’objectif est donc de maintenir une stylisation aussi continue que possible, ce qui est particulièrement important lors du zoom et aux limites d’occlusion et de desocclusion.

Malheureusement, ces objectifs sont intrinsèquement contradictoires pour les médiums complexe. Respecter parfaitement l’aspect 2D tout en suivant précisément le mouvement 3D sans introduire de discontinuité temporelle semble impos-

sible. Par conséquent, toutes les méthodes abordant le problème de la cohérence temporelle sont *nécessairement des compromis*. Cela pose plusieurs questions auxquelles nous allons tenter de répondre dans ce manuscrit : comment évaluer le compromis proposé par les différentes méthodes de stylisation ? Sur quels objectifs peut-on transiger avec le moindre impact sur le résultat final ? Comment explorer l'espace des solutions délimité par ces objectifs ?

1.3. Contributions

La thèse présentée dans ce manuscrit est double. Nous défendons que la *texture* est une représentation appropriée pour de nombreux médiums, stochastiques (e. g., pigment d'aquarelle) comme plus structurés (e. g., hachures). Nous affirmons également que les textures permettent de créer des techniques de stylisation fournissant un compromis équilibré entre l'aspect 2D, la cohérence du mouvement et la continuité temporelle.

Nous démontrerons la validité de cette thèse pour deux primitives : les lignes et les régions de couleur extraites de modèles 3D animés.

Pour la production de dessins au trait animés (section 3), nous proposons une nouvelle pyramide de textures autosimilaire qui permet de zoomer infiniment sur une ligne texturée sans introduire de déformation de son apparence 2D. Nous décrivons une méthode de synthèse automatique d'une telle pyramide d'après une texture d'exemple. Nous présentons ensuite deux méthodes pour paramétrer de façon temporellement cohérente des lignes dépendantes du point de vue. La plus simple des deux s'applique aux modèles lisses et propage la paramétrisation de la ligne d'une image à la suivante par un mécanisme de vote robuste. La seconde méthode traite les cas des modèles plus complexes en utilisant une représentation explicite et persistante des lignes extraites.

Dans la seconde partie de ce manuscrit (section 4), nous proposons deux compromis pour la stylisation temporellement cohérente des régions de couleur. Le premier utilise le placage de texture pour assure la cohérence du mouvement et étend un précédent algorithme de fractalisation de texture en espace image pour mieux préserver l'aspect 2D du médium. Visant les applications temps-réel, cette méthode s'insère parfaitement dans un moteur de jeu. La deuxième approche est basée sur une fonction de bruit définie à la surface des objets 3D mais évaluée en espace image pour préserver les caractéristiques 2D du médium. De par sa définition procédurale, cette méthode évite le problème du placage de texture de la première approche, préservant ainsi parfaitement l'aspect 2D.

Enfin, nous discutons de la question complexe de l'évaluation de telles techniques et présentons deux études perceptuelles que nous avons conduites dans ce contexte (section 5). La première vise à évaluer la fractalisation de texture et nous permet de déduire de ses résultats une mesure objective de qualité. Durant la seconde étude, nous comparons six méthodes de stylisation des régions en nous basant sur notre décomposition en trois objectifs. Cette étude montre non seulement l'efficacité des solutions que nous avons proposées, mais elle ouvre aussi de nouvelles perspectives pour des travaux futurs discutés en conclusion.

2. État de l'art

Cette section examine les principales contributions en rendu non-photoréaliste pour la stylisation temporellement cohérente d'animations 3D. Nous invitons le lecteur à consulter les livres de Gooch et Gooch [GG01] et Strothotte et Schlechtweg [SS02] pour avoir un aperçu général des techniques de rendu non-photoréaliste, en particulier pour les images fixes.

Nous considérons séparément le cas des lignes et celui des régions de couleur. Dans le contexte d'une animation 3D, ces deux primitives partagent néanmoins un problème commun : le cas problématique du zoom.

2.1. Stylisation cohérente des lignes

De nombreux efforts se sont concentrés sur l'extraction et le rendu de lignes (voir la bibliographie annotée de Rusinkiewicz et al. [RCDF08]). Néanmoins, la cohérence temporelle des lignes stylisées reste un problème majeur. Il est possible de faire la distinction entre deux approches principales pour résoudre ce problème : les méthodes en espace image et celle en espace objet. Ces dernières sont les seules permettant de simuler des effets de coup de pinceau complexes en plaquant une texture sur les lignes.

2.2. Stylisation cohérente des régions

Nous classons les méthodes précédentes pour la stylisation cohérente des régions en deux catégories principales : celle basée sur une distribution de marques et celle utilisant des textures. Choisir l'une ou l'autre de ces approches a d'importantes conséquences. La première classe de méthodes tend à sacrifier la continuité temporelle au profit des autres objectifs, tandis que la seconde classe d'approches sacrifie soit la cohérence du mouvement, soit l'aspect 2D.

Différentes stratégies ont été proposées pour gérer le cas critique du zoom dont l'amplitude peut être très grande, voire quasi infinie. L'objectif contradictoire du mécanisme de zoom infini est de maintenir une taille globalement constante de la texture à l'écran, tout en préservant l'impression de grossissement / rétrécissement du motif lors du déplacement en profondeur de la caméra.

2.3. Bilan

Bien que la cohérence temporelle soit un problème important en rendu non-photoréaliste et qu'il ait reçu une attention croissante durant ces quinze dernières années, les solutions existantes peuvent être grandement améliorer, et des défis majeurs sont encore à relever.

Des progrès peuvent être faits vis à vis des trois objectifs de la cohérence temporelle en cherchant des compromis plus équilibrés. Au-delà de ces objectifs, ces nouvelles solutions doivent prendre en compte de nombreuses contraintes supplémentaires telles que les performances, la fiabilité, le passage à l'échelle, le contrôle artistique ou la diversité des styles représentables.

Simultanément, la nécessité d'une évaluation formelle de ces techniques est criante. En dépit de l'absence d'une solution de référence, des mesures perceptuelles des artefacts produits par chacun des compromis devraient permettre de déduire des tendances objectives à partir des jugements subjectifs d'utilisateurs.

3. Stylisation des lignes

La stylisation fait partie intégrante du rendu de dessins au trait. Les artistes modifient l'épaisseur, la texture ou la précision des traits pour transmettre la forme, le mouvement ou abstraire leurs dessins. Des pointillés peuvent ainsi suggérer un objet invisible, tandis qu'un contour esquissé par plusieurs traits peut indiquer le mouvement.

L'approche générale pour simuler ces effets en informatique graphique est de définir une courbe paramétrique (appelée *chemin*) et d'appliquer une texture de coup de pinceau le long de cette courbe. Ces textures incluent les pointillés, les tirets et tous les motifs produits par l'interaction d'un crayon ou d'un stylo avec le papier. Elles renforcent l'impression que ces dessins ont été produits à la main.

Dans cette section, nous nous intéressons aux méthodes permettant de tels effets durant l'animation de modèles 2D et 3D. Le principal défi est de définir une paramétrisation temporellement cohérente des lignes animées. Pour résoudre le problème du zoom dans le cas des lignes ne dépendant pas du point de vue, nous présentons tout d'abord une nouvelle structure de données et un algorithme de synthèse par l'exemple de cette structure. Nous proposons ensuite deux méthodes pour propager une paramétrisation cohérente des lignes dépendantes du point de vue durant un mouvement 3D quelconque.

3.1. Pyramides de textures autosimilaires

L'un des deux artefacts suivants intervient lors de la stylisation de lignes animées : le glissement ou l'étirement de la texture de coup de pinceau lorsque la longueur de la ligne est modifiée. Nous proposons une nouvelle représentation de la texture de coup de pinceau, sous forme d'une pyramide de textures autosimilaires – « Self-Similar Line Artmap » (ou SLAM, Figure D.5) – qui évite ces artefacts. Cette pyramide permet un zoom infini et continu sur un dessin au trait tout en préservant l'apparence de la texture de coup de pinceau en espace image. Elle peut être générée automatiquement d'après une unique texture d'exemple.

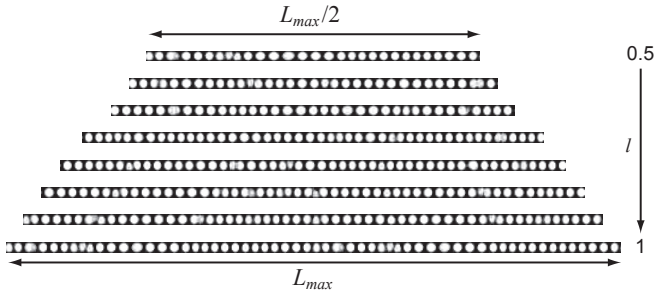
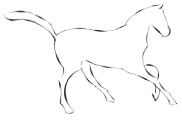


FIGURE D.5: Exemple d'une SLAM pour une texture de pointillés. Cette pyramide compte huit niveaux, mais un nombre arbitraire de textures pourrait être généré. Lorsque la taille de la texture augmente, les pointillés s'étirent jusqu'à se diviser en plus petits points. La taille irrégulière des pointillés est nécessaire pour assurer la cohérence temporelle ; les autres imperfections sont dues à la synthèse de texture paramétrique.



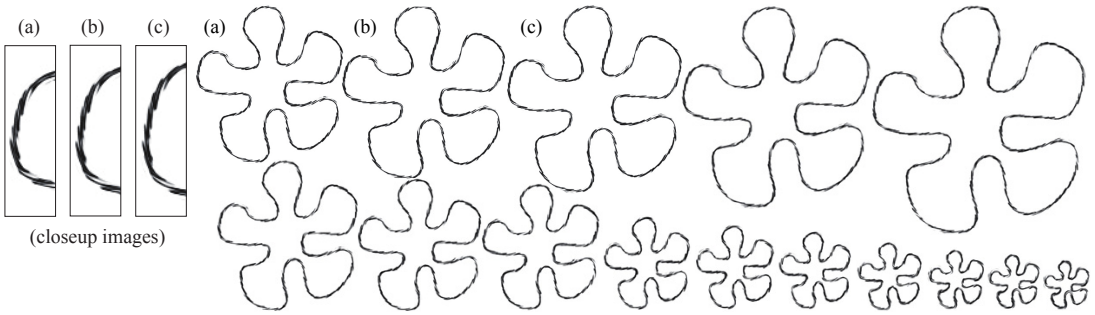


FIGURE D.6: Zoom sur un dessin de fleur vectoriel 2D. Notez que la taille de la texture à chaque image est bien préservée et qu'aucun artefact de mélange n'est visible.

Notre méthode étend les *artmaps* de Klein et al. [KLK*00] dont le nombre de textures requises est $O(\log_\sigma L_{max})$, où L_{max} est la longueur du plus long chemin et σ est le facteur d'étirement maximum autorisé. Malheureusement, L_{max} est généralement non contraint, en particulier durant le zoom. Notre nouvelle pyramide de texture est autosimilaire, ce qui permet à un nombre restreint de textures de couvrir continûment un chemin d'une longueur arbitraire.

