Vector Driven 2.5D Printing with Non-Photorealistic Rendering

Paul O'Dowd, Carinna Parraman, Mikaela Harding

Abstract

This paper documents the development of a low-cost 2.5D printing machine. The machine is being developed specifically to consistently print combined colour and texture on to 2D surfaces. The proposal is not so much to produce a painting machine in emulation of the act of human expression, rather, to produce a digital machine capable of consistently producing physically textured prints. Therefore the research also looks to develop accompanying software and data representation for a cohesive workflow toward future 2.5D printing technology and its creative applications.

Introduction

This paper documents the development of a 2.5D printer and the associated software. The research investigates the ability to physically print combined colour and texture onto a 2D substrate. The motivation is to explore human-analogous gestures for medium deposition, toward a digital machine capable of physically reproducing painterly styles and to produce visual effects which exploit the character of materials used. Our inspiration is the work of old masters of painting, able to describe complex visual details with relatively few strokes[17].

To develop and investigate such a machine requires data to drive its operation. Based on available resources the digital image (jpg, png, etc.) has been selected as the research input. From the digital image a software process generates an interpretation as a composition of three dimensional vectors. These vectors are then transformed into operational instructions for the 2.5D printing machine. This paper documents the machine and software process as the basis for future work.

To generate vectors from a digital image, Non-Photorealistic Rendering (NPR) and its subset Stroke Based Rendering (SBR) have been studied. NPR encompasses a body of computational techniques to produce synthetic figurative or abstract art[9], as well as alternative functional rendering for applications such as Computer Aided Design [8]. The principle output of NPR algorithms has been images rendered to a digital display (a computer monitor). Whilst NPR and SBR methods often incorporate sophisticated models and simulations of mediums[7][1][23][5], implements[3] and substrates[2], the final rendered product flattens any layers and depth into a 2D render for the digital display.

The presented research adapts and applies an NPR technique to produce actual paintings through a vector driven 2.5D printer. Often NPR techniques are designed to emulate famous painterly artist styles such as Matisse, Van Gogh, Seurat [24], Kandinsky [25]; as well as genres of mark making including calligraphic[16][20], line drawings[18][12], pen and ink [22], hatching[11], spot colour[19], watercolour[5][11], op-art[12], and anthropo-centric expression such as machine-vision based emotional awareness[4].

We hypothesise that these prior works, whilst authored to generate data for a 2D digital display, may encapsulate useful information within their processes for machine operation. A key challenge in adaptation is that a physical machine is constrained to make a time-ordered progression of marks, and must work with the limitations of physical materials. Mistakes can not be easily erased, and unlike purely software implementations there are no convenient filters to blend or rearrange the order of layers as a post-process. This paper documents the development of a machine that in the future could accept varying painterly styles of reproduction from software such as the NPR and SBR examples given, or otherwise operate from image file formats that store printing data as 3D vector compositions rather than pixels.

Painting & Drawing Machines

This paper advances the idea within digital printing that it is possible to render simultaneously both colour and texture information through gestural based deposition. This departs away from the common progressive-scan technologies such as inkjet or electrostatic, or stepwise decomposition methods like four-colour separation and half-toning. This paper concerns the ability to consistently print reproductions, rather than developing a machine imbued with creativity to generate unique works of art. The research contributes towards a 2.5D printing machine that operates on the basis of vectors and direct write technology, normally associated to additive layer manufacturing which deposit, fuse or bind materials along paths of object geometries.

There is a history of computer controlled machines able to create expressive marks. Most famously, the artist Harold Cohen has developed algorithmic abstract art tied to physical realisation by various machines, known as Aaron [10]. More contemporary, *eDavid*[15][6] is an industrial robotic arm programmed to simulate a human painting process, incorporating a visual feedback system to iteratively analyse, generate and place brush strokes marks to approximate a source image. Similarly, Kudoh et al[14] utilise an advanced humanoid robot, with multi-sensory feedback, to further model how humans paint as a physical iterative process. *Paul*[21] is a machine which incorporates an expert model of sketching methods, and is a robotic embodied emulation of the authors own artistic drawing style.

