Erschienen in: Computer Graphics Forum ; 34 (2015), 2. - S. 311-323
https://dx.doi.org/10.1111/cgf.12562

Hardware-Based Non-Photorealistic Rendering Using a Painting Robot

Thomas Lindemeier and Jens Metzner and Lena Pollak and Oliver Deussen

University of Konstanz, Germany

(a) (b) (c)

Figure 1: *a) Painting robot with canvas and camera; b) painting tool with distance measuring unit; c) details from a painting.*

Abstract

We describe a painting machine and associated algorithms. Our modified industrial robot works with visual feedback and applies acrylic paint from a repository to a canvas until the created painting resembles a given input image or scene. The color differences between canvas and input are used to direct the application of new strokes. We present two optimization-based algorithms that place such strokes in relation to already existing ones. Using these methods we are able to create different painting styles, one that tries to match the input colors with almost transparent strokes and another one that creates dithering patterns of opaque strokes that approximate the input color. The machine produces paintings that mimic those created by human painters and allows us to study the painting process as well as the creation of artworks.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—

1. Introduction

In non-photorealistic rendering and particular painterly rendering researchers investigate the production of artistic images. Typically, results are pixel images that look like human paintings. In this paper, however, we propose a painting machine plus associated algorithms that is able to create real paintings with either thinned or opaque paints. A modified industrial robot applies thousands of strokes to a canvas until the result resembles a given target function - an image or an image with additional information such as depth or semantics. It uses ink or acrylic paint for its works and is able to create different painting styles.

Our machine uses a visual feedback loop to control the painting process - it supervises itself while painting using a standard digital camera. Strokes are not determined at the beginning but are computed iteratively based on the difference between the given target function and the already created strokes on the canvas. This has a number of practical advantages, but also raises some interesting research questions:

Limited palette: Most painterly rendering algorithms assume that strokes can be painted with any color. Typically, the average color of the input is used for a brush stroke. However, our machine works with a set of pre-defined colors since it is not possible yet to automatically mix new colors using our system. Therefore, we have to find algorithms to approximate input colors with such a set of paints on the canvas.

Painting styles: Our visual feedback loop helps to overcome many practical problems: Real paint strokes interact in a complex way that is very hard to simulate on a computer. This is especially the case with thick paint and wet-on-wet interaction where a number of effects can be created such as 3D-profile interaction, color smearing and complex deformations of the brush tip. Using an iterative feedback loop we try to imitate the painting process performed by humans and are able to apply relatively simple simulation methods for the determination of new strokes. Introduced errors can be corrected in subsequent steps after a new feedback iteration. Such methods, however, require that painterly rendering styles are formulated in a relative manner that apply to a given target function and a canvas with a partly produced painting.

Efficiency: Most painterly rendering algorithms neglect efficiency. If simulated on a computer, tens of thousands of strokes can be created in order to produce a desired result. However, our painting robot works at the speed of a human painter or even less and needs hours to complete a painting. Thus, efficient strategies have to be developed for minimizing the required time.

In this paper we present a framework based on Voronoi diagrams that allows us to realize painting styles for thinned and also opaque paints. For thinned (almost transparent) paints, the first variant of the algorithm tries to resemble the colors of the input image as close as possible by many paint layers that gradually change the color on the canvas to approximate the input. For opaque colors we use variant that creates dithering patterns (known from halftoning methods for printing) based on discrete strokes that are carefully aligned and approximate the input colors on average.

2. Related Work

In our review of related publications, we focus on works that are directly related to our approach, thus we sketch painterly rendering and image processing methods as well as painting machines.

Painterly rendering methods: These methods create non-photorealistic images by simulating brush strokes and color interaction. Hertzmann [Her98] proposes a painting method that starts with strokes made by a large brush which are subsequently painted over with smaller brush strokes. Strokes are placed within regular grid cells, which can be seen as a jittered sampling method. This works well but results in a reasonable amount of unintended over-painting. Since we have to realize our paint strokes physically, time is a critical issue and we have to find optimal strategies here. Furthermore, Hertzmann paints brush strokes with arbitrary colors. We have a pre-defined color set and thus have to find other methods to create the wanted color on the canvas.

Chun et al. [CJK11] use color quantization and segmentation to create many small segments in an input image and choose the center of each segment as the seed point for a brush stroke. Vanderhaeghe et al. [VBTS07] propose non-uniform Poisson-disk distributions with blue-noise properties for short strokes such as stippling, pointillism, hatching and painterly rendering. The density of distribution can be controlled by integrating an importance map. Shiraishi et al. [SY00] use space-filling curves to distribute seed points at locations where brush strokes have to be placed. We found an early optimization scheme by Hertzmann [Her01] most suitable for our situation in which we have to compute new stroke on the basis of existing ones and extend his idea by using a Voronoi-cell-based optimization (Lloyd's relaxation [Llo82]) to find new stroke positions.