Lorsque σ est grand, des discontinuités peuvent être visibles durant la transition entre deux niveaux de la pyramide. Mélanger les niveaux adjacents de l'artmap réduit ces discontinuités, mais tend à produire des résultats flous. La solution est de créer une pyramide dense, avec de nombreuses textures et un σ faible. Nous proposons une méthode de génération automatique de cette pyramide, en nous basant sur la synthèse de texture paramétrique de Portilla et Simoncelli [PS00]. Notre approche permet de générer une SLAM de densité arbitraire en se basant sur une unique texture d'exemple. Par ailleurs, notre synthèse garantit que les niveaux adjacents de la pyramide se mélangent continûment sans artefacts.

Une simple fonction de placage permet d'appliquer cette SLAM aux animations vectorielles 2D (Figure D.6) et aux lignes fixes à la surface d'objets 3D.

3.2. Paramétrisation de lignes par propagation robuste de votes

Appliquer une SLAM à un chemin complexe nécessite la définition d'une paramétrisation T. Le choix le plus simple pour T est l'abscisse curviligne du chemin s en espace image (i. e., sa paramétrisation naturelle). Cependant, pour les lignes dépendantes du point de vue (silhouettes, contours suggestifs, arrêtes apparentes, etc.), cette approche entraîne des glissements ou des discontinuités temporelles, car ces lignes ne sont pas fixes à la surface du modèle.

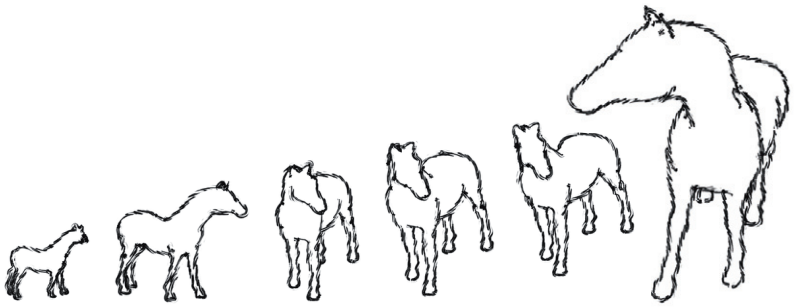


FIGURE D.7: Zoom et rotation autour d'un modèle 3D dont les silhouettes sont stylisées. Voir la vidéo accompagnant l'article [BCGF10] pour apprécier la cohérence de la stylisation.

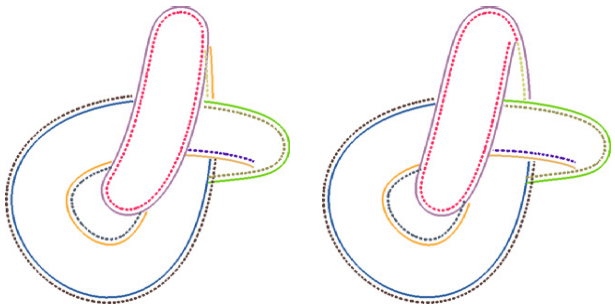


FIGURE D.8: Chaque coup de pinceau (pointillés) est défini relativement à un contour actif (solide), mais peut dévier substantiellement pour permettre un certain niveau d'abstraction. Lorsque le noeud celtique tourne, la topologie des contours évolue ; par exemple, le contour orange fusionne avec le contour violet. La cohérence de la paramétrisation est préservée en ne fusionnant pas les coups de pinceau beige et roses.

Pour résoudre ce problème, nous proposons un algorithme de paramétrisation en espace image qui détermine les paramètres (ρ, ϕ) de telle sorte que $T(s) = \rho s + \phi$ soit la plus proche de la paramétrisation de la même ligne à l'image précédente. Pour ce faire, nous propageons la paramétrisation de l'image f à l'image $f+1$ en enregistrant des votes dans un tampon 2D. Ensuite, nous mettons à jour les paramètres (ρ, ϕ) à l'aide d'une méthode d'ajustement de ligne robuste aux valeurs aberrantes (RANSAC [FB81]). Cette paramétrisation peut ensuite être utilisée en combinaison avec les SLAMs pour créer des dessins au trait animés à partir de modèles 3D simples et lisses (Figure D.7).

3.3. Contours actifs pour le suivi et la paramétrisation de lignes animées

Cette approche vise à réconcilier la simplicité et la cohérence naturelle des lignes extraites en espace image avec la richesse des styles disponibles pour les lignes extraites en espace objet, tout en fournissant une paramétrisation cohérente et robuste pour des modèles complexes.

L'entrée de notre méthode consiste en un ensemble d'échantillons indépendants pouvant provenir d'une technique d'extraction de ligne aussi bien en espace image qu'en espace objet. L'idée centrale à notre approche est d'utiliser les contours actifs – polylignes 2D suivant le modèle de Kass et al. [KWT88] – pour inférer la connectivité des échantillons et les suivre d'une image à l'autre en tenant compte le champ de mouvement de la scène.

La formulation originale des contours actifs leur permet de se déformer pour épouser la forme des échantillons. Le défi supplémentaire est de gérer correctement les changements topologiques des lignes durant l'animation. C'est pourquoi

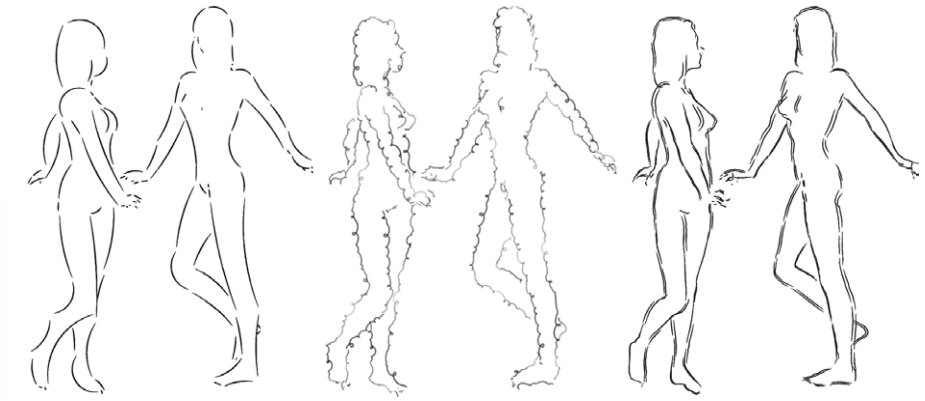


FIGURE D.9: Modèle dans deux poses et trois styles : arcs, boucles et esquisse.

nous proposons de nouveaux mécanismes pour ajouter, supprimer, contracter, diviser et fusionner les contours actifs.

D'autre part, nous proposons un mécanisme de propagation et de régularisation de la paramétrisation au cours du temps. En autorisant une ligne à être dessinée par plusieurs coups de pinceau successifs (Figure D.8), nous contrôlons les points de rupture de la paramétrisation. Un mécanisme de fusion permet d'éviter la multiplication de petits coups de pinceau au fil du temps. Afin d'autoriser une plus grande abstraction de la forme d'entrée, la position de ces coups de pinceau peut dévier significativement des contours actifs pour satisfaire d'autres contraintes de style sans compromettre la précision du suivi (Figure D.9).

4. Stylisation des régions

Les méthodes de rendu non-photoréaliste, qui visent à représenter les régions de couleur d'une scène 3D avec des médiums ou motifs 2D (tels que des pigments ou des coups de pinceau), sont confrontées au problème de la cohérence temporelle lorsqu'elles sont appliquées à des scènes dynamiques. Ce problème résulte de la difficulté à satisfaire au mieux les trois objectifs que sont le respect de l'aspect 2D, la cohérence du mouvement et la continuité temporelle.

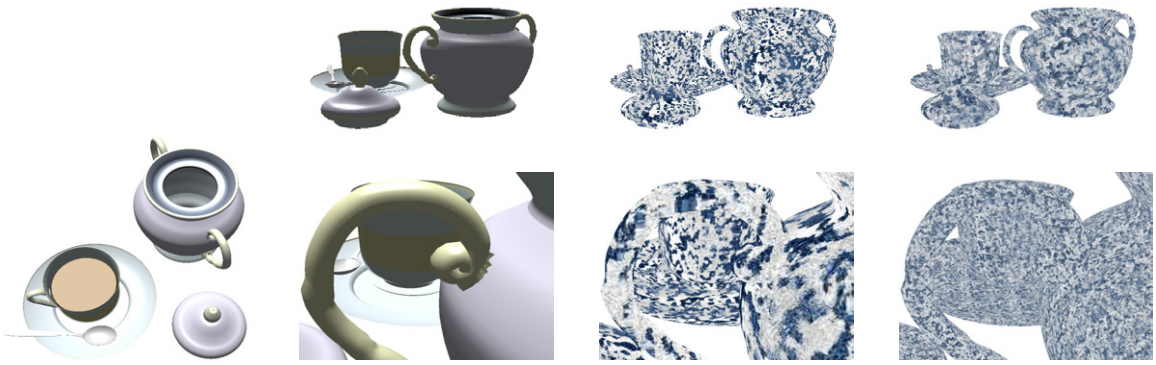
Dans cette section, nous décrivons deux méthodes faisant un compromis différent entre ces objectifs. Notre première solution favorise la cohérence du mouvement, tandis que la seconde préserve mieux l'aspect 2D. Toutes deux visent une utilisation interactive et permettent la création d'un grand nombre de motifs.

Notez que notre objectif n'est pas de reproduire un style spécifique, ce qui explique pourquoi nos résultats ne ressemblent pas parfaitement à leur équivalent réel (hachures manuelles, par exemple). Il s'agit plutôt de fournir une brique de base temporellement cohérente avec laquelle les artistes peuvent construire leur propre style sans avoir à se préoccuper de son comportement dynamique durant l'animation.

4.1. Textures volumiques autozoomables

Nous présentons les *textures autozoomables* qui facilitent l'intégration de stylisation temporellement cohérente dans un pipeline de rendu temps réel. Notre approche se base sur des textures comme support de la stylisation, afin d'éviter la mise en place de structures de données complexes. Cela rend notre approche particulièrement adaptée aux médiums présentant une texture caractéristique comme l'aquarelle, le fusain… De plus, la gestion optimisée des textures par les cartes graphiques modernes rend notre méthode bien adaptée au rendu temps réel.

Contrairement au texturage traditionnel, notre approche bénéficie d'un mécanisme de *zoom infini* produisant une distribution uniforme des éléments de texture à l'écran, quelle que soit la distance des objets à la caméra (Figure D.10). Ce mécanisme préserve la majorité des caractéristiques 2D du médium traditionnel tout en assurant la cohérence temporelle lors de la navigation dans un univers 3D. Ce



(a) Vue plongeante (b) Éclairage de Phong (c) Texture 3D (d) Texture auto-zoomable

FIGURE D.10: Comparaison de nos textures volumiques auto-zoomables avec les techniques traditionnelles à échelle fixe de texturage 2D et 3D. Notez la taille globalement constante des éléments de texture – quel que soit le facteur de zoom – et la correction de la déformation perspective qu'elle produit.

mécanisme de zoom infini peut être appliqué aux textures 2D comme 3D. Cependant, nous avons choisi de développer le cas des textures 3D, afin d'éviter le calcul complexe ou la définition manuelle d'une paramétrisation à la surface des objets 3D.

Afin de démontrer l'efficacité de cette approche, nous proposons son intégration au moteur de rendu OGRE. Les mesures de performance réalisées dans cet environnement indiquent un impact faible de la méthode sur la vitesse d'affichage pour des scènes complexes.

