

Generation of Miro's Surrealism

Lu Xiong

School of Computer Software
Tianjin University
No.135, Yaguan Rd, Tianjin 30000, China
xionglu@tju.edu.cn

Kang Zhang

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd, Richardson, TX 75080, USA
kzhang@utdallas.edu

ABSTRACT

In this paper, we present a programmed experiment and the results on automatic generation of Joan Miro style of surrealism. Combining the artist's aesthetic theory with the authors' own understanding, the paper analyzes the characteristics of Miro's work and proposes a process modeling approach that consists of four steps: structured drawing, adaptive coloring, space filling and noise injection. The generation process is described in details and sample generated images styled on Miro's paintings are also demonstrated and discussed. By extracting and coding pictorial elements in paintings of Miro, different styled images could be generated under different sets of parameters.

CCS Concepts

• Applied computing~Arts and humanities • Applied computing~Fine arts

Keywords

Surrealistic paintings; Joan Miro; Process modeling; Generative art

1. INTRODUCTION

Joan Miro was a sculptor, ceramicist and one of the greatest surrealist artists in the 20th century. The formation of Miro style is closely related to his life experience and his strong sensibility to nature. He conveys the most powerful feelings of human beings with symbolism and a quiet natural artistic language. The symbolic and poetic paintings that came out in his thirties were eventually dubbed Miro's dream paintings [26]. In his middle age, he tried to break through the rules of reason and logic, liberate the mind of unconscious and non-logical, and probe the mystery of the invisible realms and the visual world. Miro abstracted form and structure as a point, a line, and a burst of color. Those paintings are childlike and enjoyable, his *constellations* [10] contributed to the emergence of American abstract expressionist painters.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

VINCI '16, September 24-26, 2016, Dallas, TX, USA

© 2016 ACM. ISBN 978-1-4503-4149-3/16/09 \$15.00

DOI: <http://dx.doi.org/10.1145/2968220.2968232>

With the rapid advances of modern technology and particularly digital display, digital art becomes more expressive than traditional visual art. The development of computer graphics [16] and information visualization enables computer-generated images with aesthetic significance. Modern computer technology can generate aesthetic forms of visual art [12][19][30]. Future design of information visualization would also benefit from analysis of aesthetics [29].

This paper reports a process modeling approach to automatic generation of Miro style of surrealist paintings using the Processing programming language [21]. Basic graphics algorithms and simple rendering techniques for generative art could be easily programmed using Processing [20].

The rest of this paper is organized as follows. Section 2 gives a brief review of related work. Section 3 analyzes Miro's style based on his art theory. The process modeling approach is discussed in details in Section 4. Section 5 describes the implementation of generating surrealist paintings, including algorithm design and detailed steps, together with the generation results. Finally, Section 6 concludes the paper.

2. RELATED WORK

Generative art has long been discussed and attempted by computer graphics researchers. Haeberli [9] has created abstract images using an ordered collection of brush strokes, by controlling the color, shape, size, and orientation of individual brush strokes. Impressionistic paintings of computer generated or photographic images can easily be created, and now the idea has been made as apps.

Zhao and Zhu [33] presented an interactive abstract painting system named Sisley which works upon the psychological principle. Given an input photograph, Sisley decomposes it into a hierarchy (effectively tree) of its constituent image components with interactive guidance from the user, then automatically generates corresponding abstract painting images, with increased ambiguities of both the scene and individual objects at desired levels. More recently, computer vision techniques with neural networks have been used to transform input images to specific styles of paintings [7]. All the above works require an input photograph.

Computer graphics researchers and digital artists have also used computers to generate abstract art works based on fractals without image input [1][2]. It was first proposed by Taylor [24] in 1999. Taylor [23][25] used fractals to model and analyze Jackson Pollock's dripping style of paintings and generated remarkable results. Fogleman [6] has attempted to generate Mondrian-style of

abstract paintings automatically. Their pioneering works have inspired us for the present work.

Modeling Miro's style was first discussed by Kirsch and Kirsch [13]. They had not encoded the analysis into any grammar or program but demonstrated the methods to generate new compositions in Miro's style with algorithmic descriptions. Their approach stores typical Miro shapes into a database and then manually analyzes the target composition using the stored shapes. Our approach does not require a database, rather, parameterizes and randomizes Miro shapes, with automatic generation.

More recently, Zhang and Yu [31] made an interesting attempt in automatic generation of Kandinsky abstraction paintings using Processing. Tao et al [22] generated abstract paintings in Malevich style and Zheng et al [32] proposed a layered approach to modeling Jackson Pollock's dripping style of paintings. Based on these experiences and practices, we designed a programmed experiment to automatic generate Miro's surrealistic style of paintings.