The above examples are evaluated as a whole by how well they appear painterly or stylised in their output. The presented research makes a distinction in the specific emphasis of approach; to develop a machine capable of producing texture in the added dimension in the form of relief on the 2D surface (2.5D). The proposal is not so much to produce a painting machine in emulation of the act of human expression, rather, to produce a digital machine capable of consistently producing textured prints. Therefore the research also looks to develop accompanying software and data representation to produce a cohesive workflow toward future 2.5D printing technology and its creative applications.

Methodology

The developed methodology operates from a digital image and derives further information to generate brush strokes. Figure 1 is used throughout this article as a point of discussion.

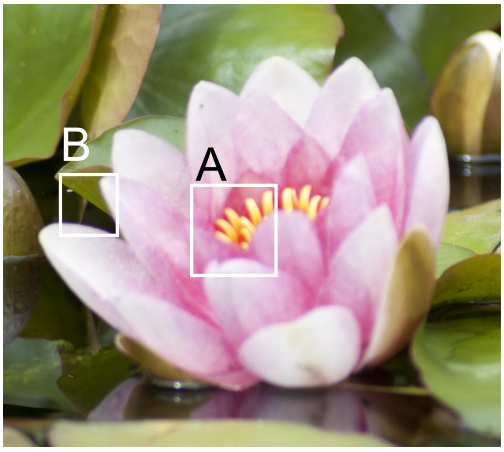

Figure 1. *The digital source image used as discussion for this methodology, with areas of interest annotated as A and B.*

The source image is used through the methodology as a reference for original colour information. From the source image, texture directionality and edge strength are extrapolated and stored per pixel. A segmentation of the image is generated using a combination of the colour information, texture directionality and edge strength information. Lastly, the segmentation, texture directionality and edge strength maps are used to generate brush strokes as vectors to send to the 2.5D printing machine. These process are given in overview in figure 2, and are detailed in the following sub-sections.

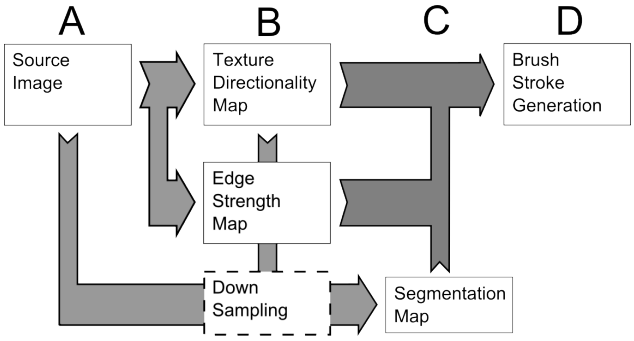

Figure 2. *An overview of the brush stroke generation process. At (A) a digital image as a colour refence and pixel position, (B) an image convolution and colour difference are used to derive texture directionality and relative edge strength, (C) colour information, edge strength and texture directionality are used to generate a spatial segmentation of the source image - an optional down sampling process may discard information from A and B prior to segmentation, (D) information at steps B and C inform the generation of brush strokes to print.*

Once the processes for segmentation and brush stroke generation have completed once, a render is presented through a graphical user interface (GUI), shown in figure 3. The GUI provides access to the underlying variables and allows the user to re-segment the image, or alter stroke parameters per segmentation to generate new brush strokes. The segmentations are presented to the user as layers to print, ordered by edge strength (highlight priority). The user is able to make their own creative interpretation of the image, and the order layers are printed, which can be consistently physically realised as editions by the 2.5D machine. The GUI provides calibration for the physical machine and operation controls.