4.2. Bruit de Gabor NPR

Nous proposons une fonction de bruit qui fournit un compromis équilibré pour la stylisation temporellement cohérente de tout type d'animations 3D, y compris des modèles déformables. En mélangeant un grand nombre de ces primitives, nous produisons un bruit procédural possédant les avantages des approches à base de marques et des approches basées sur des textures.

Notre nouvelle fonction de bruit étend le bruit de Gabor [LLDD09, LLC*10]. Nous avons choisi ce bruit, car il autorise un contrôle local du spectre ce qui nous permet non seulement de préserver l'aspect 2D du bruit, mais aussi de créer une grande variété de motifs. En définissant ce bruit à la surface des objets 3D, mais en l'évaluant en espace image, nous assurons la cohérence du mouvement tout en

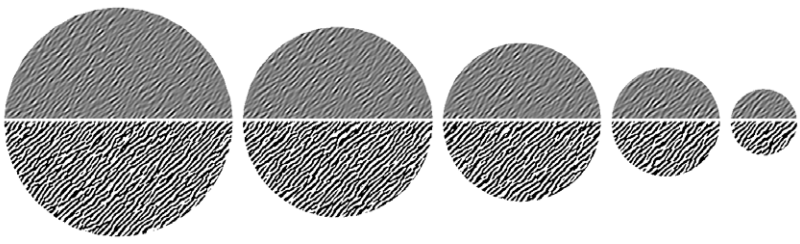


FIGURE D.11: Zoom sur une sphère texturée avec notre bruit de Gabor NPR avant (haut) et après seuillage binaire (bas). À chaque niveau de zoom, les propriétés statistiques du bruit sont bien préservées grâce à notre mécanisme de niveau de détail.

préservant l'apparence 2D du motif. En outre, nous proposons un nouveau mécanisme de niveau de détail qui assure une forte continuité temporelle (Figure D.11).

Pour garantir une visualisation interactive de la stylisation, nous avons implémenté notre bruit de Gabor NPR sur carte graphique. Celui-ci peut ensuite être utilisé, de la même façon que les textures procédurales standards, pour créer des styles continus (aquarelle, encre) comme discrets (hachures, peinture).

4.3. Styles

Nos textures volumiques autozoomables comme notre bruit de Gabor NPR peuvent être directement utilisés par de nombreuses méthodes de rendu non-photoréaliste en substituant les textures standards par notre version dynamique. Nous illustrons ce principe pour une variété de styles binaires et colorés. Certains d'entre eux trouvent leur inspiration dans des méthodes existantes (e. g., [DOM*01, BKTS06]), tandis que d'autres sont inédites, profitant de la grande variété de motifs que nous pouvons produire.

Les textures volumiques peuvent être générées, soit procéduralement en utilisant un bruit de Perlin [Per85, Ola05], soit par synthèse à partir d'exemples 2D [KFCO*07].

À partir de notre bruit de Gabor NPR, nous utilisons des techniques standards en modélisation et synthèse de texture procédurale (e. g., [EMP*02]). Voir l'Appendice B pour la description complète des « shaders » de styles, paramètres de bruit et fonctions de mélange que nous avons utilisés.

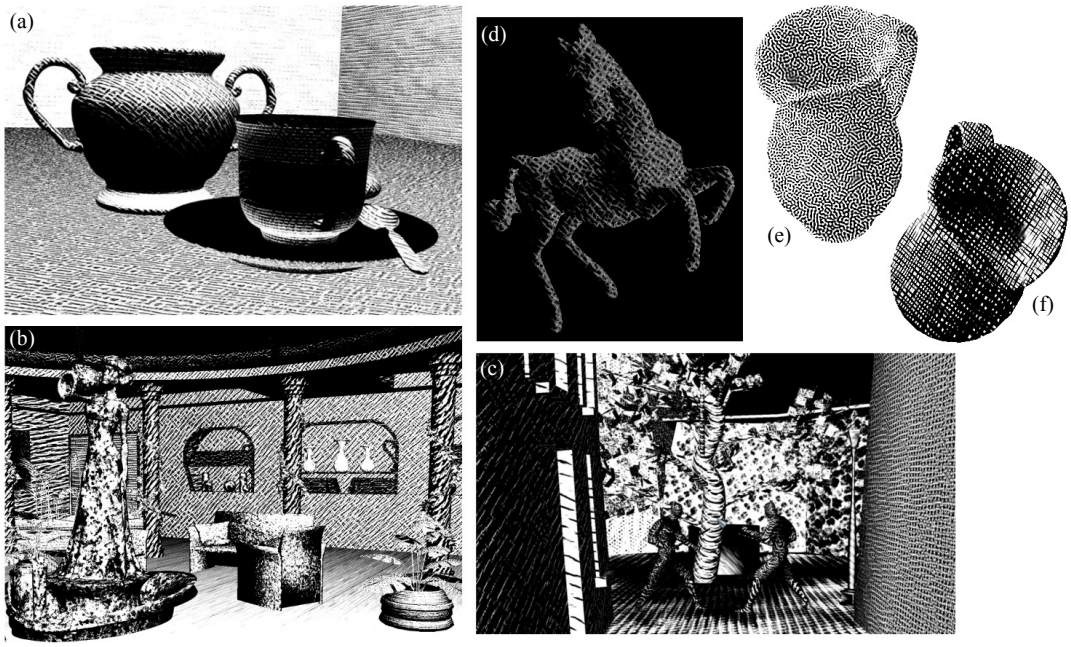


FIGURE D.12: Styles binaires. (a-c) Scènes stylisées avec nos textures volumiques auto-zoomables ; (d-f) motifs produits avec notre bruit de Gabor NPR.

5. Évaluation

L'évaluation est une question récurrente en rendu non-photoréaliste. Elle est particulièrement difficile lorsqu'il s'agit d'animations, car il n'existe généralement pas de solution de référence. Initialement basé sur une évaluation visuelle purement spéculative, des progrès ont été faits en informatique graphique en général, et en rendu non-photoréaliste en particulier, pour développer des méthodologies d'évaluation objectives en accord avec la perception.

En nous concentrant sur le problème de la cohérence temporelle lors de la stylisation des régions, nous proposons dans cette section deux études utilisateur contribuant à cette tendance. Nous explorons tout d'abord l'effet de la fractalisation sur différentes textures de médium, et nous dérivons une mesure de qualité objective depuis les jugements des utilisateurs.

Durant la seconde étude, nous comparons six méthodes – incluant nos textures volumiques autozoomables et notre bruit de Gabor NPR – selon les trois objectifs définissant le problème de la cohérence temporelle. Les résultats de cette étude tendent à montrer que la cohérence du mouvement serait l'objectif fondamental à préserver dans le compromis global.

5.1. Évaluation perceptuelle en graphique

Les progrès réalisés en informatique graphique sont généralement démontrés à travers des images ou des vidéos. L'inspection visuelle est par conséquent d'une importance capitale dans l'évaluation de ces techniques. Dans le domaine du traitement d'image ou du rendu photoréaliste, des solutions de référence existent souvent et peuvent servir d'étalon pour cette évaluation. Par exemple, une méthode de rendu temps réel approximative peut être comparée à une méthode de rendu hors-ligne plus précise pour estimer à quel point les résultats que ces deux approches produisent sont similaires. De nombreuses approches ont été proposées dans ces communautés pour évaluer la fidélité d'une image ou d'une vidéo vis-à-vis d'une référence.

Cependant, la plupart du temps, nous ne disposons pas d'une telle référence en rendu non-photoréaliste, en particulier pour les animations. Néanmoins, certaines méthodes d'évaluation ont été développées pour aller au-delà d'un jugement visuel purement spéculatif. Parmi elles, les méthodes psychophysiques sont bien établies et ont été utilisées dans de nombreuses disciplines. Elles permettent d'effectuer des mesures objectives de nos expériences perceptuelles subjectives. Les outils développés en psychophysique semblent parfaitement adaptés à nos besoins, et nous utilisons certains d'entre eux durant nos études utilisateurs.

5.2. Évaluation des textures fractalisées

La *fractalisation de texture* modifie le motif de la texture en introduisant de nouveaux éléments ou de nouvelles fréquences et en modifiant son contraste. Par conséquent, le résultat de la fractalisation est visuellement différent du motif originalement choisi par l'artiste. Nous estimons que l'évaluation automatique de cette

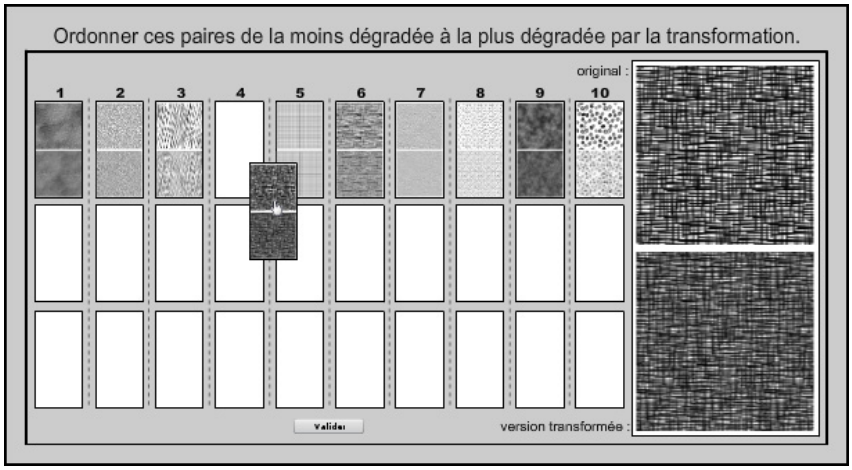


FIGURE D.13: Interface web pour réaliser le tri des deux séries de dix paires de textures. Jusqu'à trois paires peuvent recevoir le même rang (colonne).

perte de similarité pourrait être un outil intéressant pour comparer les méthodes de mélange de texture existantes, mais aussi pour favoriser le développement de nouvelles méthodes de fractalisation.

Dans ce contexte, nous définissons la *déformation* de la texture comme la dissimilarité entre la texture 2D originale et sa version transformée. Notre objectif est de définir une mesure quantitative de cette déformation. Pour cela, nous réalisons une étude utilisateur demandant aux participants de trier des pairs de textures originales/transformées de la moins déformée à la plus déformée (Figure D.13). Nous fournissons une analyse statistique de ces résultats, et nous en dérivons une échelle perceptuellement linéaire de la déformation pour dix classes de médium. Aucune information quant à la nature ou l'origine de la déformation n'est donnée aux participants. Au contraire, nous leur demandons d'identifier les critères qu'ils ont utilisés pour évaluer cette déformation.

Dans un second temps, nous étudions la corrélation des jugements subjectifs avec plusieurs mesures de qualité objectives utilisées en analyse de texture ou mesure de qualité d'images. Nous montrons que l'erreur entre les matrices de co-occurence des pairs de textures est un bon indice pour mesurer cette déformation.

5.3. Évaluation des méthodes de stylisation des régions

Nous présentons une étude utilisateur évaluant le succès de six méthodes pour styliser les régions de couleur, incluant nos deux nouvelles solutions.

Les objectifs de respect de l'aspect 2D, de cohérence du mouvement et de continuité temporelle sont contradictoires. Par conséquent, il n'existe pas de solution parfaite et évaluer ces différents compromis est difficile. Néanmoins, chaque objectif est lié à des artefacts spécifiques qu'un utilisateur peut observer. Une étude utilisateur peut donc apporter d'intéressantes informations quant à aux performances de ces méthodes pour chaque objectif. Elle peut en outre donner une indication quant aux préférences des utilisateurs et quant à l'importance relative des trois objectifs.

