

# Non-photorealistic Rendering Using Genetic Programming

Perry Barile, Vic Ciesielski, and Karen Trist

School of Computer Science and Information Technology  
RMIT University, Melbourne, 3000, VIC, Australia  
{bpasqual,vc}@cs.rmit.edu.au,  
karen.trist@rmit.edu.au

**Abstract.** We take a novel approach to Non-Photorealistic Rendering by adapting genetic programming in combination with computer graphics drawing techniques. As a GP tree is evaluated, upon encountering certain nodes referred to as “Draw” nodes, information contained within such nodes are sent to one of three virtual canvasses and a mark is deposited on the canvas. For two of the canvasses the user is able to define custom brushes to be applied to the canvas. Drawing functions are supplied with little localised information regarding the target image. Based on this local data, the drawing functions are enabled to apply contextualized information to the canvas. The obtained results include a “Shroud of Turin” effect, a “Decal” effect and a “Starburst” effect.

**Keywords:** genetic programming, non-photorealistic rendering, evolutionary computation.

## 1 Introduction

The challenge of producing photorealistic imagery has been a long term goal within the field of computer graphics[1,2]. On the other hand, non-photorealistic rendering (NPR) aims at producing images derived from disciplines such as painting, drawing, sketching, illustration and animation. The goal of NPR research in computer science is to develop algorithms and methodologies such that a computer program can emulate the work done traditionally by hand.

Our aim is to explore an evolutionary approach to non-photorealistic rendering, specifically through genetic programming. There has been a great deal of research into NPR in recent years, but few researchers have applied genetic programming to the problem. We have developed a painting program that generates trees composed of drawing commands that are applied to a canvas in order to emulate a specified target image. Each drawing command, represented as a node in the tree, deposits ink on a canvas according to local parameters defined in the node. We have developed some tree representations that utilise properties of the target image and some tree representations that do not.

We have identified that there is little work within the research community on NPR rendering using genetic programming. We provide a set of methodologies

that govern the production of non-photorealistic images using genetic programming. Moreover, we have identified four characteristics of our approach that set it apart from previous NPR work.

1. **GP approach.** While there have been some efforts in NPR research using evolutionary algorithms [3,4,5], there is very little work being conducted on NPR using genetic programming.
2. **Automatic evaluation.** Many approaches to NPR are interactive and require human input in order to determine the subjective appeal of output images. Our approach differs in the fact that the evolutionary process is not guided by human judgment, but rather by a fitness function. We are not proposing a fitness function for evaluating aesthetic appeal, but our approach allows for the emergence of computer-generated aesthetic qualities.
3. **Animated outcome.** A key feature of the evolutionary process is that it is generally interesting to observe. We believe that the journey may be just as, or more interesting, than the destination. As a consequence of this, our final output is not simply a *best* image, but an animated sequence of best images over the entire evolutionary process.

We have identified several sources in which preconceived artistic styles were emulated [2,5,6,7,8,9,10,11,12]. In other works, a novel form of rendering was discovered [1,3,4,13]. Our aim is to apply genetic programming to target images in order to discover novel program representations of such images. We hypothesize that by implementing our own approach to genetic programming, we can discover novel forms of non-photorealistic rendering. Additionally we seek to investigate the effects of applying programmatic methodologies and domain knowledge in order to promote varieties of rendering styles and to promote convergence. Our research questions are:

1. Can we discover novel forms of non-photorealistic rendering?
2. What methodologies can we employ in order to facilitate different drawing styles?
3. What types of domain knowledge can be used in order to promote faster convergence?

## 2 Related Work and Methods

In this section we first discuss work by other researchers, then we discuss the two primary methods of fitness evaluation.

### 2.1 Related Work

In [9], Haeberli suggests several techniques designed to facilitate informed heuristics in order to promote various visual effects. In [3], Lioret provides a brief survey of rendering techniques involving cellular automata, Lindenmayer systems, fractals, morphogenesis, neural networks and genetic algorithms.