Recent painterly rendering methods [ZZXZ09, ZZ10, ZZ11] divide the input images into foreground and background layers. They use graph cuts [BJ01] with foreground and background scribbles obtained using the method described in Li et al. [LSTS04] and a mixture of interaction and a bag-of-words classifier to label object types [LFT05]. We use the same approach to separate foreground from background and classify the objects contained in an input image. Collomosse and Hall [CH02] introduce salience maps to filter out regions of strong color variations that should be represented in more detail.

Color processing: To create rich colors in a painting with a fixed given set of colors, color transfer and dithering have to be applied. Reinhard et al. [RAGS01] propose color transformation from one image to another. Pitié et al. [PKD05, PKD07] refine this method to transform any input image in a given color space. Cohen-Or et al. [COSG*06] select good color schemes and use them for color transformation. Yang and Yang [YY06] produce renderings of Seurat's pointillism with dithering using colored dots. Artists often use a certain color set to create paintings instead of using the colors of a given scenery. Therefore, we have to use such methods to restrict and transfer the colors in our input image to what can be mixed from the given set of colors in our color repository.

Painting machines: Jean Tinguely (1925-1991, see [Wik13]) was one of the first artists that built painting machines to create complex but mostly random patterns. He followed a tradition from the 19th century, when people were fascinated by mechanical apparatuses. Harold Cohen [Coh12] built a plotter with the ability to paint abstract paint-

ings. His project, known as 'AARON' is regarded as the most important painting machine in contemporary art.

Other early artists such as Frieder Nake [Wik12] used the upcoming pen plotters in the 1960s to create artistic graphics. Today, a number of artists use painting machines, Ben Grosser [Gro13] and Holger Baer [Bae13] for creating abstract paintings. Specialized plotters such as Zanelle [Arm12] or Vangobot [KM13] are able to create colorful paintings today, but non of them uses a feedback mechanism. While Zanelle works with brushes and creates only very discretized, pop-art like paintings, Vangobot uses a sophisticated color mixing machine and applies paint directly on a canvas like an inkjet printer.

Tresset and Fol Leymarie [TFL12, TFL13] created a robotic installation that is able to create portrait sketches of people. They use a form of visual feedback to guide the painting process. This system creates sketches that have an own artistic style, it is however limited to sketches.

Deussen et al. [DLPT12] present e-David, a painting robot that works with visual feedback and is thus able to create subtle paint representations. Our system builds on this project, but in contrast to the pen-and-ink drawings published in [DLPT12,LPD13] we work with acrylic paint. That requires a completely modified painting pipeline and imposes a number of algorithmical challenges. Furthermore, our stroke placement mechanism is based on Voronoi diagrams which differs from their greedy optimization method.

3. Overview

The purpose of our system is to create acrylic paintings that look similar to those of human artists. This requires to be able to handle brushes and paints like humans do; thus, we use an industrial robot arm (Reis Robotics RV20-6) with six degrees of freedom instead of a simple pen plotter. This allows us to vary the angle of attack, to realize complex movements, to dip brushes into the paint containers of a repository, to clean them and also to change brushes freely during painting (cf. Figures 1(a) and (b)).

After an input image has been chosen (the system can also work with additional information such as depth values or semantic information) a set of acrylic paints is selected. We typically use harmonic color schemes and, as mentioned in the introduction, transfer the input image into a color space that can be realized with the set of selected acrylic paints (cf. [COSG*06]).

Similar to human painters we also paint from back to front; thus we have to separate the input image into a set of segments with an associated depth order (cf. [ZZXZ09, ZZ10]). Each of the segments is painted with a large brush and subsequently refined using smaller brushes. Background segments are painted with less detail than foreground segments. This is realized by defining a set of brushes for each segment. The machine always starts with a large brush and continues with smaller ones. The smallest brush determines the amount of details that can be represented by a layer. For the foreground layers we additionally use a salience map derived from Collomosse and Hall [CH02] for representing important details with higher detail. This map then refines the selection of brushes for these layers.

We use the Canon EOS 5D Mark II camera to take pictures of the canvas for visual feedback and OpenCV [Bra] for adjusting the camera image in order to get a geometrically correct and white-balanced image. Each of the segments is painted with visual feedback control. Strokes are applied until the difference between canvas and given input is sufficiently small. Details for the general feedback mechanism are described in Lindemeier et al. [LPD13].

When using acrylic paints, this general feedback process has to be modified. Opaque paints will potentially never come close to a given color of the image, instead we have to create a dither pattern consisting of carefully selected and oriented strokes of different color that *on average* represent the input color. Such patterns were invented during the period of impressionism in which painters wanted to capture a particular moment or lighting situation and had to paint fast without the time-consuming and mechanically extremely complex mixing of paints on the palette.

Using the above-mentioned iterative optimization scheme based on Voronoi diagrams (see Section 5.2), new strokes are painted in between existing ones by maximizing the minimal distance, which results in a balanced and aesthetic placement of strokes and allows for gradually approaching an input color with many layers of mostly transparent paints. This mechanism can be extended to work with opaque paints. As mentioned above, here we need to create patterns of strokes with different color that represent an input color on average over an area. We start with one color and create a layer of strokes. Using the next color, new brush strokes are placed in between existing ones. If the space does not allow new strokes to be placed in between existing ones, overpainting happens, but always in a way that stroke layers maintain their overall distribution.