3. STYLE ANALYSIS

Joan Miro is a prolific artist of multitudinous paintings which are hard to conclude. His early modernist works show the influence of Cézanne [4][5][15], and fill the canvas with a colorful surface and a more painterly treatment than the hard-edge style of most of his later works. Starting in 1920, Miro developed a quiet precise style, picking out every element in isolation and detail, and arranging the elements in deliberate compositions. The works, including *House with Palm Tree* (1918), *Nude with a Mirror* (1919), *Horse, Pipe and Red Flower* (1920), and *The Table - Still Life with Rabbit* (1920), show clear influence of Cubism [3][8], although in a restrained way, being applied to only a proportion of the subject. In 1922, Miro explored abstracted, strongly colored surrealism in at least one painting [17]. Through the mid-1920s Miro developed the pictorial sign language which would be central throughout the rest of his career. Our research focuses on the works of Miro's middle age and analyzes the following set of paintings.

- *The Nightingale's Song at Midnight and the Morning Rain* (1940)
- *Poetess* (1940)
- *Ciphers and Constellations in Love with a Woman* (1940)
- *Women and Birds at Sunrise* (1946)

- *Women in Front of the Sun* (1950)
- *Blue III* (1961)
- *The Gold of the Azure* (1967)

Having read papers on the analysis of Miro's artistic style [14][17], and searched for images on the web, we summarize the artist's artistic features as below. His works are known as organic abstraction. Organic means that he does not completely abandon the objective of graphic arts like Mondrian [18], but chooses the characteristics of the objects in the deformation. During this period, the artist formed his own abstract pictorial sign language. These abstract pictorial sign elements illustrated in Figure 1 came from his fantasy and abstraction, and appeared in many paintings composed of different shapes, sizes, colors and numbers. To facilitate the discussion, we name these pictorial elements according to their shapes.








Star	Spiral
	
Sharp angle hourglass	Smooth angle hourglass
	
irregular circle	irregular pentagram
	
compositions of irregular circles and lines	
	

Figure 1: Typical pictorial elements

The overall color of his works is bright and trenchant. Most of his works use several specific colors, including red, blue, yellow, white and black.

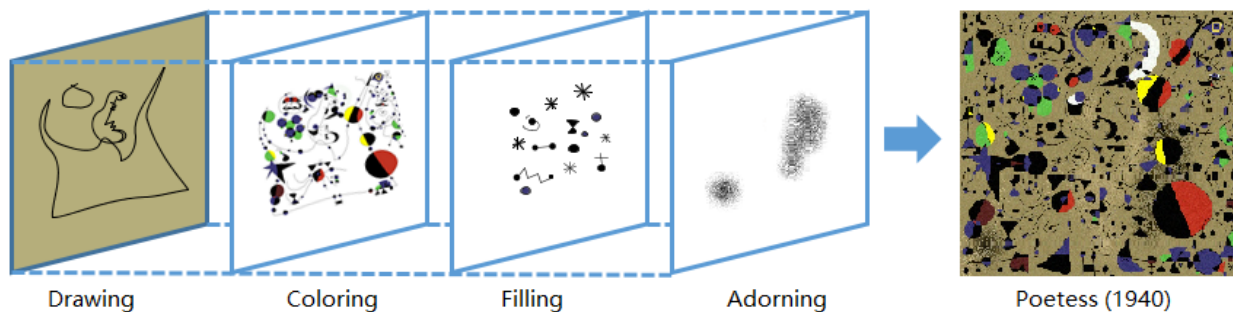


Figure 2: Process modeling approach

Color can affect mood and despite cross-cultural differences regarding what different colors meant there are cross-cultural similarities regarding what emotional states people associate with different colors [27][28]. Usually, red is the most emotional color, blue evokes broadness and peace, yellow means sun and warm, white represents nobility and holy while black delivers fear and depression. The artist communicated his emotions using these simple colors, while also achieving harmony.

Composition of surrealism paintings delivers fantastic impression through subtle combination of figurative and abstract expressions. As for Miro, he pursues liberty and harmony. This is why some of his paintings (like *Poetess* and *The Nightingale's Song at Midnight and the Morning Rain*) have more than one focus that looks chaotic but absorbing while others (like *Blue III* and *The Gold of the Azure*) are childlike but fragrant.

4. PROCESS MODELING

We start by presenting our modeling work on *Poetess* (1940, Figure 2). As discussed above, the artwork has significant features. The canvas is covered by colorful pictorial elements and curve lines. These components are not independent, the interactions between pictorial elements or between pictorial elements and lines following certain rules. Ingenious combination of several different pictorial elements creates semantic graphic objects and makes the ensembles attractive.

