Master of Science in Informatics at Grenoble
Master Informatique
Specialization Graphics, Vision and Robotics

Procedural Stylization

Isnel Maxime

June 2019

Research project performed at MAVERICK

Under the supervision of:
Romain Vergne
Joëlle Thollot

Defended before a jury composed of:
Head of the jury
Jury member 1
Jury member 2

June 2019

Abstract

Your abstract goes here...

Acknowledgement

I would like to express my sincere gratitude to .. for his invaluable assistance and comments in reviewing this report... Good luck :)

Résumé

Your abstract in French goes here...

Contents

— 1 —
Introduction

1.1 Background

1.2 Problem Statement

A part of the computer graphics is create non-photorealistic images. A method to do this is to stylize 3D objects and 3D scenes. Stylize an object means create an image that imitates the style of an artist who would have drawn it on sheet of paper. There exist many different styles like hand drawing, brush painting, pointillism painting, stippling, watercolor painting, etc.

The main problem of stylizing a 3D object in an animation is the *temporal coherence*. The *temporal coherence* problem in non-photorealistic rendering encompasses both spatial and temporal aspects of the marks. The effect given by the stylization has to be kept if the object is moving, rotating and scaling. Many research has been done to solve this problem of *temporal coherence* [25, 7, 3]. Bénard et al. separate this problem is three sections inspired by previous work[19, 11, 5, 8] the ideal solution (Figure 1.1 a) correspond to something drawn by an artist at each frame.

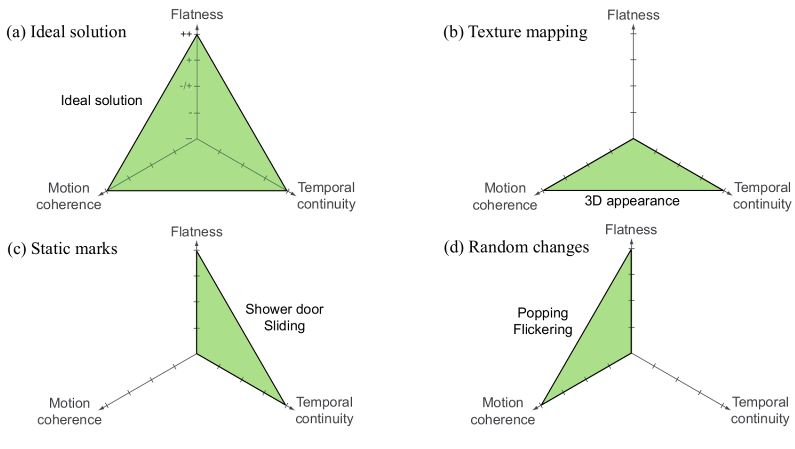

Figure 1.1: Problem involve by temporal coherence depending of the flatness, temporal continuity, motion coherence

1.2.1 Flatness

The impression of drawing on a flat surface gives the *flatness*. The stylization has a good *flatness* if the image rendered has a good 2D appearance. In order to keep this effect, the size and the distribution of the marks of your stylization have to be independent of the distance between the stylized object and the camera.

1.2.2 Motion Coherence

Motion coherence is a correlation between the motion of marks and the motion of the 3D object. Bad *Motion coherence* will give the impression to see the scene through a semi-transparent layer of marks, this is called *shower door* effect [19], an example to illustrate what happens when there is a bad *Motion coherence* is the movie *Loving Vincent*[2]. 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.

1.2.3 Temporal continuity

Temporal continuity is the quality of minimizing changes from frame to frame to ensure fluid animations. In order to have good *temporal continuity*, the marks of the image have to fade slowly during the animation. Human perception is very sensitive to *temporal incoherence* according to some perceptual studies[26, 24].

The problem introduced in the works of Bénard et al.[8] is the difficulty to have a ideal solution, the one that has a good *flatness*, a good *motion coherence* and a good *temporal continuity*. These three goals are inherently contradictory when you improve one you neglect one or maybe more. So researchers work to find solutions that make *trade-offs* between these three goals. Our solution is a *trade-offs* too.

