Automatic Generation of Artworks Using Virtual Photoelastic Material

Kazuki Miyazaki
Keio University
kazuki.miyazaki@fj.ics.keio.ac.jp

Issei Fujishiro
Keio University
fuji@fj.ics.keio.ac.jp

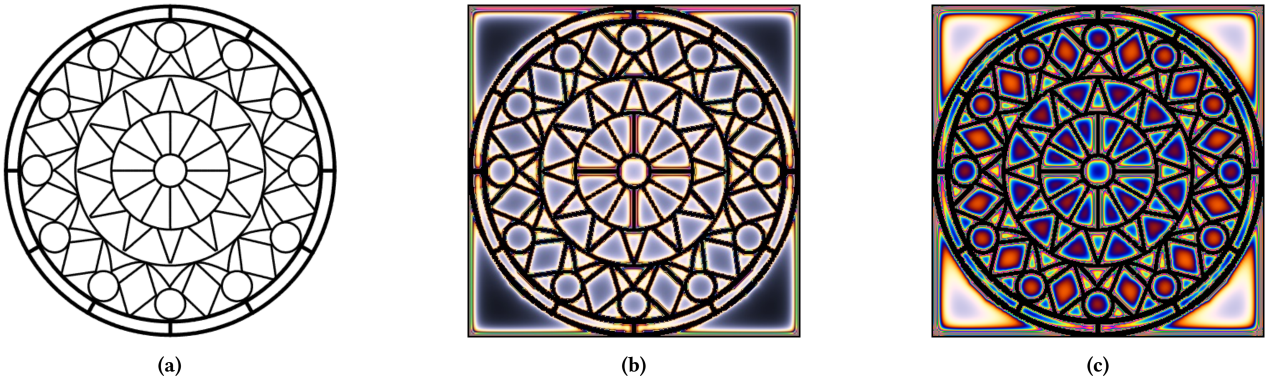

Figure 1: The visual effect of photoelastic artwork. From the same input frame in (a), different color patterns as in (b) and (c) can be generated only by varying the values of virtual stress.

ABSTRACT

Photoelasticity is known as one of the phenomena related to polarization and is defined as the change in birefringence of transparent material when internal force is applied. Interference fringes appear by irradiating the material with polarized light when viewing it through the polarizer. In this study, we attempt to apply the concept of photoelasticity to generative art. Assuming there is virtual stress distribution in the two-dimensional material, our method automatically generates artworks with photoelasticity. A GPU-based acceleration of the current implementation is also discussed.

CCS CONCEPTS

• **Computing methodologies → Image processing**;

KEYWORDS

Generative art, photoelasticity, image manipulation.

ACM Reference Format:
Kazuki Miyazaki and Issei Fujishiro. 2018. Automatic Generation of Artworks Using Virtual Photoelastic Material. In *Proceedings of SIGGRAPH '18 Posters.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3230744.3230797

1 BACKGROUND AND PURPOSE

Image manipulation is a representative approach to generative art. Orzan, et al. [Orzan et al. 2008] proposed a two–dimensional method of coloring an image with the input of contours and their color distributions that users can edit. In this study, we propose yet another method in which an input image is colorized based on the theory of two–dimensional photoelasticity. Buβler, et al. [Buβler et al. 2015] simulated photoelasticity using existing stress datasets. However, their simulation is only used for the visualization of stress tensor fields. In contrast, our method utilizes photoelasticity for generating artworks; captivating interference fringes can be produced along the contours extracted from the input image, as shown in **Figure 1**.

2 PROPOSED METHOD

To initiate the process, we extract contours from the input image to arrange the stress virtually and generate the interference fringes along them. By regarding a pixel in the input image as a particle in the material, we calculate its distance from each contour edge and create the virtual stress field from the aggregate of the tensors for those pixels. Note that each tensor is normalized by the length of an edge to reduce the influence of short contours and the considerable variation in stress among neighboring pixels. After estimating the stress field, we perform the eigenanalysis of stress tensors to obtain the principal stress and the principal axis, both of which are used for the stress analysis with photoelasticity.

Photoelasticity is related not only to the stress analysis, but also to the optical phenomena including polarization and birefringence. We use the linear polarization to reproduce photoelasticity. In general, light has an infinite number of factors including amplitude directions and wavelengths, though the linear polarization only has

---

SIGGRAPH '18 Posters, August 12-16, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5817-0/18/08.
https://doi.org/10.1145/3230744.3230797

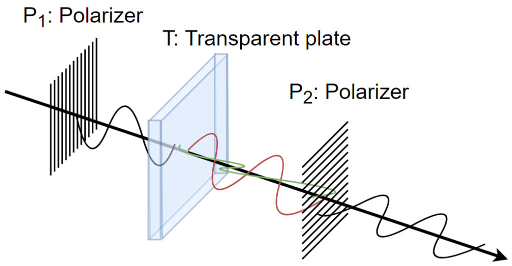

Figure 2: The linear polariscope model adopted in our method. The arrow indicates the direction of the incident light.

the amplitude directions in the same plane. When passing through the material of birefringence, the incident light can be separated into two different rays, whereas birefringence does not arise in a certain direction. This direction is referred to as the *light axis* of the material. When the stress is applied to the transparent and optically isotropic material, such as glass and acrylic resin, it changes into the anisotropic material having two light axes. This kind of phenomena is called *photoelasticity*.