Inspired by layered modeling approach [32], we designed a process modeling approach after considering the features of the painting to deliver desired results. Process modeling consists of four steps, i.e. structured drawing, adaptive coloring, space filling and noise injection, as shown in Figure 2. Each step composes of rendering components of different graphical properties. The shapes of components in each step are made consistent, and distributed on the canvas with certain randomness, similar to the drawing method developed by surrealists. The complete artwork is generated after these four steps.

The details of each step will be discussed in the following subsections.

4.1 Structured Drawing

After careful observation and analysis, we separate all components into three groups according to the drawing sequence. The first group contains long curve lines. The second group consists of semantic graphic objects that are easy to recognize as special assemblies of basic pictorial elements. Many smaller but similar pictorial elements belong to the third group. The process modeling approach draws these three groups successively.

The first step, i.e. drawing process, contains background coloring and curve plotting. In the drawing process, long curve lines of the first group are drawn with certain randomness. These curve lines have a unique style and divides a pictorial element or semantic graphic object it passes through into two or more parts, which are colored differently with contrasts. Assume these long curve lines to be drawn first in the artist's painting process, we treat them as part of background which will not be changed by other components. Before drawing curve lines, we choose a single background color to cover the entire canvas and provide the fundamental tone of the artwork.

4.2 Adaptive Coloring

In the second step, i.e. the adaptive coloring process, we paint various closed elements of the second group with colors, e.g. irregular circles, irregular pentagram and semantic graphic objects

such as moon or robot. The size and position of each element are randomly generated within a pre-determined range. A major characteristic of Miro style is that the colors of an element are relevant to the element's position and its association with other elements. If the element has no association with other elements, it has one color. If an element is placed upon existing curve lines on the canvas, or intersect with other elements, it may be divided into several parts and each part is filled with a different color.

Miro's style is dominated by primary colors, i.e. red, blue, yellow, together with white and black. We also use a small amount of other colors, such as brown and green, according to the original painting.

Each semantic graphic object drawn in this step may be cut through by one or more long curve lines. The parts of the object divided by the long curve lines are filled with different and contrasting colors depending upon the current color of the object. To support this functionality, we adapt the flood fill algorithm to color the objects, and call it adaptive fill algorithm. The color to fill is determined by the existing color (which require the background color to be single) of the element and the background color. The coloring process provides the fundamental tone for the final image as shown in Figure 2.

4.3 Space Filling

The third step is to fill the space on the canvas with the elements of the third group, including stars, spirals, sharp or smooth angled hourglasses and all kinds of compositions of irregular circles and lines as shown in Figure 1. One can observe *Poetess* (1940, Figure 2, right) that it has nearly no blank space. The elements of the third group are distributed throughout the entire canvas, fulfilling the style of the artwork. These elements are small but abundant, each having a different size, color and shape; even the same styled elements do not look exactly the same.

To generate these elements, we code each styled element into a function based on mathematical calculation, sized randomly within a pre-determined given range. As these elements are small and usually single colored, we ignore their interactions with other elements and position them randomly onto the canvas.

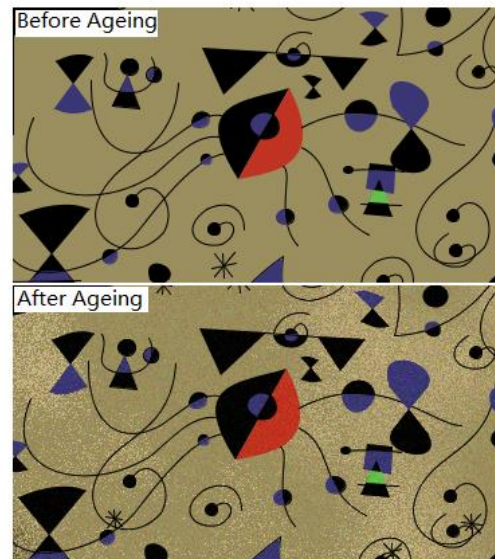


Figure 3: Comparison of before and after ageing treatment

4.4 Noise Injection

The generation process has completed ninety percent after the first three steps. The original paintings show obvious blocks of shades, due to either special paint effects or ageing. To make generated paintings more realistic, we design noise injection process as the last step. On one hand, we need to simulate shade blocks, on the other hand, we apply ageing treatment to the entire image. Figure 3 compares the effects before and after the ageing treatment. The implementation details in Processing will be demonstrated in the next section.

5. IMPLEMENTATION

We have implemented the process modeling approach in Processing to generate images of Miro's surrealistic style. This section describes the generation algorithms in more details and demonstrates the results.

5.1 Style Encoding

Based on the aforementioned analysis and design, we extract abstract pictorial elements of Miro and code them into individual functions.