1.3 Scientific approach

1.4 Contents of this report

— 2 —
Previous Work

Many research have been done in stylizing 3D scene[23, 22, 16, 7, 6, 13, 8] trying to propose solutions or trade-offs for the problem of *temporal coherence*. In this part of this report, we will present you techniques to stylize 3D scenes and we will show their advantages and disadvantages. In order to render an image of a 3D object in the screen, a graphics program goes through several steps that compute some different information like the gradient of the image, the shadows made by the object, the amount of light received by the object, etc. The gathering of all these steps is called **graphical pipeline rendering**. In this graphical pipeline, there are two moments when we can stylize the objects. The first is when we computed information about the geometry of each object in the scene, we call it *object space*. The second moment is when we gather the previously computed images of the scene in order to make for example shadows, global illumination, ambient occlusion, etc. we call it the *image space*. We will treat these two space separately and with the two different types of methods to stylize.

2.1 Object Space

In the object space, we work on the surface of the object and so we have all the knowledge about the geometry.

Texture-based methods
One of the most used ways to colored object in 3D is the *texture mapping*. It consists of mapping an image on the object 2.1. This technique is widely used in video games because it is easy to implement, it can be implemented for GPU and it needs low computation. As shown in the example, texture mapping can be used with images as a texture in order to stylize the object[22, 16, 13]. Texture mapping can also be done with *procedural noises*[21] in this case

Figure 2.1: Texture mapping: example

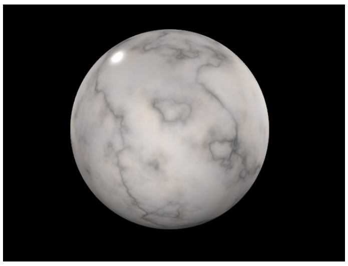

Figure 2.2: Marble with procedural texture

called *procedural textures*. Procedural textures are mathematically computed from coordinates the most famous is Perlin but there exist others like Gabor noise, Worley noise, etc. With these textures, we can create images looks like marble 2.2 or realist wood. Bénard et al.[7, 6] use this method of texture mapping with noise in order to make watercolor stylization.

This method has a problem of *foreshortening* of the silhouettes which makes an area with more elements than in the center of the object and when zooming in/out the size of the elements in the mapped texture vary depending on the distance from the camera, if we get closer to the object the elements on the texture become bigger and bigger going up to pixelization sometimes. These two problems make texture mapping bad in term of *flatness* because an artist does not draw bigger strokes if the object is close and does not draw more details at the silhouettes than in the center of his objects. Texture mapping is also bad in term of variety of style that you can do because you are depending on the texture chosen you cannot, for instance, do a rendered image that looks like painted with brushes and especially you cannot change the shape of the object in the rendered image, a perfect sphere will always appear as a perfect sphere. On the other hand, this method naturally ensures *motion coherence* and *temporal continuity* because the texture is mapped directly on the surface of the object.

Mark based methods

The natural way to stylize 3D objects is to as an artist apply paint strokes on the object. These paint strokes can be represented with smalls images also called splats. Daniels[12] and Schmid[23] propose to project splats composed of stroke and stored them on the geometry of the model but this technique is expensive in term of storage. Some works [19, 1, 10](more in the state of the art [8]) use point distribution in order to make anchor points for splats. These point distributions are often computed in image space and then are projected on the model. Anchor these splats to the model improve the *motion coherence* because each splat will follow the motion of the 3D model. These splats are rendered in the image space as a 2D sprites so preserved the *flatness*. The problem is how to have the point distribution and how can we control it in order to have a uniform, not too sparse and not too dense distribution. Moreover, these point distribution does not provide control over the *temporal continuity*. In our method, we use procedural noise to anchor the splats.

2.2 Image space

Texture-based methods

	Motion coherence	**Flatness**	**Temporal continuity**	**Style variation**
Object space				
Texture-based methods	++	- -	++	- -
Mark based methods	++	+	-	+/-
Image space				
Texture-based methods	-	++	+	- -
Mark based methods	-	++	- -	+

Figure 2.3: Summary of trade-offs made in different approaches

Many methods to stylize in image space used texture based approaches. It consists to apply the texture to the entire image [8] but in the case of stylizing animated scenes, the problem is how do we deform the texture to minimize the apparition of sliding artefacts. We can distinguish two families of approaches to solve this problem. The first family of approaches use an approximation of the 3D camera motion with 2D transformations of the texture[11]. This gives a nice trade-off between *motion coherence* and *flatness* but it is limited to static scenes and a set of few camera motions. Moreover, sliding artefacts still occur with strong parallax so Fung et al.[14] and Breslav et al.[5] improve the approximation of the scene motion in order to reduce sliding artefacts.