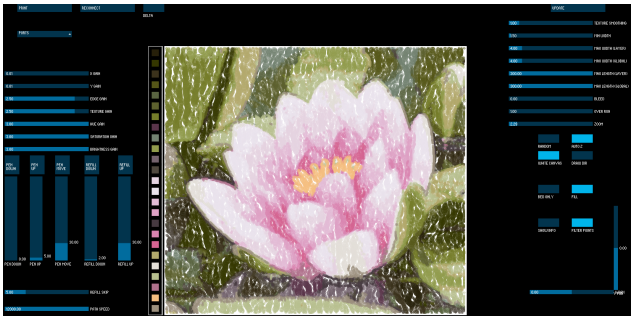

Figure 3. *A screenshot of the graphical user interface. The render can be inspected per layer and at differing resolutions. Buttons and sliders around the render expose algorithm variables, machine calibration and machine operation controls.*

2.5D Printing Machine

A low cost wood router kit with 3 degrees of freedom (x, y and z) has been modified to carry a paint brush (or other deposition tool). The machine has a printable area of 310 by 260mm and a range of 40mm vertically. RepRap 3D printer electronics the machine accepts the GCode protocol to determine movements and speed of operation. Currently the machine does not have any sensors other than mechanical switches to home the coordinate system. The machine does not have automated colour changing or a continuous loading mechanism for the brush. Instead a human operator mixes a colour manually, from which the printer is programmed to return to the pot location and reload the brush after a number of applied strokes.

Edge Detection and Texture Directionality

Edge detection and Texture Directionality are key components of the presented methodology. Edge detection and texture directionality are determined through the image convolution and colour difference calculations originally presented Kasao et al[13], with the exception that HSV (hue, saturation, brightness) has been used instead of CIELAB colour-space, and the texture angles determined are not discrete whole number values in range [0:15] and instead retain continuous value in the range $[0{:}\pi]$. The relevant equations are stated here as adapted for clarity:

$$C_{ij} = P_{ij} * M_{3-1} \tag{1}$$

$$L_{ij} = P_{ij} * M_{3-2} \tag{2}$$

$$M_{3-1} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, M_{3-2} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{3}$$

Where C_{ij} is the result of the image convolution matrix M_{3-1} about the pixel P_{ij} at position i, j within a 2D source image, and

L_{ij} is the result of the image convolution matrix M_{3-2} about the pixel P_{ij}.

From equations 1, 2 and 3, equation 4 is used to determine texture directionality (T_{ij}) of a pixel at position i, j within a 2D image and equation 5 is used to determine edge strength (E_{ij}) of a pixel at position i, j within the a 2D image.

$$T_{ij} = tan^{-1}(\frac{L_{ij}}{C_{ij}}) + (\frac{\pi}{2}) \tag{4}$$

$$E_{ij} = log(L_{ij}^2 + C_{ij}^2 + 1) \tag{5}$$

Edge strength and texture directionality values are then directly associated per pixel in the image, later used to segment the image and to derive brush strokes.

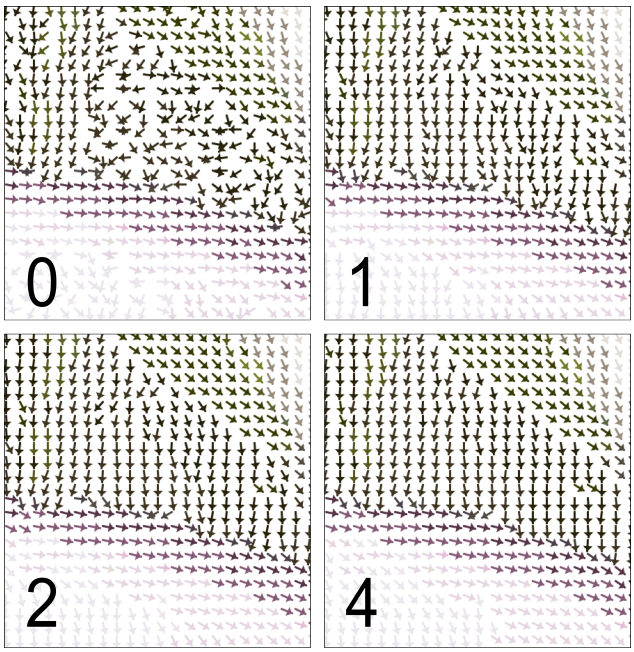

Figure 4. *Four examples of derived texture directionality, each marked with the radius of adjacent neighbouring pixels included for an average directionality weighted by edge strength.*

Down Sampling

The current methodology operates at a resolution of one pixel per mm relative to the working envelope of the 2.5D printer. The brush as a deposition tool has not been effective at sub-millimetre resolution. Therefore a high-resolution digital image (dots-per-inch) has a redundancy (excess) of information in machine-reproducable pixels per mm. However higher resolution images are desirable as they are more data informative for the algorithms to discern texture. For the later segmentation process it computationally efficient to selectively discard pixel data.