Algorithm 1: Draw irregular circle

Input: **centerX:** the x-coordinates of center of the circle
 centerY: the y- coordinates of center of the circle
 radiusX: major semi-axis of circle
 radiusY: minor semi-axis of circle
 h: parameters to control curvature

```
drawRightBezier(centerX, centerY+radiusY, centerX+radiusX,
centerY);

drawRightBezier(centerX, centerY-radiusY, centerX+radiusX,
centerY);

drawLeftBezier(centerX-radiusX, centerY, centerX, centerY-
radiusY);

drawLeftBezier(centerX-radiusX, centerY, centerX, centerY+
radiusY);
```

We use Bezier function to simulate curve lines. Bezier function generates a parametric curve and is frequently used to model smooth curves that can be scaled indefinitely. By dividing the curve lines in the original painting into several segments, each implemented as a Bezier curve, and connecting the segments, we could simulate long curves. We then use the *scale* or *translate* function in Processing to change the size or position of the entire curve lines after completing the drawing.

Researchers have shown that mathematics and painting are interrelated in many ways [11]. Using mathematics, Bezier curves could also be used to simulate irregular shapes, e.g. irregular circles and sharp or smooth angle hourglasses as shown in Figure 1. Figure 4 is a schematic drawing of simulating a circle using multiple Bezier curves, by dividing the circle into four quarter arcs. Each quarter is implemented by one Bezier curve, parameterized by two endpoints *A* and *D*, and two control points, *B* and *C*. We can see that points *B* and *C* are symmetrical about line *OF*.

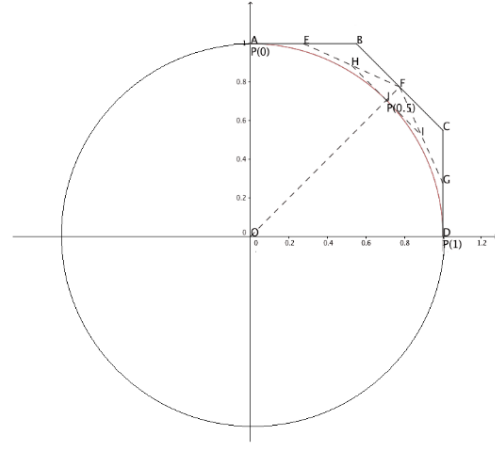


Figure 4: Using Bezier curve to simulate a circle

We introduce a parameter *h* to represent the length of line *AB* so we could get the coordinates of point *B* (*h*, *l*) and point *C* (*l*, *h*). According to the formula of Bezier curve:

$$P(t) = A \cdot (1-t)^3 + B \cdot 3(1-t)^2t + C \cdot 3(1-t)t^2 + D \cdot t^3, \\ t \in [0,1]$$

We could know from Figure 4 that when the value of *t* is 0.5, the point *P*(0.5) is the midpoint of the quarter arc *AD*. With the known coordinates of the points *A* and *B*, we could get the approximate value 0.55 of *h* by calculation. By changing the value of *h* around 0.55 we could control the curvature of the Bezier curve. When connecting four Bezier curves with different value of *h* together we could simulate an irregular circle. Algorithm 1 outlines the algorithm.

Algorithm 2: Draw star

Input: **centerX:** the x-coordinate of center of the star
 centerY: the y- coordinate of center of the star
 radius: the size of the star

```
bias1 = random(-BIAS,BIAS);

line(centerX-radius, centerY - bias1, centerX+radius, centerY
+ bias1);

bias2 = random(-BIAS,BIAS);

line(centerX-bias2,      centerY-radius,      centerX+bias2,
centerY+radius);

k = random(minK,maxK);

t1 = random(minR,maxR);

t2 = random(minR,maxR);

line(centerX + radius *t1, centerY - k* radius*t1,centerX -
radius * t2, centerY + k*radius*t2);

t3 = random(minR,maxR);

t4 = random(minR,maxR);

line(centerX - radius *t3, centerY - k* radius*t3,centerX +
radius * t4, centerY + k*radius*t4);
```

The algorithm for generating a star (Figure 1) is presented in Algorithm 2. As Figure 1 shows, a star consists of four short straight lines which meet in one center point. The approximate slope of the four lines is 1, -1, 0 and none (which means the line is vertical). Given the coordinates of the center point and the radius of the star, we could randomly generate four deviation values in the given range to control the slope of the four straight lines. Then we calculate the coordinates of two endpoints of each line and draw the line on the canvas. In this way, none of the stars we drawn would be exactly the same.

Figure 5 shows the process of simulating sharp and smooth angled hourglasses, that are both composed of two straight lines and two curve lines. The generation algorithm is outlined in Algorithm 3. Given the coordinates of the center point and radius for size control, we use the same techniques as for stars to draw two straight lines of an hourglass. Next we choose two control points for the Bezier curve according to the position of endpoints A and B . As shown in Figure 5, if the x-coordinates of control points C and D are within the range of endpoints A and B , we could draw a sharp angled hourglass. If they are out of the range, we could draw a smooth angled hourglass.