In [6], Baxter et al developed a tool which models natural media in order to simulate the flow and drying of paint. Another approach to simulating natural media is [7], in which East Asian ink and watercolour techniques are simulated.

In [8], Gooch et al take an interesting approach in which they take a photograph and apply transformations based on brightness and luminance in order to produce line art portraits. In [12], Wen et al, take input images and apply a loose segmentation algorithm to them. Drawing on the canvas is achieved by consulting a database of colours and applying a set of drawing rules defined by experts. A form of line art, with colored strokes and filled areas, is achieved.

In [10], Hertzmann takes a target image and applies a transformation or effects filter to it. A learning algorithm is employed to “observe” the transformation. The algorithm is then applied to a different target image in order to reproduce the effects of the learned filter. In [13], Hertzmann defines sets of *energy functions* that allow definitions of desires for painting and their relative importance. These functions allow different styles of painting. The system consists of applying strokes using these functions and through various means of *relaxing* the brush strokes, various effects are applied to input images.

In [1], Semet et al employ a distributed agent approach in which a colony of “ants” populates the canvas and deposit marks on the canvas according to local information, such as edge-detected gradients.

In [14], Wijesinghe et al generate an animated mosaic that converges towards a target image.

## 2.2 Fitness Evaluation

Many NPR approaches require human involvement to interactively judge the aesthetic appeal of images. There are weaknesses in this approach. For instance, humans can only stay focussed for relatively short periods. However some NPR applications are specifically designed in order to function interactively with users. The IMPasTo project[6] aims at allowing users to explore the digital painting process. In the work of Semet et al.[1], the user drives the gradual development of characteristics of ant colonies in order to emulate the sketching process of starting out with rough strokes and then gradually refining the strokes.

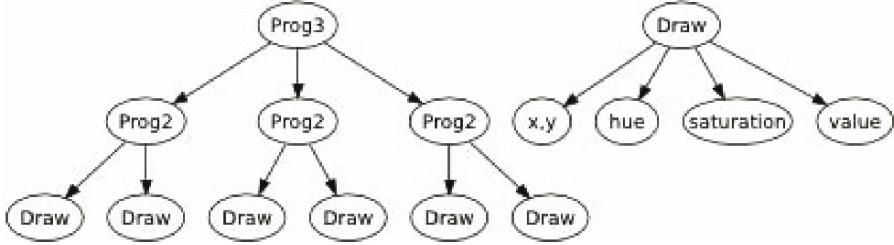
In contrast to the interactive process, many approaches employ an automatic software function to make judgements on the appeal of images[15]. Various techniques have been used to compute the fitness of an image. Most approaches (such as ours) adopt the simple heuristic of comparing an output image pixel-by-pixel to the target image. There are other approaches. In [16], Machado et al. base fitness on image complexity and uses a formula based on quad-trees in order to measure complexity. In [17], Ross et al. utilise Ralph’s bell curve model of aesthetics, based on empirical analysis of fine art, in order to evolve aesthetically pleasing images.

## 3 Our Approach

This section details our modified GP approach, then describes our specialized draw functions and GP configuration.

### 3.1 Modified GP Approach

Our representation uses GP techniques in a non-conventional manner. There are two type of function nodes: *Prog* nodes and *Draw* nodes. No values are passed up the tree. Evaluation of a GP tree results in a sequence of Draw strokes on the canvas. The GP formalism provides a convenient representation for a variable number of brush strokes and the order in which they are executed. See figure 1.



**Fig. 1.** This figure shows the relationship between Prog nodes and Draw nodes. The Draw node shown on the right of the figure would be used for issuing a drawing command at a position on the canvas, employing one input for the position and three input for each channel of an HSV colour.

We employ **Prog2** and **Prog3** nodes, which accept 2 or 3 inputs respectively. **Draw** nodes can only have terminals (random float nodes) as input. Terminals are float variables which are initialised with a value between 0.0 and 1.0 using uniform probability distribution. Prog nodes serve merely to pad the size of the tree and to spread out draw nodes. When the tree is evaluated, anytime that a Draw node is encountered, a brush stroke is made on the canvas using the information contained in each Draw node. Each canvas has its own interpretation of the information sent it by a Draw node.