The vector and motion emphasis of this research takes priority over colour. Therefore the down sampling algorithm operates on the edge strength information of the image as the basis to retain texture directionality, and lastly colour. For each new pixel in the down sampled image, the pixel with the strongest edge strength from the region of redundant pixels is preserved. The texture directionality and colour information is copied without modification. As a result of selectively down sampling by edge strength colour information at this intermediary stage in the whole process appears noisy, as shown in figure 5.

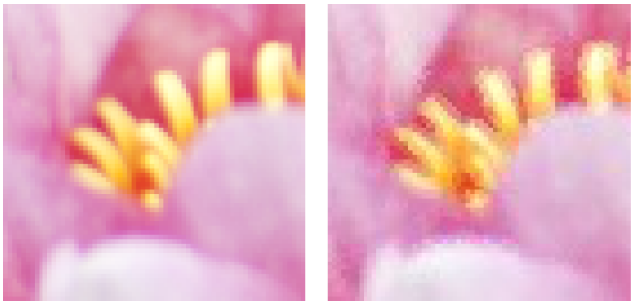

Figure 5. *Example output from down sampling. Left, image selection rescaled using Cubic Interpolation through GIMP [2] software. Right, the same image selection down sampled by preserving pixels by edge strength, exhibiting noisy colour information.*

K-Means Segmentation

A k-means clustering algorithm has been implemented based on the work of Kasao et al[13]. K-means clustering progressively divides a dataset, in this case pixels of a digital image, into a predetermined number of segmentations. The algorithm generates segmentations that are mutually exclusive image regions analogous to a paint-by-numbers template. The algorithm determines segmentations through a combinatorial evaluation of pixel spatial position in the image, colour, texture directionality and edge strength. The segmentations are not strict contiguous spatial mappings and instead group pixels of specifiable characteristics from the source image. Figure 6 shows figure 1 after segmentation into 25 regions, and thus 25 colours. Note that the coloured segmentations vary in total size, and can also be distributed across the image (non-contiguous).

At initialisation a fixed number of segmentation regions are evenly mapped in a grid like manner across the source image. Each segmentation is classified by the mean properties of the pixels assigned to it. In each iteration, individual pixels are compared against the mean pixel properties for each segmentation, and reassigned to the closest matching segmentation. As the algorithm progresses the orderly spatial mapping of segmentations degenerates with respect to the weight of the other evaluated pixel properties.

At each iteration, the mean properties of a whole segmentation alters through the gain or loss in assignment of pixels. The progression of the algorithm is monitored by how many pixels are reassigned at each iteration. When no pixels are reassigned the algorithm has determined an absolute segmentation of the dataset. In other words; every pixel, by it's evaluated properties, would not better fit an alternate segmentation. It is possible that the algorithm oscillates between an unresolvable set of segmentations, therefore the user is able to monitor and abort the process via the user interface.

[2] http://www.gimp.org

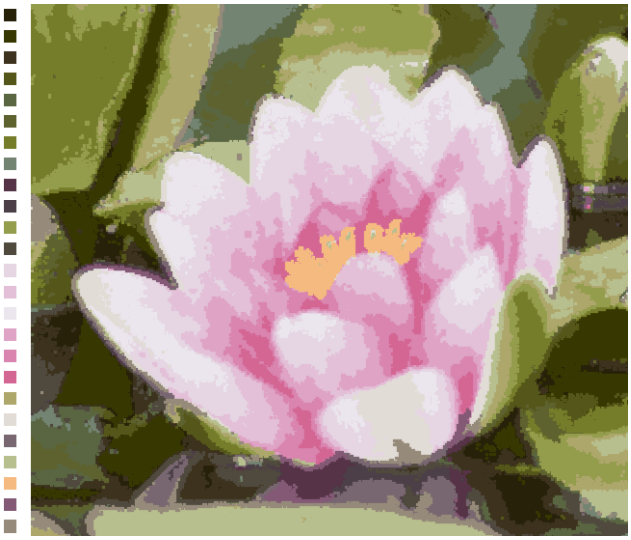

Figure 6. *The source image segmented into 25 regions, with average colour per segmentation on the left.*