Algorithm 3: Draw sharp (smooth) angle hourglass

Input: **centerX:** the x-coordinate of the element
 centerY: the y- coordinate of the element
 radius: size of the element
 k: slope of the AB
 type: type of the element

```

t1 = random(minR,maxR);      t2 = random(minR,maxR);
t3 = random(minR,maxR);      t4 = random(minR,maxR);
x1 = centerX + radius * t1;   y1 = centerY - k * radius * t1;
x2 = centerX - radius * t2;   y2 = centerY + k * radius * t2;
x3 = centerX - radius * t3;   y3 = centerY - k * radius * t3;
x4 = centerX + radius * t4;   y4 = centerY + k * radius * t4;
line(x1,y1,x2,y2);
line(x3,y3,x4,y4);
if (type == sharp ){
    bezier(x1,   y1+1,   x1-0.4*radius,   y1-0.2*radius,
           x3+0.4*radius, y3-0.2*radius, x3, y3+1);

    bezier(x2,   y2-1,   x2+0.4*radius,   y2+0.2*radius,
           x4-0.45*radius, y4+0.2*radius, x4, y4-1);
}
else if (type == smooth){
    bezier(x1-1,   y1+1,   x1+0.7*radius,   y1-1*radius,   x3-
           0.7*radius, y3-2*radius, x3+1, y3+1);

    bezier(x2+1,   y2-1,   x2-0.7*radius,   y2+1*radius,
           x4+0.8*radius, y4+1.5*radius, x4-1, y4-1);
}

```

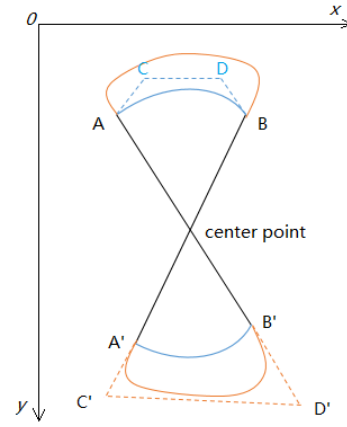


Figure 5: Simulate sharp (smooth) angle hourglass

5.2 Algorithm Design

In the second step, i.e. coloring, we adopt color-filling algorithm adapted from the flood fill algorithm. The algorithm is outlined in Algorithm 4.

Algorithm 4: Adaptive fill

Input: **startX:** the x-coordinate of start point
 startY: the y- coordinate of start point
 newCol: fill color

```

Qx.add((int) start_x); Qy.add((int) start_y);
curCol = get(start_x, start_y); //get start point color
while (min (Qx.size (), Qy.size()) > 0 ) {
    curX = Qx.get(0); curY = Qy.get(0);
    Qx.remove(0); Qy.remove(0);
    addValidXY(curX+1, curY);
    addValidXY(curX-1, curY);
    addValidXY(curX, curY+1);
    addValidXY(curX, curY-1);
    set(curX, curY, newCol);
}
while (min (BorderX.size (), BorderY.size()) > 0) {
    set(BorderX.get(0), BorderY.get(0), col);
    BorderX.remove(0);    BorderY.remove(0);
}

```

All the objects to be filled are closed shapes. The fill color for an object is chosen based on the current color of the object, which requires the background color of the canvas to be uniform. Starting from one randomly chosen pixel point within the object, the adaptive fill algorithm reads and saves the RGB values ($curCol$) of the pixel and puts the point into a queue. It then begins looping; during each iteration of the loop, it takes the first point out of the queue and replaces the point's color with a new color. The algorithm then retrieves the four neighboring points one by one

and reads their RGB values. For each point, the algorithm checks if its RGB values are the same as the initial point's RGB values *curCol* and if it is not in the queue, the algorithm inserts the point into the queue. If the RGB values are not the same as those of the start point, the algorithm pushes the point in stack as the object's border. The loop terminates once the queue is empty when the object is completely colored. To achieve a smooth effect, we replace the border's color by the new color.

The adaptive fill algorithm requires only a new color and one coordinate of a point in the object as input to fill the entire object with multiple appropriate colors. By adding a list of points into stack and using a conditional check, we could color any object with different colors in different parts if the object is divided by one or more external lines.

5.3 Layout

One of the problems we need to address is the layout of the canvas as we put components on the canvas with controlled randomness while avoiding overlaps yet creating a harmonious picture. It is hard to detect whether two components are overlapped because the boundary of each component is irregular and varied. We assume a bounding circle around each colored pictorial element or semantic graphical object composed by several basic pictorial elements. Comparing the distance between the two centers of the bounding circles with the radii of them, it is easy to know the relationship of the two components. Before we draw components on the canvas, we check if the component overlaps with any other component already existing on the canvas. We pick another position for the component once such a clash happens.