We have created three canvasses used to perform rendering. Only one canvas is used at a time. We call the canvasses: “Shroud of Turin”, “Decal” and “Starburst”. The Draw node depicted in figure 1 pertains to the Starburst canvas and the Decal canvas. In the case of the Shroud of Turin canvas, the inputs to the Draw node are:  $x/y$ , *stroke length*, *stroke angle*, *grayscale stroke color*. Again, each canvas has its own interpretation of the information sent it by a Draw node. The Draw nodes themselves are generic. The input label “x,y” may be confusing and suggests that two nodes are in fact necessary, but this is not the case. Each canvas is represented as a 2D array of pixels. It is a trivial matter to represent a 2D array as a 1D pointer and determine the  $x$  and  $y$  offset of a pixel using the width of the canvas.

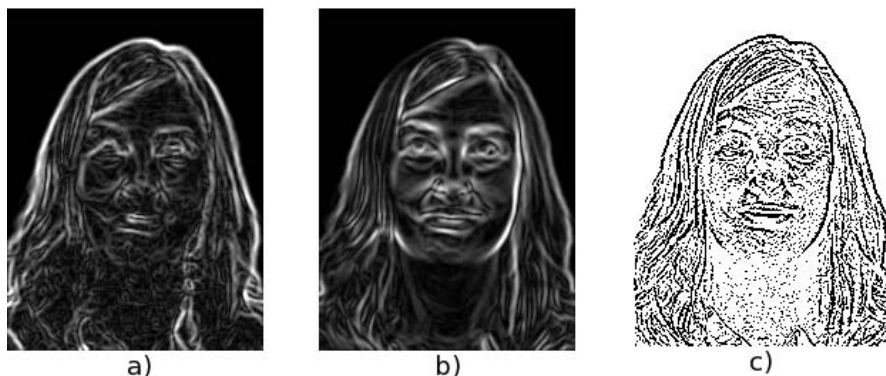
### 3.2 Draw Functions

During the course of our experiments we identified a range of input values for Draw nodes, according to a desired pictorial outcome. In addition to this, we identified some choices to be made when implementing Draw functions.

**Over-writing Previously Drawn Pixels:** What to do when about to draw a pixel that has already been drawn? Three distinct choices that were made:

1. Replace the existing pixel
2. Blend the existing pixel with the new pixel by computing their average value
3. Do nothing (preserve the existing pixel)

**Incorporating Localized Image Data in Drawing Calculations:** The target image is pre-processed by converting it from the RGB color space to the HSV color space. Gradient maps and an importance map are prepared for the target image. Examples of these maps are shown in figure 2.



**Fig. 2.** a) A saturation gradient map. b) A value gradient map. c) An importance map.

**Gradient maps:** Edge detection using a 3x3 Sobel filter is applied to both the *saturation* channel and the *value* channel of the target image. This produces a *saturation edge map* and a *value edge map*.

**Importance map:** A user can specify parts of an image considered important. Whenever a pixel is about to be drawn, six pieces of pixel data are considered: the pixel data of the pixel about to be deposited ( $P_b$ ), the pixel on the canvas about to be drawn on ( $P_c$ ) and the corresponding pixel of the target image ( $P_t$ ); the corresponding pixel in the saturation map ( $P_s$ ), the corresponding pixel in the value map ( $P_v$ ) and the corresponding pixel in the importance map ( $P_m$ ). A threshold value ( $T$ ) of 20 in a range of 0 to 255 is set. If  $P_s$  is above  $T$ , saturation matching is performed. If  $P_v$  is above  $T$ , value matching is performed. If  $P_m$  is above  $T$ , both saturation and value matching is performed. Hue matching is always performed.