A pixel is evaluated against a selected segmentation by the sum of Euclidean distance of six variables; x coordinate distance between a pixel and the segmentation centroid x coordinate (D_x), y coordinate distance between the pixel and the segmentation y centroid coordinate (D_y), a pixel edge strength (E_{ij}) versus the segmentation mean pixel edge strength (D_e), a pixel texture directionality (T_{ij}) versus the segmentation mean texture directionality (D_d); and the distance in colour space between a pixel hue, saturation and brightness value, against the average hue, saturation, and brightness of the segmentation (D_h, D_s, and D_b respectively).

The calculated values D_x, D_y, D_e, D_d, D_h, D_s and D_b are multiplied by gain values set at initialisation to provided weighted values, W_x, W_y, W_e, W_d, W_h, W_s, W_b respectively. By multiplication, gain values of less than one reduce the influence of a property, whilst gain values larger than one increase influence. The gain values are fixed for the whole image (not varied per pixel calculation).

The gain values are exposed via the user interface software allowing a user to preferentially influence the segmentation behaviour. For example, an image can be segmented with little regard for pixel spatial relationship (gain for D_x, D_y as small values, such as 0.01), and high regard for texture directionality and edge strength (gain for D_e, D_d as large values, such as 2.5). The following calculation therefore occurs between each pixel and every segmentation per iteration of the algorithm:

$$D_{ps} = W_x + W_y + W_e + W_d + W_h + W_s + W_b \tag{6}$$

Equation 6 is therefore used to determine if a pixel remains with the current segmentation assignment or should move, by selecting the minimimal distance value(D_{ps}).

Stroke Generation

The stroke generation process defines 3D vectors which the 2.5D printer will follow to deposit paint. There are several constraints to the generation of stroke marks which are customisable through the user interface. Balancing the complexity of these variables has been left to a human operator for the current research.

The maximum possible length of a single brush stroke is dependent on the actual paint loading and carrying capacity of the real brush, the viscosity of the paint medium, the paint run-off character of the brush to substrate, and whether the user wishes the stroke mark to run dry (i.e. a dry-brushing technique).The minimum and maximum width of the stroke mark is dependent on the possible radial displacement of bristles of any given brush. The variation in vertical height of a stroke mark across its length is dependent on the reach of the bristles, as well as the desired aesthetics of the mark making. Whilst these aspects can be represented virutally through the interface, the human user must calibrate their expectations via familiarty with the printing process.

In overview, stroke marks are determined per segmentation and generated sequentially to form a list. The stroke generation algorithm populates each segmentation region with brush strokes independently, allowing for differing character. The number and density of strokes is dependent on the user defined stroke length and width characteristics.

Stroke generation can either happen at random points within the segmentation, or ascend edge strength to leave highlighting marks to last. After the generation process, the list of stroke marks can be also be sorted spatially to reduce the travel time of the printer brush. These are presented as variable options to in order to not interfere with the creative process and character of the output.

When generating brush strokes, all pixel locations within the segmentation are potential location candidates from which to begin a mark. Once a stroke mark has been made, pixels that have become obscured under the width and length of the generated mark are removed from list of possible candidates for future iterations.

Two variables, ‘bleed’ and ‘over-run’ are set in units of millimetres and allow the operator to determine whether stroke marks can interfere spatially, or continue in length beyond the segmentation region. With bleed set to 0, stroke mark widths are limited so that they do not spatially interfere with existing marks. With over-run set to 0, the stroke mark length progression will be terminated if the mark leaves the segmentation region. In this way, the generated stroke length and width may not meet the maximum length and width set by the user.