As discussed before, one of the stylistic and significant features of Miro's artworks is partial overlapping of shapes. Using the bounding circle technique, we could control the components' overlapping ratio by pre-determining relevant parameters. For example, we set the distance between the centers of two bounding circles to be greater than 80% (partial overlapping) or 100% (no overlapping) of the sum of the radii. Alternatively, we could simply decrease the radius of each bounding circle so it would not cover the entire component and the outside of the bounding circle is allowed to overlap with other components.

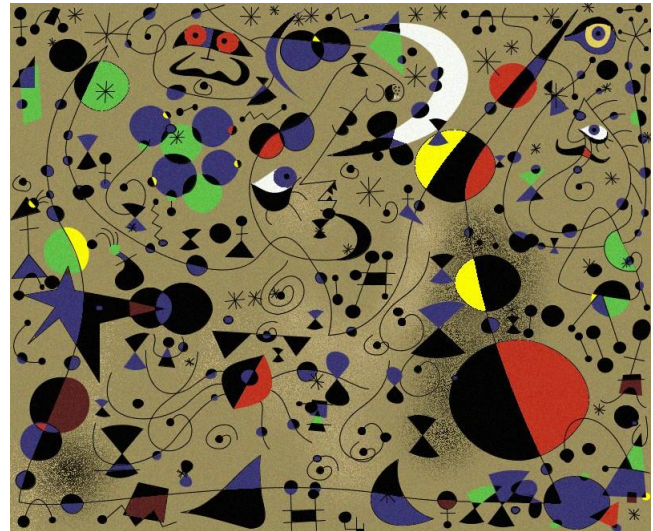
5.4 Results

We have used our process modeling approach to generate an image that mimics the original *Poetess* of Miro, as shown in Figure 6(a). Abstract pictorial elements of Miro can be identified clearly in *Poetess*. One can also easily observe long curve lines, sematic graphical objects and other significant features, based on which we divide the modeling process into four steps. We have used our approach to generate the "Poetess" style of images by randomly changing various parameters of the components, as shown in Figure 6(b) and (c).

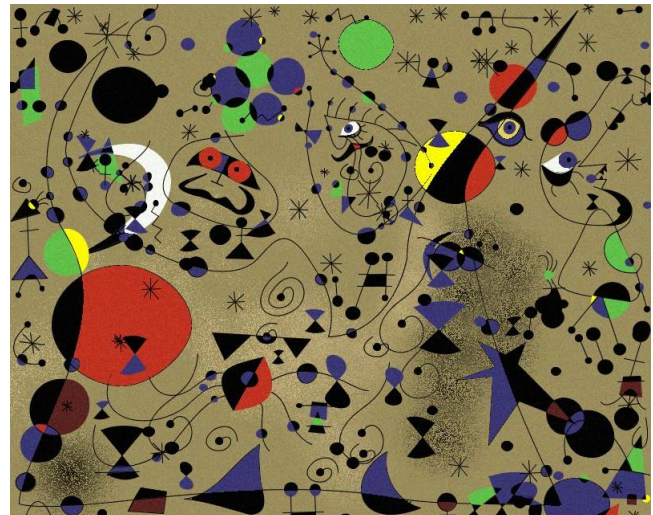
In the modeling process, we draw the background curve lines on to the canvas using the Bezier function, then draw sematic graphical objects composed of a few basic pictorial elements and color them using the adaptive fill algorithm. We extract 13 graphical objects from the original painting and surround each of them with a bounding circle. Their positions and sizes are generated with controlled randomness. Next, we fill the remaining canvas space with small abstract pictorial elements. The amounts of small abstract pictorial elements are set at 20 to 40 stars, 50 to 70 irregular circles, 6 to 10 spirals, 10 to 15 hourglasses and several compositions of irregular circles and lines. To achieve the

ageing effect, we retrieve each pixel's color and replace it by a color of the same shade with a small variation. A circular shade block is generated by points of a gradient color that form a circle and gradually fading outward. Irregular shade blocks are composed by a combination of several circular shade blocks.

To prove that our modeling approach works in general, we also generate abstract images based on another painting of Miro, *Ciphers and Constellations in Love with a Woman* (1940) (we will shorten it as "Cyphers"). This painting is also composed of abstract pictorial elements and sematic graphical objects. Figures 6(d) and (e) show two generated results. The parameter setting is similar to that in the generation process of *Poetess*, both combining the stochastic and user-defined strategies. Shade blocks are also added but with different colors and shapes.