**Hue / Saturation / Value Matching.** For each channel, whether it be hue, saturation or value, the following matching function is used:

1. Compare  $Pb$  to  $Pt$ .
2. Compare  $Pc$  to  $Pt$ .
3. For each channel, if and only if the channel value in  $Pb$  is closer to  $Pt$  than  $Pc$ , draw that channel value.

### 3.3 GP Configuration

The configuration for genetic programming is shown in table 1. We use Strongly Typed GP.

**Table 1.** Genetic Programming Configuration

Parameter	Value
Population Size	10
Crossover Rate	40%
Mutation Rate	50%
Elitism Rate	10%
Max. Generations	25000
Selection	Roulette Wheel
Initialization	Decimation from initial population of 100
Replacement Strategy	Generational replacement
Fitness	Pixel-by-pixel comparison
Termination	0.05 difference or max. generations reached
Functions	Prog2, Prog3, Draw
Terminal	
<i>frand</i>	random floating point number between 0.0 and 1.0

## 4 Experiments and Results

During exploration of the creative range of our system, we identified 3 pictorial styles that can be generated by applying particular image pre-processing steps and drawing functions. In all cases the inputs to Draw functions are *frand* terminals. Runs typically take several hours to complete. We detail these effects below.<sup>1</sup>

### 4.1 Shroud of Turin Effect

**Image Pre-Processing:** None.

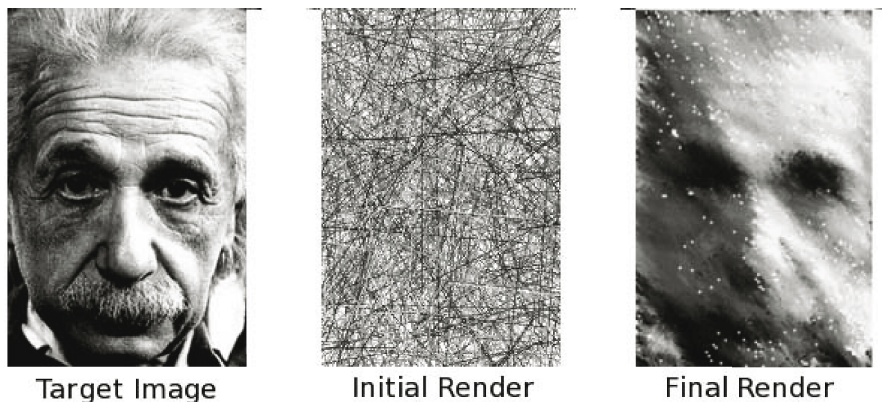
**Channel Matching:** None.

**Average Tree Size:** 12186

**Number of Draw Nodes:** 1945

<sup>1</sup> The printed format does not convey the visual richness of some of our results. Additionally our results are created as MPEG files. To view our results in full, please go to <http://www.cs.rmit.edu.au/~vc/evolved-art/npr-seal/>

**Draw Function:** This effect is so named because evolutionary sequences typically consist of a “ghostly” image gradually emerging into a “corporeal” form. The effect is best applied to grayscale targets. The Draw function takes 4 inputs and draws a straight line. The inputs define *starting coordinate*, *line length*, *line angle* and *line grayscale color*. See figure 3. As stated previously, the *x* and *y* coordinates of the starting coordinate can be computed using only one input value.



**Fig. 3.** Shroud Of Turin Effect

## 4.2 Starbursts

**Image Pre-Processing:** Saturation and Value Maps generated, Importance map specified by the user.

**Channel Matching:** Hue, saturation, value channels.

**Average Tree Size:** 1955

**Number of Draw Nodes:** 349

**Draw Function:** The effect is defined by an internal (programmer-defined) 2D array of booleans. The array consists of a set of “spokes” arranged around a central hub. The radius of the hub is programmer-defined. We set the radius to be computed as 1/20th of the width or height of the input image, whichever is smaller. The spokes are defined as true values. The entire array is processed. Any cell values in which the value is false are not drawn, hence achieving a transparent effect around the spokes. The Draw function accepts 4 inputs which define *the central pixel coordinate of each Starburst*, and the *hue*, *saturation* and *value* channels of the color to be applied by the Starburst. Each time a pixel is drawn, the color of the existing pixel is blended with the color of the current starburst. See figure 4.