4. Painting in layers from back to front

In order to paint layers from back to front we need to segment the input image. As reported by other authors [ZZXZ09, ZZ10, ZZ11] this is typically not possible to do fully automatic. Thus we use a semi-automatic separation into layers with depth order. The user scribbles the different regions and, based on a pre-segmentation with a watershed transformation, the final segmentation is computed using graph cuts as described in [LSTS04].

In contrast to Zhao and Zhu [ZZ10, ZZ11] we process our background layers further before painting them. To create good stroke directions and a smooth painting of such layers,

(a) (b) (c) (d) (e)

Figure 2: *Processing of layers: a) input image; b) separated layers; c) hole-filling; d) detail without hole-filling; e) detail with hole-filling and over-painting.*

Input : Input image
Output: Acrylic painting
$P \leftarrow$ color palette sorted from dark to bright;
$L \leftarrow$ layers sorted from back to front;
foreach *layer* $l \in L$ **do**
  $B \leftarrow$ select brushes;
  foreach *brush* $b \in B$ **do**
    $O_b \leftarrow$ orientation field according to b;
    $W \leftarrow$ distance map (see Equation 1);
    while $||W||$ *is too large* **do**
      foreach *color* $c \in P$ **do**
        $W \leftarrow$ distance map update;
        $S \leftarrow$ `SampleStrokes` (W, O_b, b);
        foreach *stroke* $s \in S$ **do**
          if *s improves canvas* **then**
            paint s;
          end
        end
      end
    end
  end
end

Algorithm 1: General painting algorithm.

we have to fill holes that are caused by foreground objects. The reason is the following: stroke directions are computed by directing them perpendicular to the image gradient. This is a common technique in painterly redering [Lit97]. If, however, small objects of the foreground disturb the image gradient of the background the corresponding direction information is also affected.

Figure 2 shows the effect. If the background layer is not processed before painting, stroke directions as well as the smooth appearance of the sky become distorted (Figure 2(d)). Thus, we use morphologic filters (dilation) to remove the foreground objects and apply the method described by Barnes et al. [BSFG09] for texture filling with the background pattern (cf. Figure 2(c)). The structuring element is in this case a circle with the maximum brush size as radius. We further process this modified layer in each painting iteration by applying a morphological opening to mask out objects of the foreground that are smaller than the currently selected brush. This enables us painting thin objects like the antenna not with the potentially large brush used for the background but with the smaller-sized brush that is used for the foreground. Figure 2(e) shows the final result.

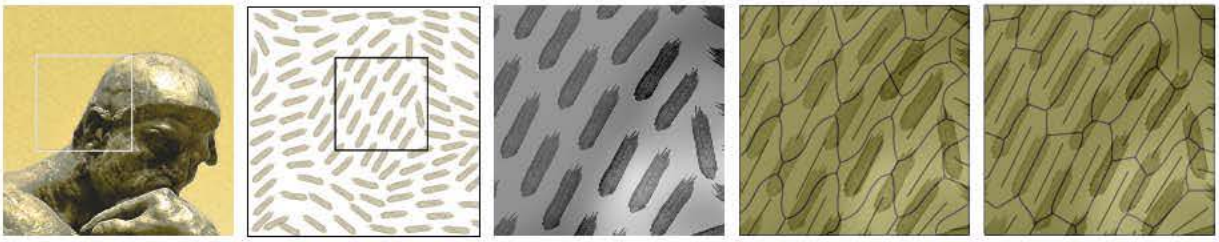
An object type (sky, water, face, etc.) can be assigned to each of the layers. This allows us to use different painting methods for depicting different materials. A tree might be represented by many small and randomly oriented strokes while for the sky long and smoothly aligned strokes can be used. Section 6 shows style parameters and their values for the presented paintings.

5. Painting a single layer

To paint a layer, we start with the largest brush and iterate through all available colors. For each iteration we take a picture of the canvas and compute the difference of canvas and input on a pixel-per-pixel basis. The resulting color distance map W is used to determine new positions for potential strokes. This is done using a Voronoi-based optimization process. Each of the stroke candidates is then evaluated and only strokes that improve the canvas using the current color are realized by the machine. For each brush size that is assigned to the segment we repeat this process until the overall difference between input and canvas is sufficiently small. Then we move on to the next layer.

5.1. Stroke orientation

A classic method for calculating the orientation of brush strokes is to direct them perpendicular to the image gra-



(a) (b) (c) (d) (e)

Figure 3: *Voronoi relaxation: (a) source image; (b) the current canvas and a rectangle showing the zoomed area for ((c), (d), (e)); (c) zoomed in color distance map W of source and canvas according to (b); (d) Voronoi diagram of initial stroke candidates; (e) Voronoi diagram and stroke candidate locations after 150 iterations of the relaxation process (see Algorithm 2). Stroke candidates place themselves in between the existing strokes.*

dient [Lit97, HE04]. In our implementation we use structure tensor fields where the strokes follow the eigenvector with smallest eigenvalue [TD07, ZHT07, KK11]. Weak tensors without meaningful orientation are smoothed to construct valid orientation fields for a large variety of input images. Sometimes image gradients are completely missing. Hays and Essa [HE04] interpolate from strong gradients using radial basis functions. We use diffusion methods from boundaries to interpolate the missing gradient information (see Orzan et al. [OBW*08]).