With these variables in mind, strokes are generated in the following way. A candidate point is chosen within the segmentation region. The width of the stroke is increased until the maximum width is reached, or the width interferes with another stroke (moderated by bleed). The stroke generation then reads the texture directionality value for the pixel location, and increments position in that direction. The process repeats for each new position until the maximum stroke length is reached, or the mark occludes another mark or segmentation region. This process happens iteratively until there are no remaining candidate points for stroke marks. Figure 7 illustrates the result of the generation process, incorporating variation in width and length of marks made.

A final variable, ‘texture smoothing’, allows for the attenuation of the texture directionality value read per position during stroke generation. For each position, texture smoothing sets the radius of adjacent pixels to include in an average of directionality. Each included pixel directionality is weighted by the pixels edge strength relative to the segmentations maximum edge strength. This average texture directionality is only computed from pixels

Figure 7. *Region A magnified, brush strokes generated from the source image. Solid lines represent trajectory and length whilst circles are indicative of the stroke width along the path.*

within the same segmentation. Set at 0, texture directionality is only drawn from the underlying pixel. The effect of this is shown in figure 4.

Outcomes

The interpretation of the waterlilly photograph into a vector based image has the following example characteristics. For an print of 310x260mm, minimum stroke width of 1mm and maximum width of 20mm, the image has a total of 4826 brush stroke marks, the colour distribution is shown in figure 8. The highest number of marks (434) is within layer 13, constituting the light pink body of the petals. The lowest number of marks (24) is the fine edge highlight of the petals coloured purple, visible in the left most petal of figure 10. Figures 12, 11, and 9 provide examples of paintings produced via the 2.5D printing machine and software.

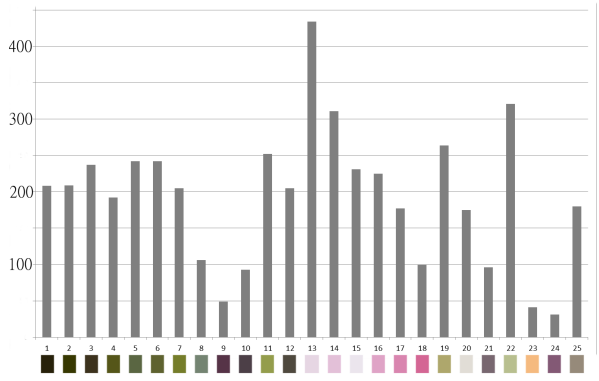

Figure 8. *After segmentation and stroke generation: a graph of the number of strokes (y axis) per layer (x axis), with associated colours per layer.*

Conclusions

This paper has documented the work to date on the development of a 2.5D printing machine. The paper documents the software processes that are used to provide operational data for the machine. The software allows a user to alter the underlying algorithm variables to produce a variety of expressive prints, which are able to be consistently physically realised by the 2.5D hardware. Future work will look at modelling mediums, substrate and

Figure 9. *A comparison of a watercolour (left) and acrylic (right), with the selective omission of several layers for artistic effect.*

Figure 10. *A software render of all generated brush strokes. Note wide marks within leaf elements, fine strokes for flower stamen.*

Figure 11. *Painting with acrylic and sparse marks.*

Figure 12. *Painting with acrylic and an oversized brush.*

stroke parametrisations, as well as improving the capabilities in hardware, and formal analysis of outputs with regard to texture reproduction.

References

[1] William Baxter, Jeremy Wendt, and Ming C. Lin. Impasto: A realistic, interactive model for paint. In *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering*, NPAR '04, pages 45–148, New York, NY, USA, 2004. ACM.

[2] Nelson S.-H. Chu and Chiew-Lan Tai. Moxi: Real-time ink dispersion in absorbent paper. *ACM Trans. Graph.*, 24(3):504–511, July 2005.

[3] N.S.H. Chu and Chiew-Lan Tai. An efficient brush model for physically-based 3d painting. In *Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on*, pages 413–421, 2002.

[4] Simon Colton, Michel F. Valstar, and Maja Pantic. Emotionally aware automated portrait painting. In *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts*, DIMEA '08, pages 304–311, New York, NY, USA, 2008. ACM.