## 4.3 Decals

**Image Pre-Processing:** Saturation and Value Maps generated, Importance map specified by the user.

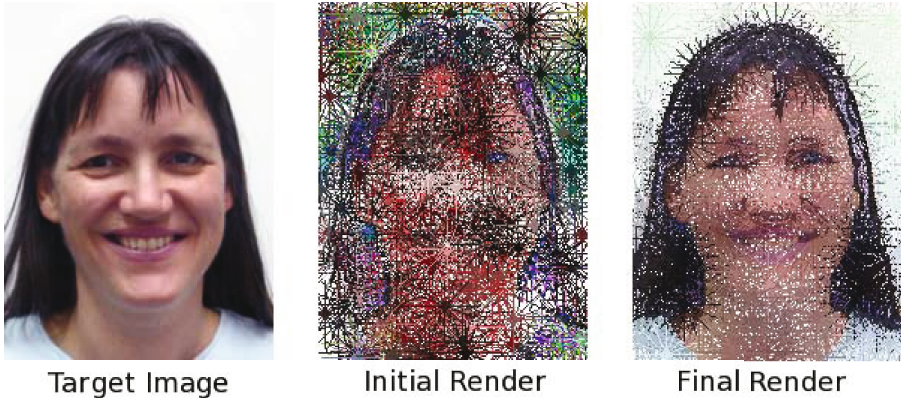


Fig. 4. Starburst Effect

**Channel Matching:** Hue, saturation, value channels.

**Average Tree Size:** 1518

**Number of Draw Nodes:** 272

**Draw Function:** With this effect, an external image (the decal), is specified by the user. The decal consists of grayscale pixels. Pixels values in the Decal image that tend toward black are drawn with greater opacity, while values that tend toward white are drawn with lower opacity. The Draw function accepts 4 inputs which define the *central pixel coordinate of each Decal*, and the *hue, saturation and value* channels of the color to be applied by the Decal. When a pixel is drawn, the color of existing pixel is blended with the color of the Decal pixel, subject to the opacity of the Decal pixel. See figure 5.

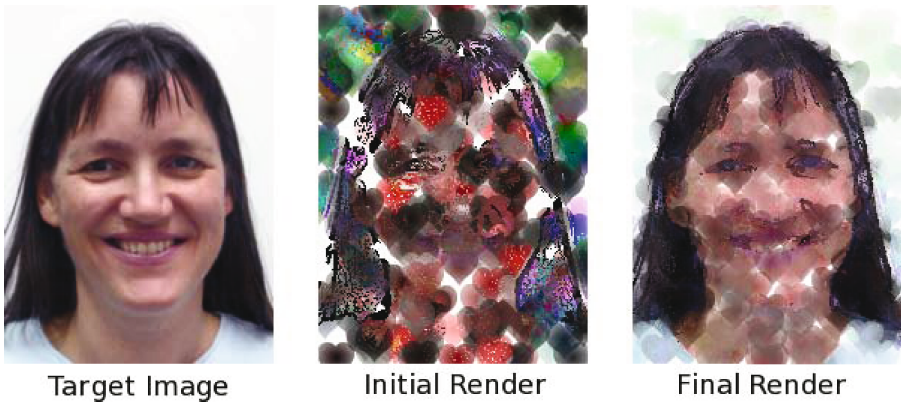


Fig. 5. Decals Effect

## 5 Conclusions and Future Work

To answer our research questions:

### Can we discover new forms of non-photorealistic rendering?

- The “Shroud of Turin” effect was discovered as a result of applying a simplistic drawing function that draws straight strokes, utilising no local image data such as areas of importance or edge maps.
- The “Starburst” and “Decal” effects employ a “paint blotch” rendering technique. The Starburst effect employs color compositing using full opacity, while the Decal effect utilises transparency while compositing.