5.2. Placement of stroke candidates

As mentioned above, three classes of methods for stroke positioning have been used in related works, random placement and grid-based or Poisson disk stroke positioning. The latter two methods position strokes with defined minimal distances to each other and thus reach a smooth appearance.

In our case, having the painting machine, the problem has to be reformulated: for each new color layer we need to paint a set of strokes that react to the already existing strokes on the canvas, are aesthetically distributed and can be arranged to a dither pattern if using opaque paints.

We solve this problem by utilizing Lloyd's method, a local Voronoi-diagram-based optimization method that allows distributing graphical objects on the plane. We compute the Voronoi cells of each object and move the objects to the center of gravity of their Voronoi cells during each iteration, see [DHVS00] for a version of this optimization with points. To create varying densities a weighted version of this optimization is used [Sec02].

The idea behind this process is as follows: Lloyd's method arranges a set of given objects in a distribution where all objects maximize their distances to their neighbours. Using weighted Voronoi relaxation [Sec02] the objects (stipples in Secord's paper) can be arranged in a way that they concentrate at places where an underlying image is darker. This way objects, if drawn with black color, can be used to represent the gray-scale values of the image. We utilize this effect and change the criterion for stroke placement: we define the weight for the weighted Voronoi relaxation in a way that the strokes concentrate at places where the color distance map W has high values, i.e. where the color distance of canvas and input is large. This results in a pleasing arrangement and concentrates strokes at the right places.

5.3. Computing the color distance map

The pictures of our camera system are converted to the perceptually linear color space CIELAB for feedback analysis. Assume a brush b with size σ is selected. Let S, T be input and feedback image, the latter taken by the camera. We blur S according to the current brush size σ with a gaussian kernel G_σ, $S_\sigma = G_\sigma * S$. Subsequently, for each pair of pixels from S_σ, T we determine the color difference by:

$$W(x,y) = ||S_\sigma(x,y) - T(x,y)|| \quad (1)$$

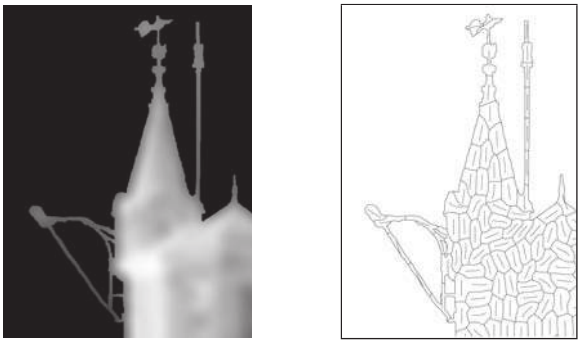
where $||x-y||$ is defined by the Euclidean distance in CIELAB color space and thus accounts for the perceived color distances [Poy03].

All values of W below a given threshold or not belonging to the current segment are set to zero. Such regions are discarded from the following relaxation process (see Figure 4).

5.4. Relaxation using Lloyd's method

Hiller et al. [HHD03] propose an enhanced version of Lloyd's method for distributing small 2D objects such as lines or triangles. They use second order moments for orienting their elements. However, in our case the orientations are given by the orientation field, which is computed from the input image.

We start our process by selecting initial seeds for stroke candidates (Algorithm 2 line 1). This can be random positions, a grid layout or a Poisson disk arrangement. In our case a grid layout seems to be sufficient. Please note again that not all of these candidates will be painted later, they just define positions for potential strokes. The number n of seeds



(a) (b)

Figure 4: *Stroke candidates: a) color distance mask; b) created positions and cells by Lloyd relaxation.*

Input : Weight map W, orientations O_b, Brush b;
Output: Relaxed stroke distribution;
1 $S \leftarrow$ seed points (see Equation 2);
2 $m \leftarrow \infty$;
3 **while** $m > 0$ **do**
4 $\quad T \leftarrow$ create strokes from S, O_b, b;
5 $\quad V \leftarrow$ Voronoi diagram of T;
6 $\quad S^W \leftarrow$ weighted centroids of V according to W;
7 $\quad m \leftarrow ||S - S^W||$;
8 $\quad S \leftarrow S^W$;
9 **end**

Algorithm 2: Stroke generation and relaxation.

depends on the current brush size σ and a sampling scale λ, w and h are the dimensions of the input image:

$$n = \frac{wh}{(\lambda\sigma)^2} \tag{2}$$

The choice of λ influences the size of the Voronoi cells during the relaxation and thus defines the area which is used to evaluate the stroke candidate later. If the cell size is as small as the stroke, only the underlying pixels will be counted. If it encompasses a larger area, the stroke is able to react to its neighbourhood, which allows us to create the already mentioned dither patterns.