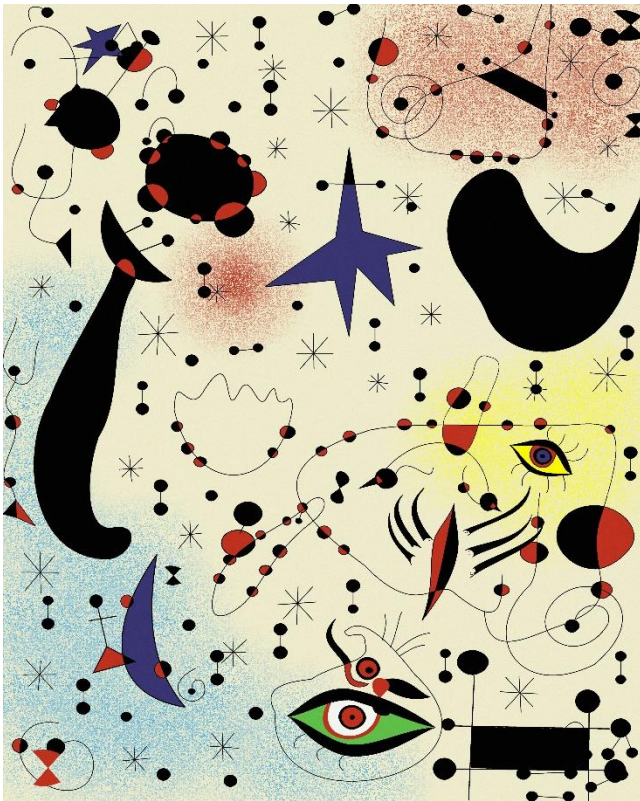
(a) Modeling the original *Poetess*



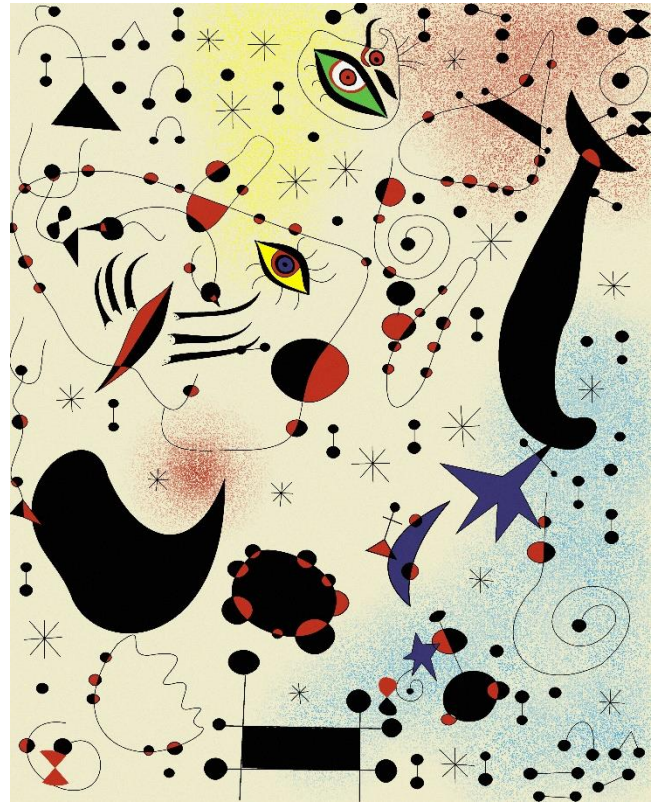
(b) A varied version of *Poetess*



(c) Another varied version of *Poetess*



(d) A varied version of *Cyphers*



(e) Another varied version of *Cyphers*

Figure 6: Generated images of Miro style

The entire generated image is finally saved as a .jpg file, whose size is determined by the canvas size set in the program. The current size of “Poetess” image is set at 800×660 , and “Ciphers” image is set at 910×1133 , but much larger images could also be generated by changing the size and scale within the program.

6. Conclusion

This paper has introduced a process modeling approach to generating abstract images in Miro surrealist style. Based on the distinguishing features of Miro’s works, especially those in his middle age, we model the artworks in four steps: structured drawing, adaptive coloring, space filling and noise injection. First we draw long curve lines as part of background. Next we draw semantic graphic objects and coloring them, and then fill the remaining canvas space with small abstract pictorial elements. We finally inject noises to simulate the ageing effects with the original look. By resetting or randomizing various parameters, various pictorial components could be reconstructed and recombined to form a new image. Our approach is scalable and generic in generating other artworks of Miro in the same period.

Our work generates encouraging results and establishes a solid foundation for further refinement. Adding more artistic elements may achieve unexpected effects. Moreover, generated styled paintings could be more aesthetic when considering semantic factors and interactions of various painting elements and coding them into rules. As a future work, we will attempt to model Miro’s works of other periods, and possibly combine them to generate more dramatic Miro images. Furthermore, we would attempt to display abstract paintings in dynamic forms.