FIGURE D.14: Participante réalisant notre étude utilisateur.

Les résultats de notre étude montrent que la cohérence du mouvement est perçue comme le critère le plus important à préserver. Ils montrent également que

nos deux méthodes – les textures volumiques autozoomables et le bruit de Gabor NPR –, bien que proposant un compromis différent, se comportent bien pour les objectifs qu'ils favorisent.

6. Conclusion

Ce manuscrit défend que les textures constituent la brique de base la plus appropriée pour styliser des animations 3D de façon temporellement cohérente. Non seulement les lignes et les régions sont largement enrichies par les médiums et motifs portés par la texture, mais encore le comportement dynamique de la stylisation est amélioré, car la texture permet des transitions ou modifications spatiales et temporelles douces et continues.

6.1. Résumé des contributions

La première contribution de cette thèse est la *formalisation* du problème de la cohérence temporelle et son *évaluation* dans le cas des régions de couleur. En décomposant ce problème en trois objectifs – *aspect 2D, cohérence du mouvement* et *continuité temporelle* – les travaux précédents ainsi que les solutions que j'ai proposées avec mes collaborateurs peuvent être comparées à l'aide de critères bien définis. De plus, cette décomposition permet une évaluation expérimentale plus précise que l'inspection visuelle et s'affranchit des considérations esthétiques.

La seconde contribution consiste en deux *méthodes de paramétrisation* pour les lignes dépendantes du point de vue. La première, simple à implémenter, est appropriée pour les modèles 3D lisses et simples. La seconde approche supporte les modèles complexes ainsi que les lignes extraites en espace image en séparant l'étape de suivi des lignes de l'étape de stylisation.

La dernière contribution de cette thèse consiste en trois solutions pour gérer les grandes disparités et variations de profondeur au sein des scènes 3D.

- La première approche se concentre sur les textures de coup de pinceau pour les lignes. Elle utilise la synthèse de texture paramétrique pour générer automatiquement des pyramides de texture *autosimilaires* permettant de zoommer infiniment sur des lignes texturées.

FIGURE D.15: Modèle de singe dont les régions ont été stylisées avec notre bruit de Gabor NPR, et les lignes avec nos contours actifs.

- La seconde solution *fractalise* des textures volumiques pour assurer une taille quasi constante du motif en espace écran à n'importe quel niveau de zoom.
- La dernière méthode utilise un bruit procédural, et propose un nouveau mécanisme de niveau de détail qui respecte parfaitement les caractéristiques 2D du motif sans introduire d'artefact temporel.

La combinaison de ces deux dernières contributions permet aux artistes de créer une grande variété de styles (Figure D.15). Les résultats ainsi produits – en particulier au cours d'une animation – démontrent la validité de la thèse supportée par ce travail.

6.2. Perspectives

Même si les méthodes présentées dans ce manuscrit ont déjà de potentielles applications pratiques, j'espère qu'elles vont aussi ouvrir la voie à d'autres recherches. Outre les travaux futurs directement liés à mes contributions, j'entrevoie quatre directions principales d'améliorer et d'extension.

Stylisation par optimisation. Durant ma thèse, je me suis efforcé à explorer autant que possible le triangle délimité par les trois objectifs définissant le problème de la cohérence temporelle. Néanmoins, toutes les méthodes proposées à ce jour – y compris les miennes – font un compromis fixe dans ce triangle. Par conséquent, le champ des compromis est restreint à un ensemble de techniques relativement éparses.

Il serait utile de pouvoir se déplacer continûment dans cet espace des compromis. Tout d'abord, cela permettrait aux artistes un contrôle bien plus précis de la cohérence temporelle. Ensuite, cela rendrait possible la création de transitions douces entre différents compromis au sein d'une même image ou bien durant une animation. Par exemple, l'artiste pourrait choisir de privilégier la cohérence du mouvement pour les objets au premier plan et l'aspect 2D pour l'arrière-plan.

Pour autoriser de tels effets, il est nécessaire de concevoir un système versatile offrant un contrôle direct et continu sur les trois axes du compromis. Cela implique la mesure automatique, objective et quantitative de la réalisation de chaque objectif. Ensuite, en se basant sur ces mesures, le système doit être capable d'*optimiser* localement (dans le temps et l'espace) le processus de stylisation de façon à se conformer à l'intention de l'artiste. Cela pose plusieurs questions : Quelles sont les bonnes – c'est-à-dire perceptuellement valides – mesures ? Comment exprimer la stylisation en un problème général d'optimisation ? Quelle est la meilleure représentation pour le résoudre ?

De la même façon que les modèles d'attention visuelle [Itt00, RTAK07] et les plateformes d'évaluation de la qualité d'images [FP04, AvMS10], ce système suit une approche ascendante, c'est à dire que des mesures de bas niveau sont intégrées pour construire une mesure complexe. L'étape d'intégration pose une difficulté supplémentaire : Comment combiner les trois mesures de façon perceptuellement valide ? Une simple combinaison linéaire ne semble pas être la bonne approche. Non seulement les trois objectifs ne sont pas perçus de façon équivalente, mais encore certains effets temporels peuvent changer la perception d'effets spatiaux (le mouvement et l'apparence 2D, par exemple). De plus, d'autres phénomènes

perceptuels, comme la *cécité au changement* [SR05], indiquent que l'attention visuelle est nécessaire pour percevoir les modifications. Si ces effets compliquent l'intégration des mesures, ils montrent également qu'il pourrait être possible de les exploiter pour tromper la perception humaine lors d'inévitables changements.

Les méthodes basées sur le mélange de nombreuses marques semblent prometteuses pour implémenter ce système versatile. En effet, elles autorisent un contrôle local – comme les approches utilisant peu de marques – et conservent les avantages des méthodes basées sur des textures (i. e., leur continuité temporelle et spatiale). Par exemple, il semble possible de créer une interpolation continue entre un bruit de Gabor évalué à la surface du modèle 3D et notre version NPR, en inclinant progressivement les noyaux de Gabor du plan tangent à la surface vers le plan de la caméra.

Évaluation perceptuelle. Le système versatile décrit précédemment pourrait permettre non seulement d'améliorer la stylisation, mais aussi l'évaluation perceptuelle. Avec un tel système, nous pourrions échantillonner de façon bien plus dense l'espace des solutions, en produisant ainsi un grand nombre de stimuli expérimentaux. Nous pourrions alors construire une étude utilisateur dans l'esprit de Willis et al. [WAKB09] pour évaluer la pertinence de notre décomposition en trois objectifs.

Willis et al. [WAKB09] construisent un plongement spatial de la brillance d'un matériau à partir de comparaisons pair-à-pair d'images. Ces images représentent un même objet avec une illumination constante, mais différentes BRDF. Un triplet de ces images est montré aux participants de l'étude qui doivent indiquer les deux d'entre elles qui sont les plus similaires. En utilisant un nouvel algorithme de « multidimensional scaling », ils déduisent de ces comparaisons un plongement 2D (sans à priori sur sa dimension), et étudient sa corrélation avec différentes mesures formelles de la brillance.

De la même façon, il pourrait être possible de créer un plongement du problème de la cohérence temporelle en utilisant des comparaisons pair-à-pair d'animations stylisées. La dimension de ce plongement indiquerait si notre décomposition en trois axes est perceptuellement valide. La position des stimuli le long de ces axes permettrait la validation des mesures objectives utilisées pour produire les séquences stylisées.

Contrôle artistique. Les méthodes décrites dans cette dissertation visent toutes des applications interactives. Celles-ci ont l'avantage de donner un retour instantané à l'artiste lors d'un changement de point de vue ou de certains paramètres. Cependant, elles peuvent empêcher un contrôle fin et des corrections locales durant l'animation, ce qui est généralement nécessaire lors de productions de grande qualité. À l'inverse, dans le cas de pipeline hors-ligne, l'animation complète (mouvements des objets et de la caméra) est connue à l'avance, et il est possible d'utiliser cette information pour permettre de telles corrections.

Nos contours actifs semblent déjà une représentation adéquate pour obtenir de telles fonctionnalités. Ils persistent en 2D et sont partiellement indépendants de la géométrie 3D sous-jacente. Par conséquent, un artiste pourrait corriger le comportement automatique de nos contours actifs en fixant des contraintes fortes (de position, topologie, présence) à certaines images clés de l'animation. De telles outils sont fournis par le système de rotoscopie assistée par ordinateur de Agrawala

et al. [AHSS04], à ceci près que leurs contours ne peuvent pas changer de topologie. Le défi principal est ensuite de propager ces contraintes dans le temps (dans le passé comme le futur) tout en préservant la localité de ces modifications. Ni notre approche gloutonne, ni une optimisation complètement globale [AHSS04] ne me semble approprié pour assurer ces deux propriétés, ce qui implique de trouver une solution intermédiumsire.

Le problème est encore plus complexe pour les régions de couleur, car il n'est pas aussi simple de délimiter leur contour. Les techniques de segmentation en espace image appliquées à des « deep buffers » pourraient constituer une bonne représentation intermédiumsire entre la géométrie 3D et l'image finale stylisée.Contrôler le champ de mouvement (e. g., le simplifier ou l'éditer) et les paramètres de style à l'intérieur de chaque région serait alors possible. A nouveau, cela pose la question de la propagation de ces modifications dans le temps.

Outre le contrôle du comportement temporel de la stylisation, la définition du style par les artistes – c'est à dire la fonction de transfert depuis les primitives de la scène vers les attributs des marques – requière de plus amples travaux. Bien que cela n'ait pas été au centre de cette thèse, les techniques de stylisation que nous avons proposées utilisent les trois approches principales pour définir ce transfert : le contrôle direct des paramètres (via des curseurs, des boutons...) la programmation de scripts et « shaders », et une approche par l'exemple. Chacune d'entre elles ont matière à amélioration :

- Des interfaces plus intuitives et des techniques d'interaction directe, comme le « sketching », pourraient être développées pour créer des textures autosimilaires ou guider la synthèse de texture.
- La création de motifs procéduraux avec notre bruit de Gabor NPR bénéficierait de l'estimation automatique des paramètres du bruit de telle sorte qu'ils correspondent à ceux d'une texture d'exemple [LVLD10, GD10, JCW11].

Le défi supplémentaire serait de combiner ces différentes approches de définition du style. Certaines techniques de création de texture réalistes – comme les « smart textures » d'Allegorithmic© qui mélangent des images matricielles et vectorielles avec des éléments procéduraux – semblent une source d'inspiration intéressante.

Animation 2D. A l'exception des SLAMs, cette thèse s'intéresse uniquement aux animations 3D et s'efforce à donner une apparence 2D au images stylisées ainsi produites. Cette approche à certes d'indéniables avantages, mais il semble raisonnable de considérer d'autres entrées, en particulier des animations vectorielles 2D. Les logiciels actuels (e. g., Flash©, ToonBoom©) fournissent peu d'outils pour enrichir de telles animations avec des médiums ou des motifs. Au mieux, ils permettent de plaquer une texture 2D rigide à l'intérieur d'une forme fermée, mais cette texture n'est pas affectée par la déformation que peut subir la forme au cours de l'animation, ce qui produit des glissements, voire l'effet « rideau de douche ».