For a given seed point the path of a stroke is computed using stroke integration as proposed by Hertzmann [Her98] (Algorithm 2 line 4). The path is integrated back and forth along the orientation field with the current brush radius as step size. The length is randomly chosen between a given minimum and maximum value according to the style parameters (see Section 6).

The Voronoi diagram is computed using a method proposed in [Hae90, WND97, HKL*99] that utilizes graphics hardware (Algorithm 2 line 5). They render three dimensional meshes and exploit the z-buffer to compute distances of objects. Thus, in fact they compute a Voronoi diagram, for details please refer to Hiller et al. [HHD03]. We compute the Voronoi diagram of n Strokes $S = \{s_1, ..., s_n\}$, sampled as our initial stroke set with $V(s_i)$ being the Voronoi cell of a Stroke s_i. The moments and weighted centroids of Voronoi cells are computed similar to Hiller et al. [HHD03]. They pointed out that each Voronoi region can be represented by a function $\Psi_{V(s_i)}$:

$$\Psi_{V(s_i)}(x,y) = \begin{cases} 0 & \text{if } (x,y) \notin V(s_i) \\ W(x,y) & \text{if } (x,y) \in V(s_i) \end{cases} \tag{3}$$

with $W(x,y)$ being the value from the distance map at (x,y)(see Section 5.3).

The weighted centroid c_i of a Voronoi cell is defined by the moments of a Voronoi cell (Algorithm 2 line 6). These moments are computed by accumulating the weights and coordinates of pixels that belong to $V(s_i)$ (cf. [DI13]):

$$m_i^{0,0} = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} \Psi_{V(s_i)}(x,y) \tag{4}$$

$$m_i^{1,0} = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} x \cdot \Psi_{V(s_i)}(x,y) \tag{5}$$

$$m_i^{0,1} = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} y \cdot \Psi_{V(s_i)}(x,y) \tag{6}$$

$$c_i = \begin{pmatrix} m_i^{1,0}/m_i^{0,0} \\ m_i^{0,1}/m_i^{0,0} \end{pmatrix} \tag{7}$$

Strokes that form cells with a high accumulated color distance (high zero moments $m^{0,0}$) should be moved at a slower velocity during the relaxation process since they already lie in regions where the color distance is large. Cells with maximal moments of first order are given velocities that are close to zero in the relaxation step. Cells with minimal moments are given the full velocities. The adaptive relaxation velocity α_s is computed by normalizing the moment of first order of a Voronoi cell with the maximal moment of all Voronoi regions:

$$\alpha_i = 1 - \frac{m_i^{0,0}}{\max(\{m_1^{0,0}, ..., m_n^{0,0}\})} \tag{8}$$

The movement within one step of the relaxation is then defined by:

$$p(s_i)^{t+1} = p(s_i)^t + \alpha_i \cdot (c_i^t - p(s_i)^t) \tag{9}$$

with $p(s_i)$ being the stroke seed used to integrate the stroke s_i.

We compute the stroke path after each relaxation step according to the new stroke seed (Algorithm 2 line 8 and 1). This update of the stroke path and resulting change of orientation influences the next relaxations. As a result, the process

does not always converge. However, this is typically negligible since it affects only a few strokes and the resulting distortions will be overpainted in the next iteration due to the feedback loop. The iteration is stopped once the movement of the strokes is below a given threshold.

In Figure 3 the process is shown. Assume the input image is already approximated by a stroke layer. In subfigure (c) the color distances with respect to the current painting color are shown. A set of seed points is distributed by using a grid layout. After applying the Voronoi relaxation with regards to the distance map the strokes distribute in between the existing strokes.

5.5. Stroke candidate evaluation

After the relaxation process has finished we have a large number of well-positioned *stroke candidates*. Now we need to check the quality of each candidate with respect to improving the painting (Algorithm 1 line 13). Only candidates that improve the color locally will be realized.

We use information contained in the Voronoi diagram to evaluate the quality of our stroke candidates. The Voronoi diagram partitions the image in regions that belong to certain strokes. All pixels in such a Voronoi cell are closest to the cell-defining stroke candidate. Therefore we roughly know the region that is affected by each stroke. To determine the quality of a stroke candidate, we simulate the application of the stroke to the canvas and compare the color distances of all pixels of the cell before and after stroke placement.

Color mixing on our virtual canvas is approximated by a simple alpha blending that yields a sufficient approximation of dry color mixing. Stroke textures are created from a photograph of real brush strokes created by the robot. We are thinking about integrating more complex color mixing models as discussed in [CAS*97] in the future by using multispectral imaging. However, the results we achieve using alpha blending seem doing well for our purpose. Please recall here, that due to the visual feedback mechanism and iterative nature of our painting machine a simple color model and simulation is sufficient.