The second family of approaches use non-rigid deformations to animate the texture[4]. These deformations are computed from the optical flow of a video. This is an extension of the methods used in vector field visualization by Neyret[20]. These deformations can distort the texture and alter the original pattern. The method of Bousseau et al.[4] is very effective with stochastic textures as the fractalization process but creates artefacts with structured patterns.

Mark based methods

A method very used to stylize in image space consists to draw strokes/splats at some place of the image[3, 25, 9, 27, 15]. The question of these mark based method is where do we place the marks in order to have a stylized rendering without losing the meaning of the scene. A first approach is to extract lines that are relevant like the silhouettes, etc. [25, 15, 17] and then stylize the image with this information, like keeping only the extracted lines and change the shape of each line or apply strokes along these lines as Vergne et al.[25] did try to have a good *temporal coherence*. The problem of these techniques is the popping marks due to a bad *temporal continuity*. A second approach is to segment the image in order to have the different parts of the scene[27, 18]. Thanks to this segmentation, they apply different strokes for each part of the image with the corresponding colors. The work of Lin et al.[18] is about videos so they use the optical flow of the videos in order to have a good *temporal coherence*. These mark based methods have a good impression of *flatness* thanks to the splatting in image space, this is something that we will use in our approach.

— 3 —
Realisation

3.1 Overview

As explains above in the state of the art and in the figure2.3 each approach has its advantages and its disadvantages. That is why in our solution we tried to take the better of the two worlds. We stylize the 3D scene in image space (screen space) but with all the information about the 3D object and the camera (camera matrices, position, normals, tangents, UV coordinates, distance from the camera). This solution permits to apply something like 2D images on the screen so have a good *flatness* while keeping the information on the silhouettes, the orientation, the depth, etc. This solution permits also to easily integrate the stylizing of a scene in a pipeline rendering because it can be done at the end during the post-processing rendering pass.

We chose to use mark based methods to stylize our scene because texture based methods in image space give a poor variety of styles as said in the work of Bénard et al.[7]. This mark based method implies to decide where in the image the splats will be drawn. In our problem, the goal is to anchor these splats with the objects in order to have the same motion for the splats and the object. This avoid the problem of *shower door effect* and ensure the good *motion coherence*. So we needed anchor points depending on the position of our object. Therefore in our approach, we used procedural noise[21] as a texture of our 3D object. The procedural noises are easy to implement, fast to compute and easy to manipulate. Like every texture computed in object space, it has a good motion coherence. Each value different of zero of this texture represents an anchor point for a splat.

3.2 Procedural noise and fractalization

fractalization Bénard et *al.*[6] use the same principle but with procedural textures. They create multiple noises with different frequency and combine them playing with transparency. Moreover, they overlap the noise to make an impression of infinite zoom effect (like in this example: ShaderToy). With this method patterns of the texture have an almost constant size regardless of the size of the object but it can create small problems of *temporal continuity*. In our method, we will use this technique of fractalization of a procedural noise.

3.3 Splatting

3.4 Stylization

— 4 —

Practical implementation

Bibliography

[1] *NPAR '00: Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering*, New York, NY, USA, 2000. ACM. ACM Order No: 434000.

[2] Loving vincent, 2017.

[3] Alexandre Bléron, Romain Vergne, Thomas Hurtut, and Joëlle Thollot. Motion-coherent stylization with screen-space image filters. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering - Expressive '18*, pages 1–13, Victoria, British Columbia, Canada, 2018. ACM Press.

[4] Adrien Bousseau, Fabrice Neyret, Joëlle Thollot, and David Salesin. Video watercolorization using bidirectional texture advection. *ACM Transactions on Graphics*, 26(3):104, July 2007.

[5] Simon Breslav, Karol Szerszen, Lee Markosian, Pascal Barla, and Joëlle Thollot. Dynamic 2d Patterns for Shading 3d Scenes. page 6.

[6] P. Bénard, A. Lagae, P. Vangorp, S. Lefebvre, G. Drettakis, and J. Thollot. A Dynamic Noise Primitive for Coherent Stylization. *Computer Graphics Forum*, 29(4):1497–1506, August 2010.