**What methodologies can we employ in order to facilitate different drawing styles?** To achieving stylised looks, we focussed on exactly what happens when a stroke is deposited on the canvas. We explored several stroke-based techniques, but failed to achieve any effects that evoked genuine excitement from our artistic collaborators. Switching from a stroke based approach to the decal approach elicited interest from our advisors. We found that hue/saturation/value channel matching had to be employed in order to promote convergence.

**What types of domain knowledge can be used in order to promote faster convergence?** Applying channel data matching when compositing pixels speeds convergence considerably. Allowing the user to specify important areas on the target image allows the program to more accurately render important details within the image. Applying edge detection provides the program with important information regarding changes in contrast and tonality, speeding convergence.

We have identified several issues that we intend to investigate in future work. We intend to investigate methods of optimizing our generated GP structures. Furthermore we intend to investigate alternate evolutionary forms, such as linear genetic programming and grammatical evolution, in order to produce string representations of our GP trees. We also intend to investigate methodologies for evolving line drawings. Future work will also focus on incorporating aesthetic sense into the fitness function.

## References

1. Semet, Y., O’Reilly, U.-M., Durand, F.: An Interactive Artificial Ant Approach To Non-Photorealistic Rendering. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103. Springer, Heidelberg (2004)
2. Strothotte, T., Schlechtweg, S.: Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation. Morgan Kaufmann, San Francisco (2002)
3. Lioret, A.: Being Paintings. In: ACM SIGGRAPH 2005 Electronic Art and Animation Catalog (2005)
4. Sims, K.: Artificial evolution for computer graphics. In: SIGGRAPH: Proceedings Of The 18th Annual Conference On Computer Graphics And Interactive Techniques (1991)

5. Terzopoulos, D.: Artificial Life For Computer Graphics. *Commun. ACM* 42(8), 32 (1999)
6. Baxter, W., Wendt, J., Lin, M.C.: IMPasTo: A Realistic, Interactive Model For Paint. In: *NPAR: Proceedings Of The 3rd International Symposium On Non-Photorealistic Animation And Rendering* (2004)
7. Chu, N.S.-H., Tai, C.-T.: MoXi: Real-time Ink Dispersion In Absorbent Paper. In: *ACM SIGGRAPH* (2005)
8. Gooch, B., Reinhard, E., Gooch, A.: Human Facial Illustrations: Creation And Psychophysical Evaluation. *ACM Transaction On Graphics* (2004)
9. Haeberli, P.: Paint By Numbers: Abstract Image Representations. *Computer Graphics* 24(4), 207–214 (1990)
10. Hertzmann, A.: Image Analogies. In: *SIGGRAPH: Proceedings Of The 28th Annual Conference On Computer Graphics And Interactive Techniques* (2001)
11. Hertzmann, A.: A Survey Of Stroke-Based Rendering. *Computer Graphics and Applications* (2003)
12. Wen, F., Luan, Q., Liang, L., Xu, Y.-Q., Shum, H.-Y.: Color Sketch Generation. In: *NPAR: Proceedings Of The 4th International Symposium On Non-Photorealistic Animation And Rendering*, pp. 47–54 (2006)
13. Hertzmann, A.: Paint By Relaxation. In: *Proceedings of Computer Graphics International*, pp. 27–54 (2001)
14. Wijesinghe, G., Mat Sah, S.B., Ciesielski, V.: Grid vs. Arbitrary Placement of Tiles for Generating Animated Photomosaics. In: *2008 World Congress on Computational Intelligence* (2008)
15. McCormack, J.: New Challenges For Evolutionary Music And Art. *SIGEVOlution* 1, 5–11 (2006)
16. Machado, P., Cardoso, A.: All The Truth About NEvaR. *Applied Intelligence* 16(2), 101–118 (2002)
17. Ross, B.J., Ralph, W., Zong, H.: Evolutionary Image Synthesis Using a Model of Aesthetics. In: *Proc. 2006 IEEE Congress on Evolutionary Computation* (2006)