Deux récentes méthodes basées sur les courbes de diffusion [OBW*08] proposent des techniques plus complexes pour ajouter une texture à des formes vectorielles 2D statiques. Jeschke et al. [JCW11] utilisent un bruit de Gabor [LLD10], défini le long des courbes, qui varie spatialement pour générer des motifs stochastiques ou irréguliers. Bien que non démontré dans cet article, il est certainement possible d'animer ce bruit par images clés, ce qui devrait produire des résultats

relativement similaires à ceux obtenus avec notre bruit de Gabor NPR. Winnemöller et al. [WOBT09] proposent des outils pour créer et manipuler intuitivement des motifs vectoriels réguliers. Adapter cette méthode pour l'animation n'est pas évident dans la mesure où l'utilisateur doit spécifier le placage du motif à chaque image.

Étendre ces méthodes à une plus grande classe de motifs, voire à n'importe quelle texture matricielle, pose plusieurs défis : Comment lier la texture à la forme ? Comme modifier cette texture lorsque la forme se déforme ? Comment mesurer cette déformation – en tenant compte de la perception – et régénérer le motif pour éviter de trop grandes distorsions ?

Si certains travaux précédents, comme l'advection de texture, peuvent certainement être réutilisés dans ce contexte, la nature vectorielle de l'entrée devrait permettre de définir des solutions spécifiques. De récentes méthodes de déformation et d'interpolation [HF06, WBCG09, BBA09, WG10] produisent des résultats prometteurs dans le cas de dessins. Il me semble qu'elles pourraient être utilisées pour transférer la déformation de la forme à la texture.

BIBLIOGRAPHY

[Aga02] AGARWALA A.: SnakeToonz. In *Proceedings of the second international symposium on Non-photorealistic animation and rendering - NPAR '02* (New York, New York, USA, 2002), ACM Press, p. 139. 10.1145/508530.508554.

[AHSS04] AGARWALA A., HERTZMANN A., SALESIN D., SEITZ S. M.: Keyframe-based tracking for rotoscoping and animation. In *ACM SIGGRAPH 2004 Papers on - SIGGRAPH '04* (New York, New York, USA, 2004), ACM Press, p. 584. 10.1145/1186562.1015764.

[AMMS08] AYDIN T. O., MANTIUK R., MYSZKOWSKI K., SEIDEL H.-P.: Dynamic range independent image quality assessment. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 1. 10.1145/1360612.1360668.

[AS01] AGRAWALA M., STOLTE C.: Rendering effective route maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01* (New York, New York, USA, 2001), ACM Press, pp. 241–249. 10.1145/383259.383286.

[AvMS10] AYDIN T. O., ČADÍK M., MYSZKOWSKI K., SEIDEL H.-P.: Video quality assessment for computer graphics applications. *ACM Transactions on Graphics 29*, 6 (Dec. 2010), 1. 10.1145/1882261.1866187.

[Bar06] BARLA P.: *Representation and acquisition models for expressive rendering*. PhD thesis, INP Grenoble, 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 - France, Nov. 2006.

[BB95] BEAUCHEMIN S. S., BARRON J. L.: The computation of optical flow. *ACM Computing Surveys 27*, 3 (Sept. 1995), 433–466. 10.1145/212094.212141.

[BBA09] BAXTER W., BARLA P., ANJYO K.: N-way morphing for 2D animation. *Computer Animation and Virtual Worlds 20*, 2-3 (June 2009), 79–87. 10.1002/cav.310.

[BBT09] BÉNARD P., BOUSSEAU A., THOLLOT J.: Dynamic solid textures for real-time coherent stylization. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games - I3D '09* (New York, New York, USA, 2009), ACM Press, p. 121. 10.1145/1507149.1507169.

[BCGF10] BÉNARD P., COLE F., GOLOVINSKIY A., FINKELSTEIN A.: Self-similar texture for coherent line stylization. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering - NPAR '10* (New York, New York, USA, 2010), ACM Press, p. 91. 10.1145/1809939.1809950.

[BEDT08] BEZERRA H., EISEMANN E., DÉCORET X., THOLLOT J.: 3d dynamic grouping for guided stylization. In *NPAR '08: Proceedings of the 6th International Symposium on Non-photorealistic Animation and Rendering* (2008), ACM, pp. 89–95.

[BFP*11] BUCHHOLZ B., FARAJ N., PARIS S., EISEMANN E., BOUBEKEUR T.: Spatio-temporal analysis for parameterizing animated lines. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR) 2011* (2011).

[BG93] BALBOA R. M., GRZYWACZ N. M.: Power spectra and distribution of contrasts of natural images from different habitats. *Vision Research 43*, 24 (1993), 2527–2537.

[BKTS06] BOUSSEAU A., KAPLAN M., THOLLOT J., SILLION F. X.: Interactive watercolor rendering with temporal coherence and abstraction. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering - NPAR '06* (New York, New York, USA, 2006), ACM Press, p. 141. 10.1145/1124728.1124751.

[BLV*10a] BÉNARD P., LAGAE A., VANGORP P., LEFEBVRE S., DRETTAKIS G., THOLLOT J.: A Dynamic Noise Primitive for Coherent Stylization. *Computer Graphics Forum 29*, 4 (Aug. 2010), 1497–1506. 10.1111/j.1467-8659.2010.01747.x.

[BLV*10b] BÉNARD P., LAGAE A., VANGORP P., LEFEBVRE S., DRETTAKIS G., THOLLOT J.: NPR Gabor noise for coherent stylization. In *ACM SIGGRAPH 2010 Talks on - SIGGRAPH '10* (New York, New York, USA, 2010), ACM Press, p. 1. 10.1145/1837026.1837079.

[BNTS07] BOUSSEAU A., NEYRET F., THOLLOT J., SALESIN D.: Video watercolorization using bidirectional texture advection. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007), 104. http://doi.acm.org/10.1145/1275808.1276507.

[Bou98] BOURDEV L.: *Rendering Nonphotorealistic Strokes with Temporal and Arc-Length Coherence*. Master's thesis, Brown University, Providence, RI, 1998.

[BSM*07] BRESLAV S., SZERSZEN K., MARKOSIAN L., BARLA P., THOLLOT J.: Dynamic 2D patterns for shading 3D scenes. *ACM Transactions on Graphics 26*, 3 (July 2007), 20. 10.1145/1276377.1276402.

[BTM06] BARLA P., THOLLOT J., MARKOSIAN L.: X-Toon: An extended toon shader. In *NPAR'2006: international symposium on non-photorealistic animation and rendering* (2006), ACM.

[BTS05] BARLA P., THOLLOT J., SILLION F.: Geometric Clustering for Line Drawing Simplification. In *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering2005)* (2005).

[BZOP07] BOUVIER-ZAPPA S., OSTROMOUKHOV V., POULIN P.: Motion cues for illustration of skeletal motion capture data. In *Proceedings of NPAR 2007* (2007), pp. 133–140. 10.1145/1274871.1274891.

[CDF*06] COLE F., DECARLO D., FINKELSTEIN A., KIN K., MORLEY K., SANTELLA A.: Directing Gaze in 3D Models with Stylized Focus. In *Eurographics Workshop on Rendering* (June 2006).

[CDH06] COCONU L., DEUSSEN O., HEGE H.-C.: Real-time pen-and-ink illustration of landscapes. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering - NPAR '06* (New York, New York, USA, 2006), ACM Press, p. 27. 10.1145/1124728.1124734.

[CDHR08] CHEN Y., DAVIS T. A., HAGER W. W., RAJAMANICKAM S.: Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw.*, 3 (2008).

[CF08] COLE F., FINKELSTEIN A.: Partial visibility for stylized lines. In *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering - NPAR '08* (New York, New York, USA, 2008), ACM Press, p. 9. 10.1145/1377980.1377985.

[CF10] COLE F., FINKELSTEIN A.: Two fast methods for high-quality line visibility. *IEEE transactions on visualization and computer graphics 16*, 5 (2010), 707–17. 10.1109/TVCG.2009.102.

[CL06] CHI M.-T., LEE T.-Y.: Stylized and abstract painterly rendering system using a multiscale segmented sphere hierarchy. *IEEE transactions on visualization and computer graphics 12*, 1 (2006), 61–72. 10.1109/TVCG.2006.14.

[CRH05] COLLOMOSSE J. P., ROWNTREE D., HALL P. M.: Stroke surfaces: temporally coherent artistic animations from video. *IEEE transactions on visualization and computer graphics 11*, 5 (2005), 540–9. 10.1109/TVCG.2005.85.

[CRL01] CORNISH D., ROWAN A., LUEBKE D.: View-dependent particles for interactive non-photorealistic rendering. In *Proceedings of Graphics Interface 2001* (Toronto, Ont., Canada, Canada, 2001), Canadian Information Processing Society, pp. 151–158.

[CRT01] COPELAND A. C., RAVICHANDRAN G., TRIVEDI M. M.: Texture synthesis using gray-level co-occurrence models: algorithms, experimental analysis, and psychophysical support. *Optical Engineering 40*, 11 (2001), 2655. 10.1117/1.1412851.

[CTP*03] CUNZI M., THOLLOT J., PARIS S., DEBUNNE G., GASCUEL J., DURAND F.: Dynamic canvas for non-photorealistic walkthroughs. In *Graphics Interface 2003* (2003), Citeseer.

[Dal92] DALY S. J.: Visible differences predictor: an algorithm for the assessment of image fidelity. *Proceedings of SPIE 1666*, 1 (1992), 2–15. 10.1117/12.135952.

[Dan99] DANIELS E.: Deep canvas in Disney's Tarzan. In *ACM SIGGRAPH 99 Conference abstracts and applications on - SIGGRAPH '99* (New York, New York, USA, 1999), ACM Press, p. 200. 10.1145/311625.312010.

[DCFR07] DECORO C., COLE F., FINKELSTEIN A., RUSINKIEWICZ S.: Stylized shadows. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering* (2007), ACM, pp. 77–83. 10.1145/1274871.1274884.

[DFR04] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S.: *Interactive rendering of suggestive contours with temporal coherence*. ACM Press, New York, New York, USA, 2004.

[DM00] DELINGETTE H., MONTAGNAT J.: New Algorithms for Controlling Active Contours Shape and Topology. In *European Conference on Computer Vision (ECCV'2000), number 1843 in LNCS* (2000), Springer, pp. 381–395.

[DOM*01] DURAND F., OSTROMOUKHOV V., MILLER M., DURANLEAU F., DORSEY J.: Decoupling Strokes and High-Level Attributes for Interactive Traditional Drawing. In *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering2001)* (London, UK, 2001), Springer-Verlag, pp. 71–82.

[DPF03] DUMONT R., PELLACINI F., FERWERDA J.: Perceptually-driven decision theory for interactive realistic rendering. *ACM Transactions on Graphics 22*, 2 (2003), 152–181.

[Dur02] DURAND F.: *An invitation to discuss computer depiction*. ACM Press, New York, New York, USA, 2002.

[EC90] ELBER G., COHEN E.: Hidden curve removal for free form surfaces. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), SIGGRAPH '90, ACM, pp. 95–104. doi.acm.org/10.1145/97879.97890.