Figure 2 illustrates our model for generating interference fringes. In this model, polarizers P$_1$ and P$_2$ are arranged behind and before the transparent plate T and are irradiated from behind with the white light comprising a countless number of wavelengths. Vibrating surfaces of polarized light made by P$_1$ and P$_2$ are orthogonal to each other. If stress is not applied to the plate T, it remains transparent and appears dark because P$_1$ and P$_2$ block out the incident light; otherwise T has stress birefringence and an interference fringe appears on it. When the light enters T, only the incident light can pass through the plate as two polarized lights separated in the directions of the principal stress σ_1 and σ_2. If we let δ be a phase difference between the two rays after passing the plate, δ is experimentally given as:

$$\delta = \frac{2\pi t C}{\lambda}(\sigma_1 - \sigma_2), \tag{1}$$

where λ denotes the wavelength of the incident light, t the thickness of the plate, and C the stress–optic coefficient. When we implemented this formula, we assumed that t and C are fixed to 0.5 mm and $72 \times 10^{-13}\ \mathrm{cm}^2/\mathrm{dyn}$, respectively, and changed the principal stress. Moreover, the intensity I of this model is given by:

$$I = A^2 \sin^2 2\phi \sin^2 \frac{\delta}{2}, \tag{2}$$

where A denotes the amplitude of the incident light, and ϕ the angle between the direction of the vibrating surface and that of the principal axis. Substituting Equation (1) into Equation (2), the intensity changes depending on the wavelength. We estimate the intensity for each wavelength, multiply it by RGB values corresponding to the wavelength, and add the multiplied RGB values to the pixel value to achieve the color variation. We have empirically used 40 wavelength samples for the current implementation to balance the execution time with the sharpness of the output image. In our method, we rely on two–dimensional photoelasticity theory and thus do not leverage any raytracing algorithm.

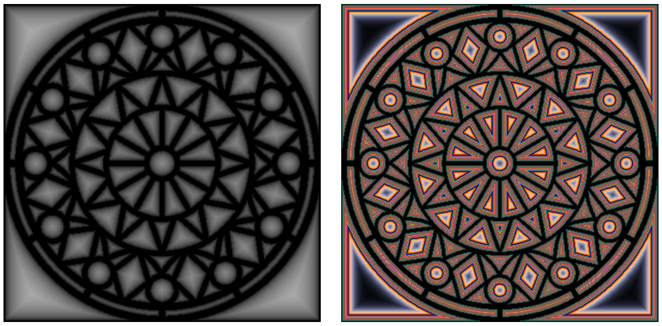
(a) (b)

Figure 3: The distance map (a) on GPU implementation and the result (b) using the map.

3 RESULTS AND FUTURE WORK

We implemented the present method on Visual Studio 2017 with an Intel Core i7-4770 3.40GHz CPU and 12GB RAM. **Figure 1** exemplifies that our method can reflect varying stress distributions to generate different interference fringes along the contours in the same input image. The image size is 400 × 400 and the current code took 51 seconds to generate one output image.

We also implemented the drawing process on a GPU to accelerate our method. The process is suitable for GPU implementation because of its recursive nature with respect to pixels and wavelengths. Additionally, we modified the way to estimate virtual stress for the GPU version. We set virtual stress field considering distances from all pixels to all contours for the CPU implementation, while we retained the distance from each pixel to the nearest contour and estimated the virtual stress to make the process on a GPU succinct. **Figure 3** shows the minimum distance map and the result on a GPU. With the GPU version, the output image has its own contrast, which the CPU version cannot generate. Nonetheless, the input image having the same resolution can be processed with a GPU in 12 seconds, which is more than four times faster than with a CPU.

The proposed method is quite simple, but could pave the way to a new style of generative art based on photoelasticity. As a future work, flexibility of parameters should be explored. In our method, the stress–optic coefficient and the spectrum of the incident light can be dealt with as parameters, and thereby making more diversity of designs by varying these variables. Additionally, further acceleration will be necessary for interactive response.

REFERENCES

M. Buβler, T. Ertl, and F. Sadlo. 2015. Photoelasticity Raycasting. *Computer Graphics Forum* 34, 3 (2015). https://doi.org/10.1111/cgf.12626

A Orzan, A Bousseau, H Winnemöller, P Barla, J Thollot, and D Salesin. 2008. Diffusion Curves: A Vector Representation for Smooth-Shaded Images. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)* 27, 3 (2008). http://dx.doi.org/10.1145/1360612.1360691