7. REFERENCES

- [1] Ammeraal, L. & Zhang, K. (2007). *Computer graphics for Java programmers*. 2nd Edition, John Wiley & Sons.
- [2] Barnsley, M. F. (2014). *Fractals everywhere*. Academic press.
- [3] Braun, E., & Rabinow, R. (Eds.). (2014). *Cubism: The Leonard A. Lauder Collection*. Metropolitan Museum of Art.
- [4] Chun, Y. P., & Pollock, G. (2008). Melancholia and Cézanne's portraits: Faces beyond the mirror. *Psychoanalysis and the Image: Transdisciplinary Perspectives*, 94-126.
- [5] Danchev, A. (2012). *Cézanne: A life*. Pantheon.
- [6] Fogleman, M. (2011). *Procedurally generating images in the style of Piet Mondrian*.
- [7] Gatys, L.A., Ecker, A.S., & Bethge, M. (2016) A Neural Algorithm of Artistic Style, <http://arxiv.org/abs/1508.06576>.
- [8] Golding, J. (1988). *Cubism: a History and an Analysis, 1907-1914*. Harvard Univ Pr.
- [9] Haeberli, P. (1990, September). Paint by numbers: Abstract image representations. In *ACM SIGGRAPH Computer Graphics* (Vol. 24, No. 4, pp. 207-214). ACM.
- [10] Hubert, R. R. (1964). Miró and Breton. *Yale French Studies*, (31), 52-59.
- [11] Jensen, H. J. (2002). Mathematics and painting. *Interdisciplinary Science Reviews*, 27(1), 45-49.
- [12] Judelman, G. (2004, July). Aesthetics and inspiration for visualization design: bridging the gap between art and science. In *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on* (pp. 245-250). IEEE.
- [13] Kirsch, J. L., & Kirsch, R. A. (1988). The anatomy of painting style: Description with computer rules. *Leonardo*, 437-444.
- [14] Lubar, R. S. (1994). Miró's defiance of painting.
- [15] Machotka, P., & Cézanne, P. (1996). *Cézanne: Landscape into art*. Yale University Press.
- [16] Marcos, A. F. (2007). Digital art: when artistic and cultural muse merges with computer technology. *Computer Graphics and Applications, IEEE*, 27(5), 98-103.
- [17] Miró, J., & Rowell, M. (1992). *Joan Miró: selected writings and interviews*. Da Capo Press.
- [18] Mondriaan, P. C., & Stijl, D. (1965). Piet Mondrian.
- [19] Noll, A. M. (1966). Human or machine: A subjective comparison of Piet Mondrian's "Composition With Lines" (1917) and a computer-generated picture. *The psychological record*.
- [20] Pearson, M. (2011). *Generative Art*. Manning Publications Co..
- [21] Reas, C. & Fry, B. (2007). *Processing: a programming handbook for visual designers and artists* (Vol. 6812). MIT Press.
- [22] Tao, W., Liu, Y., & Zhang, K. (2014, June). Automatically generating abstract paintings in Malevich Style. In *2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS)* (pp. 201-205). IEEE.
- [23] Taylor, R. P. (2002). Order in Pollock's chaos. *Scientific American*, 287(6), 84-89.
- [24] Taylor, R. P., Micolich, A. P., & Jonas, D. (1999). Fractal analysis of Pollock's drip paintings. *Nature*, 399(6735), 422-422.
- [25] Taylor, R. P., Micolich, A. P., & Jonas, D. (2002). The construction of Jackson Pollock's fractal drip paintings. *Leonardo*, 35(2), 203-207.
- [26] Umland, A. (2004). A Challenge to Painting: Miro and Collage in the 1920s. *Agnes De la Beaumelle*. London: Paul Holberton Publishing, 61-69.
- [27] Whitfield, T. W., & Whiltshire, T. J. (1990). Color psychology: a critical review. *Genetic, social, and general psychology monographs*.
- [28] Wiedemann, D., Barton, R. A., & Hill, R. A. (2012). Evolutionary perspectives on sport and competition. *Applied evolutionary psychology*, 290-307.
- [29] Zhang, K. (2007). From abstract painting to information visualization. *Computer Graphics and Applications, IEEE*, 27(3), 12-16.
- [30] Zhang, K., Harrell, S., & Ji, X. (2012). Computational Aesthetics: On the Complexity of Computer-Generated Paintings. *Leonardo*, 45(3), 243-248.
- [31] Zhang, K., & Yu, J. (2014). Generation of Kandinsky art. *Leonardo*.
- [32] Zheng, Y., Nie, X., Meng, Z., Feng, W., & Zhang, K. (2015). Layered modeling and generation of Pollock's drip style. *The Visual Computer*, 31(5), 589-600.
- [33] Zhao, M., & Zhu, S. C. (2010, June). Sisley the abstract painter. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (pp. 99-107). ACM.