[EMP*02] EBERT D. S., MUSGRAVE F. K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modeling: A Procedural Approach*, 3rd ed. Morgan Kaufmann Publishers, Inc., 2002.

[FA91] FREEMAN W., ADELSON E.: The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 9 (1991), 891–906. 10.1109/34.93808.

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (1981), 381–395.

[Fer08] FERWERDA J. A.: Psychophysics 101. In *ACM SIGGRAPH 2008 classes on - SIGGRAPH '08* (New York, New York, USA, 2008), ACM Press, p. 1. 10.1145/1401132.1401243.

[FMS01] FREUDENBERG B., MASUCH M., STROTHOTTE T.: Walk-Through Illustrations: Frame-Coherent Pen-and-Ink Style in a Game Engine. *Computer Graphics Forum 20*, 3 (2001), 184–191.

[FP04] FARRUGIA J.-P., PÉROCHE B.: A progressive rendering algorithm using an adaptive perceptually based image metric. *Computer Graphics Forum 23* (2004), 605–614.

[FTA04] FLEMING R. W., TORRALBA A., ADELSON E. H.: Specular reflections and the perception of shape. *Journal of Vision 4*, 9 (2004), 798–820. 10.1167/4.9.10.

[FVA03] FUNG J., VERYOVKA O., ARTS E.: Pen-and-ink textures for real-time rendering. In *Proceedings of Graphics Interface* (2003), pp. 131–138.

[FW07] FRISVAD J. R., WYVILL G.: Fast high-quality noise. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia - GRAPHITE '07* (New York, New York, USA, 2007), ACM Press, p. 243. 10.1145/1321261.1321305.

[GD10] GILET G., DISCHLER J.-M.: An Image-Based Approach for Stochastic Volumetric and Procedural Details. *Computer Graphics Forum 29*, 4 (Aug. 2010), 1411–1419. 10.1111/j.1467-8659.2010.01738.x.

[GDS04] GRABLI S., DURAND F., SILLION F.: Density measure for line-drawing simplification. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.* (Washington, DC, USA, 2004), IEEE, pp. 309–315. 10.1109/PCCGA.2004.1348362.

[GG01] GOOCH B., GOOCH A.: *Non-Photorealistic Rendering*. AK Peters Ltd, 2001.

[Gla99] GLASSNER A.: *Andrew Glassner's notebook: recreational computer graphics*. No. April. Morgan Kaufmann Pub, 1999.

[GRG04] GOOCH B., REINHARD E., GOOCH A.: Human facial illustrations: Creation and psychophysical evaluation. *ACM Transactions on Graphics 23*, 1 (2004), 27–44.

[GSG*99] GOOCH B., SLOAN P.-P. J., GOOCH A., SHIRLEY P., RIESENFELD R.: Interactive technical illustration. In *Proceedings of the 1999 symposium on Interactive 3D graphics - SI3D '99* (New York, New York, USA, 1999), ACM Press, pp. 31–38. 10.1145/300523.300526.

[GTDS10] GRABLI S., TURQUIN E., DURAND F., SILLION F. X.: Programmable rendering of line drawing from 3D scenes. *ACM Transactions on Graphics 29*, 2 (Mar. 2010), 1–20. 10.1145/1731047.1731056.

[Gui54] GUILFORD J. P.: *Psychometric methods*. McGraw-Hill, New York, 1954.

[GVH07] GOODWIN T., VOLLICK I., HERTZMANN A.: Isophote distance: a shading approach to artistic stroke thickness. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering* (2007), ACM. 10.1145/1274871.1274880.

[GVWD06] GRUNDLAND M., VOHRA R., WILLIAMS G. P., DODGSON N. A.: Cross Dissolve Without Cross Fade: Preserving Contrast, Color and Salience in Image Compositing. In *Computer Graphics Forum* (Sept. 2006), vol. 25, pp. 577–586. 10.1111/j.1467-8659.2006.00977.x.

[GW02] GOOCH A. A., WILLEMSEN P.: Evaluating space perception in NPR immersive environments. In *Proceedings of the second international symposium on Non-photorealistic animation and rendering - NPAR '02* (New York, New York, USA, 2002), ACM Press, p. 105. 10.1145/508530.508549.

[HE04] HAYS J., ESSA I.: Image and video based painterly animation. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering - NPAR '04* (New York, New York, USA, 2004), ACM Press, p. 113. 10.1145/987657.987676.

[Her01] HERTZMANN A.: Paint by relaxation. In *Computer Graphics International 2001* (July 2001), pp. 47–54.

[Her02] HERTZMANN A.: Fast paint texture. In *Proceedings of the second international symposium on Non-photorealistic animation and rendering - NPAR '02* (New York, New York, USA, 2002), ACM Press, p. 91. 10.1145/508530.508546.

[Her03] HERTZMANN A.: A survey of stroke-based rendering. *IEEE Computer Graphics and Applications 23*, 4 (July 2003), 70–81. 10.1109/MCG.2003.1210867.

[HF06] HORMANN K., FLOATER M. S.: Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics 25*, 4 (2006), 1424–1441. http://doi.acm.org/10.1145/1183287.1183295.

[HJO*01] HERTZMANN A., JACOBS C. E., OLIVER N., CURLESS B., SALESIN D. H.: *Image analogies*. ACM Press, New York, New York, USA, 2001.

[Hor05] HORN D.: Stream Reduction Operations for {GPGPU} Applications. In *GPU Gems 2*, Pharr M., (Ed.). Addison-Wesley Professional, 2005, ch. 36, pp. 573–589.

[HP00] HERTZMANN A., PERLIN K.: Painterly rendering for video and interaction. In *Proceedings of the first international symposium on Non-photorealistic animation and rendering - NPAR '00* (New York, New York, USA, 2000), ACM Press, pp. 7–12. 10.1145/340916.340917.

[HRRG08] HAN C., RISSER E., RAMAMOORTHI R., GRINSPUN E.: Multiscale texture synthesis. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 1. 10.1145/1360612.1360650.

[HZ00] HERTZMANN A., ZORIN D.: Illustrating smooth surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00* (New York, New York, USA, 2000), ACM Press, pp. 517–526. 10.1145/344779.345074.

[IHS02] ISENBERG T., HALPER N., STROTHOTTE T.: Stylizing Silhouettes at Interactive Rates: From Silhouette Edges to Silhouette Strokes. *Computer Graphics Forum (Proceedings of Eurographics2002) 21*, 3 (2002), 249–258.

[INC*06] ISENBERG T., NEUMANN P., CARPENDALE S., SOUSA M. C., JORGE J. A.: Non-photorealistic rendering in context. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering - NPAR '06* (New York, New York, USA, 2006), ACM Press, p. 115. 10.1145/1124728.1124747.

[Itt00] ITTI L.: A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research 40*, 10-12 (June 2000), 1489–1506. 10.1016/S0042-6989(99)00163-7.

[JCW11] JESCHKE S., CLINE D., WONKA P.: Estimating Color and Texture Parameters for Vector Graphics. *Computer Graphics Forum 2*, 30 (Apr. 2011).

[JDR04] JAGNOW R., DORSEY J., RUSHMEIER H.: Stereological techniques for solid textures. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 329. 10.1145/1015706.1015724.

[JF97] J. L., FIBUSH D.: *Sarnoff JND vision model*. Tech. rep., T1A1.5 Working Group Document {#97-612}, ANSI T1 Standards Committee, 1997.

[JGSF73] JULESZ B., GILBERT E. N., SHEPP L. A., FRISCH H. L.: Inability of humans to discriminate between visual textures that agree in second-order statistics – revisited. *Perception 2*, 4 (1973), 391–405. 10.1068/p020391.

[Ji02] JI L.: Robust topology-adaptive snakes for image segmentation. *Image and Vision Computing 20*, 2 (Feb. 2002), 147–164. 10.1016/S0262-8856(01)00093-2.

[Joh99] JOHNSTON S.: *Nonphotorealistic rendering with Renderman*. Morgan Kaufmann Publishers Inc., 1999, ch. 16, pp. 441—-480.

[KBD*10] KAGAYA M., BRENDEL W., DENG Q., KESTERSON T., TODOROVIC S., NEILL P. J., ZHANG E.: Video painting with space-time-varying style parameters. *IEEE transactions on visualization and computer graphics 17*, 1 (2010), 74–87. 10.1109/TVCG.2010.25.

[KC05] KAPLAN M., COHEN E.: A Generative Model For Dynamic Canvas Motion. In *Proceedings of the First Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging 2005* (Aire-la-Ville, Switzerland, 2005), Neumann L., Casasayas M. S., Gooch B., Purgathofer W., (Eds.), Eurographics Association, pp. 49–56.

[KCODL06] KOPF J., COHEN-OR D., DEUSSEN O., LISCHINSKI D.: Recursive Wang tiles for real-time blue noise. *ACM Transactions on Graphics 25*, 3 (July 2006), 509. 10.1145/1141911.1141916.

[KDMF03] KALNINS R. D., DAVIDSON P. L., MARKOSIAN L., FINKELSTEIN A.: Coherent stylized silhouettes. *ACM Transactions on Graphics 22*, 3 (July 2003), 856. 10.1145/882262.882355.

[KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture optimization for example-based synthesis. *ACM Transactions on Graphics 24*, 3 (July 2005), 795. 10.1145/1073204.1073263.

[Ken38] KENDALL M. G.: A New Measure of Rank Correlation. *Biometrika 30*, 1-2 (1938), 81–93.

[Ken75] KENDALL M. G.: *Rank correlation methods*. Hafner Publishing Company, Inc, 1975.

[KFCO*07] KOPF J., FU C.-W., COHEN-OR D., DEUSSEN O., LISCHINSKI D., WONG T.-T.: Solid texture synthesis from 2D exemplars. *ACM Transactions on Graphics 26*, 3 (July 2007), 2. 10.1145/1276377.1276380.

[KGC00] KAPLAN M., GOOCH B., COHEN E.: Interactive artistic rendering. In *Proceedings of the first international symposium on Non-photorealistic animation and rendering - NPAR '00* (New York, New York, USA, 2000), ACM Press, pp. 67–74. 10.1145/340916.340925.

[KH11] KARSCH K., HART J. C.: Snaxels on a plane. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR) 2011* (2011).

[KK11] KYPRIANIDIS J. E., KANG H.: Image and video abstraction by coherence-enhancing filtering. *Computer Graphics Forum 30*, 2 (2011). Proceedings Eurographics 2011.

[KLK*00] KLEIN A. W., LI W., KAZHDAN M. M., CORRÊA W. T., FINKELSTEIN A., FUNKHOUSER T. A.: Non-photorealistic virtual environments. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00* (New York, New York, USA, 2000), ACM Press, pp. 527–534. 10.1145/344779.345075.

[KMI*09] KIM S. Y., MACIEJEWSKI R., ISENBERG T., ANDREWS W. M., CHEN W., SOUSA M. C., EBERT D. S.: Stippling by example. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering - NPAR '09* (New York, New York, USA, 2009), ACM Press, p. 41. 10.1145/1572614.1572622.

[KMM*02] KALNINS R. D., MARKOSIAN L., MEIER B. J., KOWALSKI M. A., LEE J. C., DAVIDSON P. L., WEBB M., HUGHES J. F., FINKELSTEIN A.: WYSIWYG NPR: drawing strokes directly on 3D models. In *ACM Transactions on Graphics* (New York, New York, USA, July 2002), vol. 21, ACM Press, p. 755. 10.1145/566654.566648.