[5] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. Computer-generated watercolor. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 421–430, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[6] Oliver Deussen, Thomas Lindemeier, Sören Pirk, and Mark Tautzenberger. Feedback-guided stroke placement for a painting machine. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, CAe '12, pages 25–33, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[7] S. DiVerdi, A. Krishnaswamy, R. Mech, and D. Ito. Painting with polygons: A procedural watercolor engine. *Visualization and Computer Graphics, IEEE Transactions on*, 19(5):723–735, May 2013.

[8] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 447–452, New York, NY, USA, 1998. ACM.

[9] Paul Haeberli. Paint by numbers: Abstract image representations. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 207–214, New York, NY, USA, 1990. ACM.

[10] Cohen Harold. Artist webpage, portfolio and bibliography. http://www.aaronshome.com/aaron/index.html, August 2015. Accessed: 2015-12-1.

[11] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 327–340, New York, NY, USA, 2001. ACM.

[12] Tiffany C. Inglis, Stephen Inglis, and Craig S. Kaplan. Op art rendering with lines and curves. *Computers & Graphics*, 36(6):607 – 621, 2012.

[13] A. Kasao and M. Nakajima. A resolution independent nonrealistic imaging system for artistic use. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 358–367, 1998.

[14] Shunsuke Kudoh, Koichi Ogawara, Miti Ruchanurucks, and Katsushi Ikeuchi. Painting robot with multi-fingered hands and stereo vision. *Robotics and Autonomous Systems*, 57(3):279 – 288, 2009.

[15] Thomas Lindemeier, Sren Pirk, and Oliver Deussen. Image stylization with a painting machine using semantic hints. *Computers & Graphics*, 37(5):293 – 301, 2013.

[16] Xie Ning, Hamid Laga, Suguru Saito, and Masayuki Nakajima. Contour-driven sumi-e rendering of real photos. *Computers & Graphics*, 35(1):122 – 134, 2011. Extended Papers from Non-Photorealistic Animation and Rendering (NPAR) 2010.

[17] Carinna Parraman. The visual appearance and surface texture of materials according to the old masters. In *Proc. SPIE 9018, Measuring, Modeling, and Reproducing Material Appearance*, 2014.

[18] Paul L. Rosin and Yu-Kun Lai. Artistic minimal rendering with lines and blocks. *Graphical Models*, 75(4):208 – 229, 2013.

[19] Paul L. Rosin and Yu-Kun Lai. Non-photorealistic rendering with spot colour. In *Proceedings of the Symposium on Computational Aesthetics*, CAE '13, pages 67–75, New York, NY, USA, 2013. ACM.

[20] Suguru Saito, Akane Kani, Youngha Chang, and Masayuki Nakajima. Curvature-based stroke rendering. *The Visual Computer*, 24(1):1–11, 2008.

[21] Patrick Tresset and Frederic Fol Leymarie. Portrait drawing by paul the robot. *Computers & Graphics*, 37(5):348 – 363, 2013.

[22] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 91–100, New York, NY, USA, 1994. ACM.

[23] Songhua Xu, Haisheng Tan, Xiantao Jiao, Francis C.M. Lau, and Yunhe Pan. A generic pigment model for digital painting. *Computer Graphics Forum*, 26(3):609–618, 2007.

[24] Chuan-Kai Yang and Hui-Lin Yang. Realization of seurats pointillism via non-photorealistic rendering. *The Visual Computer*, 24(5):303–322, 2008.

[25] Kang Zhang and Jinhui Yu. Generating abstract paintings in kandinsky style. In *SIGGRAPH Asia 2013 Art Gallery*, SA '13, pages 18:1–18:6, New York, NY, USA, 2013. ACM.

Author Biography

Paul O'Dowd received his BSc (Hons) in Robotics from the University of the West of England Bristol (2008), he co-founded the company Rusty Squid Ltd.(2011) specialising in interactive technology design within the Arts. He gained his PhD in Evolutionary Swarm Robotics from the Bristol Robotics Laboratory (2012). He has since worked as a Research Fellow at the Centre for Fine Print Research (2013) at the intersection of art and technology.