[7] Pierre Bénard, Adrien Bousseau, and Joëlle Thollot. Dynamic solid textures for real-time coherent stylization. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games - I3D '09*, page 121, Boston, Massachusetts, 2009. ACM Press.

[8] Pierre Bénard, Adrien Bousseau, and Joëlle Thollot. State-of-the-Art Report on Temporal Coherence for Stylized Animations. *Computer Graphics Forum*, 30(8):2367–2386, December 2011.

[9] Pierre Bénard, Jingwan Lu, Forrester Cole, Adam Finkelstein, and Joëlle Thollot. Active Strokes: Coherent Line Stylization for Animated 3d Models. page 10.

[10] Ming-Te Chi and Tong-Yee Lee. Stylized and abstract painterly rendering system using a multiscale segmented sphere hierarchy. *IEEE transactions on visualization and computer graphics*, 12(1):61–72, February 2006.

[11] Matthieu Cunzi, Joelle Thollot, Jean-Dominique Gascuel, Sylvain Paris, and Gilles Debunne. Dynamic Canvas for Non-Photorealistic Walkthroughs. page 10.

[12] Eric Daniels. Deep canvas in disney's tarzan. In *ACM SIGGRAPH 99 Electronic Art and Animation Catalog*, SIGGRAPH '99, pages 124–, New York, NY, USA, 1999. ACM.

[13] Bert Freudenberg, Maic Masuch, and Thomas Strothotte. Walk-Through Illustrations: Frame-Coherent Pen-and-Ink Style in a Game Engine. *Computer Graphics Forum*, 20(3):184–192, 2001.

[14] Jennifer Fung and Oleg Veryovka. "Pen-and-ink textures for real-time rendering". page 6.

[15] Stéphane Grabli, Emmanuel Turquin, Frédo Durand, and François X. Sillion. Programmable rendering of line drawing from 3d scenes. *ACM Transactions on Graphics*, 29(2):1–20, March 2010.

[16] Allison W. Klein, Wilmot Li, Michael M. Kazhdan, Wagner T. Corrêa, Adam Finkelstein, and Thomas A. Funkhouser. Non-photorealistic virtual environments. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, pages 527–534, Not Known, 2000. ACM Press.

[17] Yunjin Lee, Lee Markosian, Seungyong Lee, and John F Hughes. Line drawings via abstracted shading. page 5.

[18] Liang Lin, Kun Zeng, Yizhou Wang, Ying-Qing Xu, and Song-Chun Zhu. Video Stylization: Painterly Rendering and Optimization with Content Extraction. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, page 13.

[19] Barbara J. Meier. Painterly rendering for animation. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, pages 477–484, Not Known, 1996. ACM Press.

[20] Fabrice Neyret. iMAGIS-GRAVIR† / IMAG-INRIA. page 8.

[21] Ken Perlin. Improving Noise. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 681–682, New York, NY, USA, 2002. ACM. event-place: San Antonio, Texas.

[22] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time hatching. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, page 581, Not Known, 2001. ACM Press.

[23] Johannes Schmid, Martin Sebastian Senn, Markus Gross, and Robert W. Sumner. OverCoat: an implicit canvas for 3d painting. In *ACM SIGGRAPH 2011 papers on - SIGGRAPH '11*, page 1, Vancouver, British Columbia, Canada, 2011. ACM Press.

[24] Michael Schwarz and Marc Stamminger. On predicting visual popping in dynamic scenes. In *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization*, APGV '09, pages 93–100, New York, NY, USA, 2009. ACM.

[25] Romain Vergne, David Vanderhaeghe, Jiazhou Chen, Pascal Barla, Xavier Granier, and Christophe Schlick. Implicit Brushes for Stylized Line-based Rendering. *Computer Graphics Forum*, 30(2):513–522, April 2011.

[26] Steven Yantis and John Jonides. Abrupt visual onsets and selective attention: Evidence from visual search. *Journal of experimental psychology. Human perception and performance*, 10:601–21, 11 1984.

[27] Kun Zeng, Mingtian Zhao, Caiming Xiong, and Song-Chun Zhu. From image parsing to painterly rendering. *ACM Transactions on Graphics*, 29(1):1–11, December 2009.