[KP11] KASS M., PESARE D.: Coherent noise for non-photorealistic rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH)* (Aug. 2011).

[KWH06] KOLLIOPOULOS A., WANG J. M., HERTZMANN A.: Segmentation-Based 3D Artistic Rendering. In *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering2006)* (2006), pp. 361–370.

[KWT88] KASS M., WITKIN A., TERZOPOULOS D.: Snakes: Active contour models. *International Journal of Computer Vision 1*, 4 (1988), 321–331.

[LCTS05] LEDDA P., CHALMERS A., TROSCIANKO T., SEETZEN H.: Evaluation of tone mapping operators using a High Dynamic Range display. *ACM Transactions on Graphics 24*, 3 (July 2005), 640. 10.1145/1073204.1073242.

[LD05] LAGAE A., DUTRÉ P.: A procedural object distribution function. *ACM Transactions on Graphics 24*, 4 (Oct. 2005), 1442–1461. 10.1145/1095878.1095888.

[LD06] LUFT T., DEUSSEN O.: Real-time watercolor illustrations of plants using a blurred depth test. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering - NPAR '06* (New York, New York, USA, 2006), ACM Press, p. 11. 10.1145/1124728.1124732.

[Lew84] LEWIS J.-P.: Texture synthesis for digital painting. *ACM SIGGRAPH Computer Graphics 18*, 3 (July 1984), 245–252. 10.1145/964965.808605.

[Lew89] LEWIS J. P.: Algorithms for solid noise synthesis. In *ACM SIGGRAPH Computer Graphics* (July 1989), vol. 23, pp. 263–270. 10.1145/74334.74360.

[LH06] LEFEBVRE S., HOPPE H.: Appearance-space texture synthesis. In *ACM Transactions on Graphics* (New York, New York, USA, July 2006), vol. 25, ACM Press, p. 541. 10.1145/1141911.1141921.

[Lit97] LITWINOWICZ P.: Processing images and video for an impressionist effect. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97* (New York, New York, USA, 1997), ACM Press, pp. 407–414. 10.1145/258734.258893.

[LLC*10] LAGAE A., LEFEBVRE S., COOK R., DEROSE T., DRETTAKIS G., EBERT D., LEWIS J., PERLIN K., ZWICKER M.: A Survey of Procedural Noise Functions. In *Computer Graphics Forum* (Dec. 2010), vol. 29, pp. 2579–2600. 10.1111/j.1467-8659.2010.01827.x.

[LLD10] LAGAE A., LEFEBVRE S., DUTRE P.: *Improving Gabor Noise.* Report CW 569, Department of Computer Science, K.U.Leuven, Oct. 2010.

[LLDD09] LAGAE A., LEFEBVRE S., DRETTAKIS G., DUTRÉ P.: Procedural noise using sparse Gabor convolution. *ACM Transactions on Graphics 28*, 3 (July 2009), 1. 10.1145/1531326.1531360.

[LMHB00] LAKE A., MARSHALL C., HARRIS M., BLACKSTEIN M.: Stylized rendering techniques for scalable real-time 3D animation. In *Proceedings of the first international symposium on Non-photorealistic animation and rendering - NPAR '00* (New York, New York, USA, 2000), ACM Press, pp. 13–20. 10.1145/340916.340918.

[LMJY95] LANDY M. S., MALONEY L. T., JOHNSTON E. B., YOUNG M.: Measurement and modeling of depth cue combination: in defense of weak fusion. *Vision Research 35*, 3 (1995), 389–412.

[LMLH07] LEE Y., MARKOSIAN L., LEE S., HUGHES J. F.: Line drawings via abstracted shading. In *ACM SIGGRAPH 2007 papers on - SIGGRAPH '07* (New York, New York, USA, 2007), ACM Press, p. 18. 10.1145/1275808.1276400.

[LSF10] LU J., SANDER P. V., FINKELSTEIN A.: Interactive painterly stylization of images, videos and 3D animations. In *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2010), ACM, pp. 127–134. 10.1145/1730804.1730825.

[LVLD10] LAGAE A., VANGORP P., LENAERTS T., DUTRÉ P.: Procedural isotropic stochastic textures by example. *Computers & Graphics 34*, 4 (Aug. 2010), 312–321. 10.1016/j.cag.2010.05.004.

[LZL*10] LIN L., ZENG K., LV H., WANG Y., XU Y., ZHU S.-C.: Painterly animation using video semantics and feature correspondence. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering - NPAR '10* (New York, New York, USA, 2010), vol. 1, ACM Press, p. 73. 10.1145/1809939.1809948.

[McI00] MCINERNEY T.: T-snakes: Topology adaptive snakes. *Medical Image Analysis 4*, 2 (June 2000), 73–91. 10.1016/S1361-8415(00)00008-6.

[Mei96] MEIER B. J.: Painterly rendering for animation. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96* (New York, New York, USA, 1996), ACM Press, pp. 477–484. 10.1145/237170.237288.

[MESA*10] MCGUIRE M., EKANAYAKE C., ST-AMOUR J.-F., HAL\'EN H., THIBAULT A., MARTEL B.: Stylized Rendering in Games. In *ACM SIGGRAPH 2010 Courses* (Los Angeles, California, 2010), ACM.

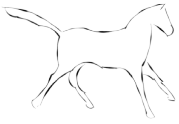
[MH04] MCGUIRE M., HUGHES J. F.: Hardware-determined feature edges. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering - NPAR '04* (New York, New York, USA, 2004), ACM Press, p. 35. 10.1145/987657.987663.

[MIA*08] MACIEJEWSKI R., ISENBERG T., ANDREWS W. M., EBERT D. S., SOUSA M. C., CHEN W.: Measuring Stipple Aesthetics in Hand-Drawn and Computer-Generated Images. *IEEE Computer Graphics and Applications 28*, 2 (Mar. 2008), 62–74. 10.1109/MCG.2008.35.

[MKG*97] MARKOSIAN L., KOWALSKI M. A., GOLDSTEIN D., TRYCHIN S. J., HUGHES J. F., BOURDEV L. D.: Real-time nonphotorealistic rendering. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97* (New York, New York, USA, 1997), ACM Press, pp. 415–420. 10.1145/258734.258894.

[MS09] MCCRAE J., SINGH K.: Sketching piecewise clothoid curves. *EG Sketch-Based Interfaces and Modeling (SBIM) 33*, 4 (2009).

[MSS98] MASUCH M., SCHUMANN L., SCHLECHTWEG S.: Animating Frame-to-Frame Consistent Line Drawings for Illustrative Purposes. In *SimVis* (1998), pp. 101–112.



[MSS99] MASUCH M., SCHLECHTWEG S., SCHULZ R.: Speedlines: depicting motion in motionless pictures. In *ACM SIGGRAPH '99 Conference Abstracts and Applications* (1999).

[MT95] MCINERNEY T., TERZOPOULOS D.: Topologically adaptable snakes. In *Computer Vision, 1995. Proceedings., Fifth International Conference on* (June 1995), pp. 840–845.

[MTAS01] MYSZKOWSKI K., TAWARA T., AKAMINE H., SEIDEL H.-P.: Perception-guided global illumination solution for animation rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01* (New York, New York, USA, 2001), ACM Press, pp. 221–230. 10.1145/383259.383284.

[MZD05] MATUSIK W., ZWICKER M., DURAND F.: Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics 24*, 3 (July 2005), 787. 10.1145/1073204.1073262.

[ND05] NIENHAUS M., DÖLLNER J.: *Blueprint Rendering and Sketchy Drawings*. Addison-Wesley Professional, 2005, ch. 15, pp. 235–252.

[Ney03] NEYRET F.: Advected Textures. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003).

[NM00] NORTHRUP J. D., MARKOSIAN L.: Artistic silhouettes. In *Proceedings of the first international symposium on Non-photorealistic animation and rendering - NPAR '00* (New York, New York, USA, 2000), ACM Press, pp. 31–37. 10.1145/340916.340920.

[NS04] NEHAB D., SHILANE P.: Stratified Point Sampling of 3D Models. In *Eurographics Symposium on Point-Based Graphics* (2004), pp. 49–56.

[OBW*08] ORZAN A., BOUSSEAU A., WINNEMÖLLER H., BARLA P., THOLLOT J., SALESIN D.: Diffusion curves: a vector representation for smooth-shaded images. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 1. 10.1145/1360612.1360691.

[OH11] O'DONOVAN P., HERTZMANN A.: AniPaint: Interactive Painterly Animation from Video. *IEEE transactions on visualization and computer graphics* (Mar. 2011). 10.1109/TVCG.2011.51.

[OHM*04] O'SULLIVAN C., HOWLETT S., MCDONNELL R., MORVAN Y., O'CONOR K.: *Perceptually Adaptive Graphics*. Tech. rep., Image Synthesis Group, Trinity College Dublin, 2004.

[Ola05] OLANO M.: Modified noise for evaluation on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware - HWWS '05* (New York, New York, USA, 2005), no. July, ACM Press, p. 105. 10.1145/1071866.1071883.

[Pal99] PALMER S. E.: *Vision Science: Photons to Phenomenology*. MIT Press, 1999.

[Per85] PERLIN K.: An image synthesizer. *ACM SIGGRAPH Computer Graphics 19*, 3 (July 1985), 287–296. 10.1145/325165.325247.

[PFH00] PRAUN E., FINKELSTEIN A., HOPPE H.: Lapped textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00* (New York, New York, USA, 2000), ACM Press, pp. 465–470. 10.1145/344779.344987.

[PFS03] PASTOR O., FREUDENBERG B., STROTHOTTE T.: Real-time animated stippling. *IEEE Computer Graphics and Applications 23*, 4 (July 2003), 62–68. 10.1109/MCG.2003.1210866.

[PHWF01] PRAUN E., HOPPE H., WEBB M., FINKELSTEIN A.: Real-time hatching. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01* (New York, New York, USA, 2001), Fiume E., (Ed.), ACM Press, p. 581. 10.1145/383259.383328.

[PS00] PORTILLA J., SIMONCELLI E. P.: A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. *International Journal of Computer Vision 40*, 1 (2000), 49–70. 10.1023/A:1026553619983.

[QY07] QIN X., YANG Y.-H.: Aura 3D textures. *IEEE transactions on visualization and computer graphics 13*, 2 (2007), 379–89. 10.1109/TVCG.2007.31.

[Rad99] RADEMACHER P.: View-dependent geometry. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99* (New York, New York, USA, 1999), ACM Press, pp. 439–446. 10.1145/311535.311612.

[RCDF08] RUSINKIEWICZ S., COLE F., DECARLO D., FINKELSTEIN A.: Line drawings from 3D models. In *ACM SIGGRAPH 2008 classes on - SIGGRAPH '08* (New York, New York, USA, 2008), ACM Press, p. 1. 10.1145/1401132.1401188.

[RFWB07] RAMANARAYANAN G., FERWERDA J., WALTER B., BALA K.: Visual equivalence. In *ACM Transactions on Graphics* (July 2007), vol. 26, p. 76. 10.1145/1276377.1276472.

[Ris86] RISSET J.-C.: Pitch and rhythm paradoxes: Comments on "Auditory paradox based on fractal waveform". *The Journal of the Acoustical Society of America 80*, 3 (1986), 961–962. 10.1121/1.393919.