Thinned paint: An interesting aspect of this evaluation scheme is that it can be applied to both, thinned paints and opaque paints (and all intermediate transparency levels). Figures 1(c), 9(a),(b), 10(a) and 11 show results obtained with a set of thinned paints that allow to gradually approximate the input colors by many layers of paint. In this case many stroke candidates are distributed onto the canvas so that each of the Voronoi cells is not much larger than the later realized stroke. We estimate the amount of strokes using Eq. (2) with $\lambda = 1.5$. This way the color differences of the cell reflect directly what would be gained if the stroke is realized.

Opaque paint: If opaque paints are used we distribute a smaller number of stroke candidates on the canvas by setting $\lambda = 3$ to reduce the amount of strokes fed to the relaxation process. The resulting Voronoi cells encompass now a larger area than occupied by the realized stroke and thus measure the visual effect of an applied stroke in its entire vicinity.

As mentioned above, this enables us to introduce a form of dithering: strokes do not just improve the colors of the canvas directly located under themselves but also their local neighborhood. Thus, a stroke with a color that is too dark for representing the color directly on the canvas may be placed in a certain region since the average of the color differences within the associated Voronoi cell still improves the result. In the next painting iteration of the algorithm, when a light color is selected, it might be placed in its direct vicinity to improve its Voronoi cell that is far too dark on average due to the preceding dark stroke. As a result strokes of different colors will be placed in a close neighborhood to create impressions of new colors not contained in the palette if viewed from a far distance.

Figure 5 gives an overview of the process: given is a color ramp we want to paint with yellow and black color, in one case with opaque paints, in the other case with thinned paints (we typically thin the acrylic paints by dissolving them in acrylic medium with a factor of 1:4-1:10). In both cases our algorithm is able to produce painterly representations of the input images. The dithering effect is furthermore illustrated in Figure 6. The painting simulation was rendered using a palette of six colors and one brush size. If viewed from a distance it can be seen that the application with just a few colors in a local neighborhood produces an impression of multiple colors (e).

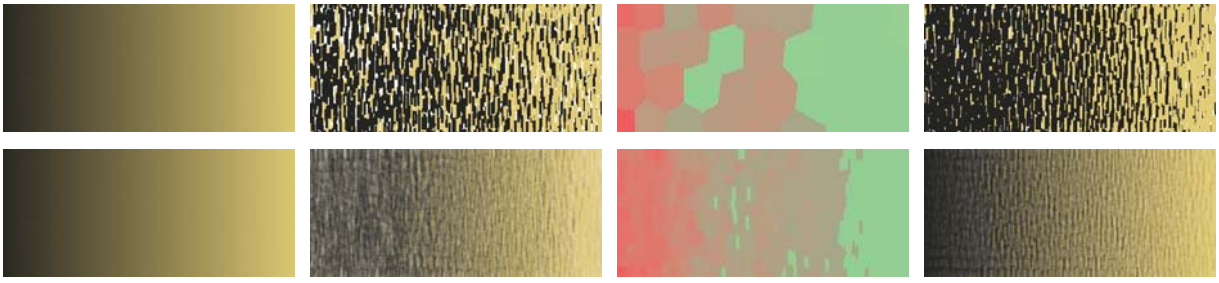
The strokes are then applied one after the other to the canvas. This follows the general methods that were described in [DLPT12, LPD13].

6. Results

We demonstrate our results by showing photographs of the paintings created by our machine.

Figure 9 shows three versions of a small tower painted with a set of four transparent paints (mixed from pale gray, raw umber, van Dyke brown, light ochre, sand and titanium white). No visual feedback was used to create the painting in (a). The system was therefore not able to react to imprecise applications of brush strokes and inaccurately measured color values and opacity. This results in wrong colors and missing details. The other two paintings in subfigures (b) and (c) were painted using visual feedback. While the painting shown in (b) was painted without using layers, in (c) three different layers were applied. This allows us to combine a softly painted background with enhanced details in the foreground. All paintings were created using approximately 9,000 strokes and needed 17 hours each to complete.

A comparison of our Voronoi-based relaxation method



(a) (b) (c) (d)

Figure 5: *Visualization of the evaluation process: Given is a color ramp (a) that is to be represented with yellow and black color. In the upper row opaque paints are used, in the lower row thinned paints. (b) canvas after a number of painting iterations (25 for opaque and 39 for thinned paint, (see Algorithm 1 line 8)); (c) Voronoi diagram in both cases, for the opaque paint much less cells are used. The color indicates if a stroke applied for this cell would improve the painting (red=worsen, green=improving); (d) when the process is continued in both cases an approximation to the input is reached (65 iterations for opaque paint, 137 for thinned paint).*

(a) (b) (c) (d) (e)

Figure 6: *Simulation of the Voronoi cell dithering painting process using six different colors and a single layer: (a) input image; (b) canvas after one iteration, each color has been applied once; (c) iteration 2; (d) iteration 3; (e) iteration 10, viewing the result from a distance shows that the dithered strokes approximate the input quite well.*

with Hertzmann [Her98] shows that on average we are able to save 20% of the strokes, while still retaining a similar visual quality. For the above paintings this means three hours less of painting time.

