Painting With DEGAS

(Digitally Extrapolated Graphics via Algorithmic Strokes)

Steve Caruso, MLIS*
Raritan Valley Community College
Computer Science & Visual and Performing Arts
Branchburg, New Jersey
steven.caruso@raritanval.edu

Figure 1: "Nayla" (2018)'s reference image at original size (7 megapixels), its finished DEGAS painting, and an excerpt from the extrapolated painting (250 megapixels). If printed out at 300dpi, the extrapolated image would be roughly 46x61 inches – or a large gallery-sized canvas.

CCS CONCEPTS

• **Computing methodologies** → **Non-photorealistic rendering**; **Computational photography**; **Image processing**;

KEYWORDS

Non-photorealistic rendering, HTML5, JavaScript, 3D printing

ACM Reference Format:
Steve Caruso, MLIS. 2018. Painting With DEGAS: (Digitally Extrapolated Graphics via Algorithmic Strokes). In *Proceedings of SIGGRAPH '18 Posters.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3230744.3230777

*Also with Rutgers University, School of Communication & Information.

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

ACM ISBN 978-1-4503-5817-0/18/08.
https://doi.org/10.1145/3230744.3230777

1 INTRODUCTION

Most photograph-to-oil-painting algorithms are based off of the techniques described by [Litwinowicz 1997] and improved upon by [Hertzmann 2001]. They are essentially fully-automated processes which place strokes at random, choosing stroke orientations to follow local gradient normals. These strokes are built up over several layers, each layer in a decreasing order of stroke size – painting broad strokes and then filling in details. Outside of the initial chosen parameters and some masking considerations, the user has little agency in how the algorithm chooses stroke placement, nor has the ability to make direct changes or touch-ups until every stroke is laid down on the canvas – essentially it behaves like an "image filter" applied as one would adjust contrast or add texture.

DEGAS, however, is a novel way of transforming a photograph into an oil painting by treating the transformative algorithm as a tool or "brush" in a conventional drawing program. This process involves repeatedly drawing brushstroke outlines (whose orientations are chosen by a least-color-variance method) over the user's direct input, while giving the user direct access to the algorithm's parameters and the ability to alter them in real time. At its heart

this is a process-oriented approach (following the general stroke-by-stroke technique used by Impressionist painters) rather than a product-oriented approach like a neural network (such as Ostagram [Morugin 2015]) – which produces patterns visually, but not the choices behind them – or a physical viscosity simulation like Adobe Wet Brush [Chen 2016] – which simulates the fluidity of paint. Where prior techniques required strong hardware, DEGAS is computationally inexpensive enough to run on a mobile web browser on a modern consumer-grade tablet computer with JavaScript and HTML5 Canvas capabilities.

2 PROCESS & ALGORITHM

Figure 2: A memory dump snapshot representing a single stroke placement; size=100x100px, angles = 20 from 1-360°.

When a brush stroke outline is placed on an image, its center is chosen at random within the scatter zone adjacent to the user's input coordinates and the color is sampled from that center point. The portion of the photograph that the brush would overlap is loaded into a comparison buffer, and the stroke at each possible angle is rendered against the buffer and assigned a color variance index, based upon its total difference from the original image in either Euclidean or Manhattan space. All of the final indices are compared, and the lowest index (i.e. the least color variance) "wins." For sampled brush strokes, with all of their texture and imperfections, this technique gives much better results than local gradient normal calculations. The stroke is then rendered on the canvas in the chosen orientation and its information (coordinates, color, orientation, etc.) is stored in a strokes array for later reference. Strokes can be placed singularly, or rapidly while the user clicks or taps and holds.

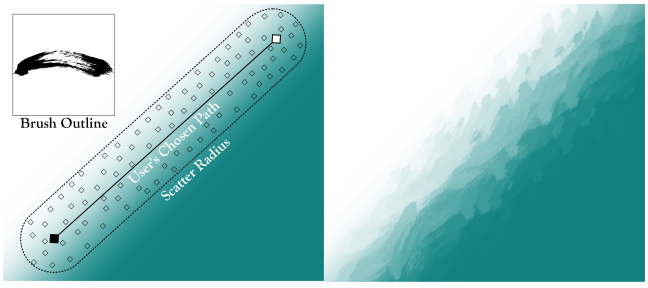

Figure 3: An example of painting over a color gradient to show scatter and color sampling.

3 EXTRAPOLATING PAINTED STROKES TO A HIGHER RESOLUTION IMAGE

Since the painting is now represented in memory by an order of layered strokes (rather than just a bitmap) the painting can be scaled up beyond the input image's original resolution, provided that the stroke outlines, themselves, are of sufficient resolution. In practice, consistently painting and extrapolating 7-megapixel images with 512x512 pixel stroke outlines to over 100 megapixels (on desktop systems) has been easy with very high quality results. However, the current tested maximum, as shown in Figure 1, took a 7-megapixel overpainted selfie and successfully extrapolated it to a quarter-gigapixel (~250-megapixel) image with no practical loss in resolution.

4 EXTRAPOLATING PAINT THICKNESS FOR 3D EFFECTS AND PRINTING

Strokes can also be painted in white at low opacity against a black background to quickly generate a height map of the paint thickness on the canvas. Once a height map is generated, a normal map can be derived and used in post-process to shade the painting's brush strokes in 3D. The shadows add a palpable sense of depth and realism to the final image.

From there, the height map can also be used to slice the painting up into a number of physical layers for 3D printing. A system such as a UV-ink printer would be an ideal choice – and was the printing technique used in the The Next Rembrandt project (a neural-network driven system that generated height maps based upon 3-dimensional input of Rembrandt's work [Flores and Korsten 2016]). Multiple layers of UV ink can be built reliably up on top of each other to accumulated depths up to a quarter of an inch – provided that registration and calibration are sufficiently exact. This would allow a painting created in DEGAS to be realized as a 3-dimensional gallery-quality artefact that further blurs the line between physical and digital.

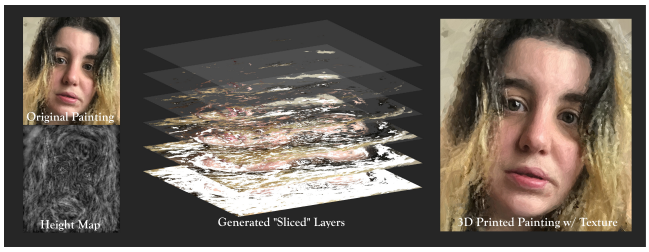

Figure 4: Example of slicing an image into printable layers (6) based upon its height map (>20 layers would be optimal). A UV ink printer can then be used to print out each layer onto a canvas or other suitable surface.

REFERENCES

Zhili Chen. 2016. *Adobe WetBrush.* https://www.youtube.com/watch?v=k_ndr3qDXKo

Emmanuel Flores and Bas Korsten. 2016. The Next Rembrandt. http://www.nextrembrandt.com/

Aaron Hertzmann. 2001. *Algorithms for rendering in artistic styles.* Ph.D. Dissertation. https://cs.nyu.edu/media/publications/hertzmann_aaron.pdf

Peter Litwinowicz. 1997. *Processing images and video for an impressionist effect.* Association for Computing Machinery, 407–414. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.6457&rep=rep1&type=pdf

Sergey Morugin. 2015. http://www.ostagram.ru/,https://github.com/SergeyMorugin/ostagram