[RLW*09] RAY N., LÉVY B., WANG H., TURK G., VALLET B.: Material Space Texturing. *Computer Graphics Forum 28*, 6 (Sept. 2009), 1659–1669. 10.1111/j.1467-8659.2009.01423.x.

[RSK10] RUITERS R., SCHNABEL R., KLEIN R.: Patch-based Texture Interpolation. *Computer Graphics Forum 29*, 4 (Aug. 2010), 1421–1429. 10.1111/j.1467-8659.2010.01739.x.

[RTAK07] RAPANTZIKOS K., TSAPATSOULIS N., AVRITHIS Y., KOLLIAS S.: Bottom-up spatiotemporal visual attention model for video analysis. *IET Image Processing 1*, 2 (2007), 237. 10.1049/iet-ipr:20060040.

[Rus04] RUSINKIEWICZ S.: *Estimating curvatures and their derivatives on triangle meshes*. IEEE, 2004.

[SB06] SHEIKH H., BOVIK A.: Image information and visual quality. *IEEE Transactions on Image Processing 15*, 2 (Feb. 2006), 430–444. 10.1109/TIP.2005.859378.

[SB10] SESHADRINATHAN K., BOVIK A. C.: Motion tuned spatio-temporal quality assessment of natural videos. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society 19*, 2 (Feb. 2010), 335–50. 10.1109/TIP.2009.2034992.

[SBCv*11] SÝKORA D., BEN-CHEN M., ČADÍK M., WHITED B., SIMMONS M.: Textoons: Practical texture mapping for hand-drawn cartoon animations. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering* (2011).

[SC08] SHESH A., CHEN B.: Efficient and Dynamic Simplification of Line Drawings. *Computer Graphics Forum 27*, 2 (Apr. 2008), 537–545. 10.1111/j.1467-8659.2008.01151.x.

[SD04] SANTELLA A., DECARLO D.: Visual interest and NPR: an evaluation and manifesto. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering* (2004), ACM, p. 150.

[SDC09] SÝKORA D., DINGLIANA J., COLLINS S.: As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering* (2009), pp. 25–33.

[SFWG04] STOKES W. A., FERWERDA J. A., WALTER B., GREENBERG D. P.: Perceptual illumination components. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 742. 10.1145/1015706.1015795.

[She29] SHEPARD R.: Circularity in judgments of relative pitch. *Psychol. Rev 36* (1929), 172–180.

[SR05] SIMONS D. J., RENSINK R. A.: Change blindness: past, present, and future. *Trends in cognitive sciences 9*, 1 (Jan. 2005), 16–20. 10.1016/j.tics.2004.11.006.

[SS02] STROTHOTTE T., SCHLECHTWEG S.: *Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation*. Morgan Kaufmann, 2002.

[SS09] SCHWARZ M., STAMMINGER M.: On predicting visual popping in dynamic scenes. In *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization - APGV '09* (New York, New York, USA, 2009), ACM Press, p. 93. 10.1145/1620993.1621012.

[SSB06] SHEIKH H. R., SABIR M. F., BOVIK A. C.: A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms. *IEEE Transactions on Image Processing 11* (2006), 3440–3451.

[SSBG10] SCHMID J., SUMNER R. W., BOWLES H., GROSS M.: Programmable motion effects. *ACM Transactions on Graphics 29*, 4 (July 2010), 1. 10.1145/1778765.1778794.

[SSGS11] SCHMID J., SENN M. S., GROSS M., SUMNER R. W.: OverCoat: an implicit canvas for 3d painting. *ACM Transactions on Graphics (Proc. SIGGRAPH)* (Aug. 2011).

[SSLR96] SCHUMANN J., STROTHOTTE T., LASER S., RAAB A.: Assessing the effect of non-photorealistic rendered images in CAD. In *Proceedings of the SIGCHI conference on Human factors in computing systems common ground - CHI '96* (New York, New York, USA, 1996), ACM Press, pp. 35–41. 10.1145/238386.238398.

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-D shapes. *Proceedings of the 17th annual conference on Computer graphics and interactive techniques - SIGGRAPH '90 24*, 4 (1990), 197–206. 10.1145/97879.97901.

[ST06] SAND P., TELLER S.: Particle video: Long-range motion estimation using point trajectories. In *Proc. of IEEE CVPR* (2006), pp. 2195–2202.

[SZK*06] SNAVELY N., ZITNICK C. L., KANG S. B., , COHEN M. F.: Stylizing 2.5-d video. In *Proc. Symposium on Non-Photorealistic Animation and Rendering (NPAR)* (2006).

[TJ93] TUCERYAN M., JAIN A. K.: Texture analysis. *Handbook of pattern recognition & computer vision 1* (1993), 235–276.

[TMY78] TAMURA H., MORI S., YAMAWAKI T.: Textural Features Corresponding to Visual Perception. *IEEE Transactions on Systems, Man, and Cybernetics 8*, 6 (1978), 460–473. 10.1109/TSMC.1978.4309999.

[Tod04] TODD J. T.: The visual perception of 3D shape. *Trends in cognitive sciences 8*, 3 (Mar. 2004), 115–21. 10.1016/j.tics.2004.01.006.

[Tor58] TORGERSON W. S.: *Theory and methods of scaling*. Wiley, 1958.

[TTD*07] TODD J. T., THALER L., DIJKSTRA T. M. H., KOENDERINK J. J., KAPPERS A. M. L.: The effects of viewing angle, camera angle, and sign of surface curvature on the perception of three-dimensional shape from texture. *Journal of Vision 7*, 12 (Sept. 2007), 1–16. 10.1167/7.12.9.

[UJ03] UMBACH D., JONES K. N.: A few methods for fitting circles to data. *IEEE Transactions on instrumentation and measurement 52*, 6 (2003), 1881–1885.

[USSK11] UMENHOFFER T., SZÉCSI L., SZIRMAY-KALOS L.: Hatching for motion picture production. *Comp. Graphics Forum 30* (2011), 533–542.

[VBTS07] VANDERHAEGHE D., BARLA P., THOLLOT J., SILLION F.: Dynamic point distribution for stroke-based rendering. In *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering2007)* (2007), pp. 139–146.

[vdBV96] VAN DEN BRANDEN LAMBRECHT C. J., VERSCHEURE O.: Perceptual Quality Measure using a Spatio-Temporal Model of the Human Visual System. In *IS&T/SPIE* (1996).

[Ver02] VERYOVKA O.: Animation with Threshold Textures. In *Proceedings of Graphics Interface 2002* (2002), pp. 9–16.

[VPB*09] VERGNE R., PACANOWSKI R., BARLA P., GRANIER X., SCHLICK C.: Light warping for enhanced surface depiction. *ACM Transactions on Graphics 28*, 3 (July 2009), 1. 10.1145/1531326.1531331.

[VVC*11] VERGNE R., VANDERHAEGHE D., CHEN J., BARLA P., GRANIER X., SCHLICK C.: Implicit Brushes for stylized Line-based rendering. *Comp. Graphics Forum* (2011).

[vW91] VAN WIJK J. J.: Spot noise texture synthesis for data visualization. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques - SIGGRAPH '91* (New York, New York, USA, 1991), vol. 25, ACM Press, pp. 309–318. 10.1145/122718.122751.

[vWNA08] ČAD\'IK M., WIMMER M., NEUMANN L., ARTUSI A.: Evaluation of HDR Tone Mapping Methods Using Essential Perceptual Attributes. *Computers & Graphics 32*, 3 (2008), 330–349.

[WAKB09] WILLS J., AGARWAL S., KRIEGMAN D., BELONGIE S.: Toward a perceptual space for gloss. *ACM Transactions on Graphics 28*, 4 (Aug. 2009), 1–15. 10.1145/1559755.1559760.

[WBC*07] WALLRAVEN C., BÜLTHOFF H. H., CUNNINGHAM D. W., FISCHER J., BARTZ D.: Evaluation of real-world and computer-generated stylized facial expressions. *ACM Transactions on Applied Perception 4*, 3 (Nov. 2007). 10.1145/1278387.1278390.

[WBCG09] WEBER O., BEN-CHEN M., GOTSMAN C.: Complex Barycentric Coordinates with Applications to Planar Shape Deformation. *Computer Graphics Forum 28*, 2 (Apr. 2009), 587–597. 10.1111/j.1467-8659.2009.01399.x.

[WBS*04] WANG Z., BOVIK A. C., SHEIKH H. R., MEMBER S., SIMONCELLI E. P.: Image quality assessment: from error measurement to structural similarity. *IEEE Transactions on Image Processing 13* (2004), 600–612.

[WFGS07] WINNEMÖLLER H., FENG D., GOOCH B., SUZUKI S.: Using NPR to evaluate perceptual shape cues in dynamic environments. In *Proc. 5th Int. Symposium on Non-Photorealistic Animation and Rendering* (2007), pp. 85–92.

[WG10] WEBER O., GOTSMAN C.: Controllable conformal maps for shape deformation and interpolation. *ACM Transactions on Graphics 29*, 4 (July 2010), 1. 10.1145/1778765.1778815.

[Wil45] WILCOXON F.: Individual Comparisons by Ranking Methods. *Biometrics Bulletin 1*, 6 (1945), 80–83.

[Wil83] WILLIAMS L.: *Pyramidal parametrics*. ACM Press, New York, New York, USA, 1983.

[Wil97] WILLATS J.: *Art and representation: new principles in the analysis of pictures*. Princeton University Press, 1997.

[Win05] WINKLER S.: Perceptual Video Quality Metrics – A Review. In *Digital Video Image Quality and Perceptual Coding*. CRC Press, 2005.

[WM02] WATSON A. B., MALO J.: Video quality measures based on the standard spatial observer. In *Image Processing. 2002. Proceedings. 2002 International Conference on* (2002), vol. 3, IEEE.

[WOBT09] WINNEMÖLLER H., ORZAN A., BOISSIEUX L., THOLLOT J.: Texture Design and Draping in 2D Images. *Computer Graphics Forum 28*, 4 (June 2009), 1091–1099. 10.1111/j.1467-8659.2009.01486.x.

[WOG06] WINNEMÖLLER H., OLSEN S. C., GOOCH B.: Real-time video abstraction. *ACM Transactions on Graphics (proc. of SIGGRAPH 2006) 25*, 3 (2006), 1221 – 1226.

[WXSC04] WANG J., XU Y., SHUM H.-Y., COHEN M. F.: Video tooning. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 574. 10.1145/1015706.1015763.

[WY04] WU Q., YU Y.: Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 364. 10.1145/1015706.1015730.

[YJ84] YANTIS S., JONIDES J.: Abrupt visual onsets and selective attention: evidence from visual search. *J. Experimental Psychology 10*, 5 (1984), 601–621.

[YNBH10] YU Q., NEYRET F., BRUNETON E., HOLZSCHUCH N.: Lagrangian Texture Advection: Preserving both Spectrum and Velocity Field. *IEEE transactions on visualization and computer graphics* (Dec. 2010), 1–13. 10.1109/TVCG.2010.263.

[YPG01] YEE H., PATTANAIK S., GREENBERG D. P.: Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Transactions on Graphics 20*, 1 (Jan. 2001), 39–65. 10.1145/383745.383748.