Figure 11 shows a winter scene composed of six layers. The background layers were painted in a very rough style, while the bridge has been realized with detailed strokes. Here we used five colors (mixed from van Dyke brown, titanium white, lamp black, cobalt blue hue deep, lilac, cerulean blue and primary blue cyan) performed approximately 8,000 strokes in total and needed 15 hours for completion.

Figure 10 shows the effect of slightly changing the used colors. Our visual feedback adapts to the new situation and creates a new and yet not too dissimilar painting. By just using black and white as colors in combination with a gray background, our robot is able to create aesthetically pleasing paintings (see Figure 8) with expressive character.

Figure 12 demonstrates the dithering possibility of our proposed method. A palette with four different paints mixed from krapp dark, carmine red, cadmium red, indian yellow, sand, black and white were used to represent the image. All paints were opaque, thus the corresponding version of our algorithm was used. Especially the sky shows that the dithering pattern represents the needed variations in the input image quite well.

6.1. Style parameters

Table 1 shows the different style parameters we use to control the layout of strokes. The values l_{min} and l_{max} are the minimum and maximum number of sample points defining the path of a stroke. Since sampling is done with approx. one mm this also defines the length of the strokes. All painting shown below are painted on a 50x70cm painting cardboard that was mounted on our stable canvas.

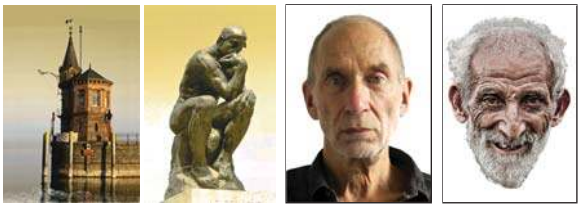
Table 1: *Style parameters and available labels.*

Label	l_{min}	l_{max}	ν	h_c
Sky (Figure 9)(b)	5	10	2	1
Water (Figure 9)(b)	3	7	1.5	0
Building (Figure 9)(b)	3	5	1	0.4
Winter tree (Figure 11)	4	8	1	0.8
Desert tree (Figure 12)	7	9	1	0.9
Sand (Figure 12)	6	8	1	0.6

The degree of smoothness of the orientation field that is used to guide the brush strokes can be scaled (see Section 5.1). The smoothness scale ν is an additional parameter we use for our stylization. Large values for smoothing are used, for example, to paint skies or water since we do not want to include fine details here (see background in Figure 9).

Since we are restricted by the size of real brushes we have to use a fixed set of predefined brush sizes for each layer. Every brush in a set is defined by its radius, another important parameter is the maximum number of color and brush layers allowed for painting on the canvas. This number prevents the robot from too much over-painting a certain area on the canvas.

The curvature of the stroke h_c as introduced by Hertzmann [Her98] defines the impact of the current directions of the brush during integration of a stroke (see Section 5.2). A value of $h_c = 0$ leads to straight strokes that are oriented according to the orientation of the image gradient at the first point of the stroke. A value of $h_c = 1$ results in curved strokes where the stroke follows the orientation field for its whole path.



(a) (b) (c) (d)

(e) (f)

Figure 7: *Input images for paintings in Figures 9, 10, 12 and 11.*

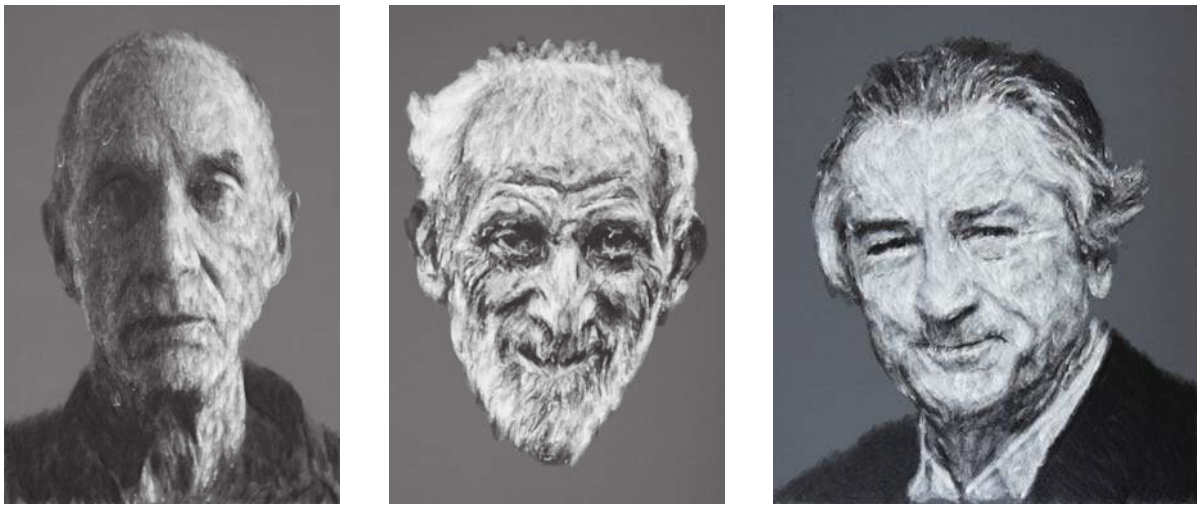
7. Conclusion

Our painting machine in combination with acrylic paints and its feedback control allows us to create paintings in a painterly style that reminds to those of human painters. We separate the painting into a number of layers of different depth and use individual style parameters for each of the layers to create an adequate painterly representation.

The method allows us to separate foreground objects from the background visually. The optimization method based on Voronoi diagrams reduces painting time about 20%. Furthermore, the optimization allows us to paint with thinned paints as well as with opaque paints while distributing the strokes in a aesthetically pleasing way. We are sure that this unified way of processing color layers will be useful for a number of applications.

Moreover, the robot can be used to study the painting process and to find out to what extent artistic paintings can be created by machines. This raises interesting questions about the necessity and importance of creative processes during painting and could even enable new forms of paintings in the future: painters can focus on the creative aspects of a painting and teach their ideas to the painting machine. The machine then realizes the artisanal aspects of the painting and eventually the artist finishes it to reach the desired effect. This way manual labor and creative idea might be decoupled to a large extent.

Future works will furthermore encompass new painting styles such as expressionism and pointillism. We further want to move from digital images as our input towards more complex target functions such as 3D scenes and implicitly given objects such as fractals. We also think about automatic composition as described in Kalaidjian et al. [KKM09]. Therefore, we have to teach the robot how to draw certain objects and object classes in particular painting styles. The robot then chooses by itself how to represent objects in a painting.



(a) (b) (c)

Figure 8: *Black and white paintings created with our painting robot. (a) Input Figure 7 (c); (b) input Figure 7 (d); (c) painting after a photograph of Robert de Niro.*

References

[Arm12] ARMAN P. V.: Zanelle. http://www.vanarman.com/, 2012. March 13th, 2012. 3

[Bae13] BAER H.:. http://www.holgerbaer.com/, 2013. Januar 3rd, 2013. 3

[BJ01] BOYKOV Y., JOLLY M.-P.: Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on* (2001), vol. 1, pp. 105–112 vol.1. doi:10.1109/ICCV.2001.937505. 2

[Bra] BRADSKI G.:. *Dr. Dobb's Journal of Software Tools*. 3

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (Aug. 2009). 4

[CAS*97] CURTIS C. J., ANDERSON S. E., SEIMS J. E., FLEISCHER K. W., SALESIN D. H.: Computer-generated watercolor. *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97* (1997), 421–430. doi:10.1145/258734.258896. 7

[CH02] COLLOMOSSE J., HALL P.: Painterly rendering using image salience. *Proceedings 20th Eurographics UK Conference* (2002), 122–128. doi:10.1109/EGUK.2002.1011281. 2, 3

[CJK11] CHUN S., JUNG K., KIM J.: Oil painting rendering through virtual light effect and regional analysis. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry* (New York, NY, USA, 2011), VRCAI '11, ACM, pp. 419–422. doi:10.1145/2087756.2087833. 2

[Coh12] COHEN H.: Aaron. http://crca.ucsd.edu/ hcohen/, 2012. March 13th, 2012. 2

[COSG*06] COHEN-OR D., SORKINE O., GAL R., LEYVAND T., XU Y.-Q.: Color harmonization. *ACM Transactions on Graphics 25*, 3 (2006), 624. 2, 3

[DHVS00] DEUSSEN O., HILLER S., VAN OVERVELD C., STROTHOTTE T.: Floating Points: A Method for Computing Stipple Drawings. *Computer Graphics Forum 19*, 3 (Sept. 2000), 41–50. doi:10.1111/1467-8659.00396. 5

[DI13] DEUSSEN O., ISENBERG T.: Halftoning and stippling. In *Image and Video-Based Artistic Stylisation*. Springer, 2013, pp. 45–61. 6

[DLPT12] DEUSSEN O., LINDEMEIER T., PIRK S., TAUTZENBERGER M.: Feedback-guided stroke placement for a painting machine. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging* (Aire-la-Ville, Switzerland, Switzerland, 2012), CAe '12, Eurographics Association, pp. 25–33. 3, 7

[Gro13] GROSSER B.:. http://bengrosser.com/, 2013. January 13th, 2013. 3

[Hae90] HAEBERLI P.: Paint by numbers: Abstract image representations. *SIGGRAPH Comput. Graph. 24*, 4 (Sept. 1990), 207–214. doi:10.1145/97880.97902. 6

[HE04] HAYS J., ESSA I.: Image and video based painterly animation. In *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering* (New York, NY, USA, 2004), NPAR '04, ACM, pp. 113–120. doi:10.1145/987657.987676. 5

[Her98] HERTZMANN A.: Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 453–460. doi:10.1145/280814.280951. 2, 6, 8, 9

[Her01] HERTZMANN A.: Paint by relaxation. In *Proceedings of the International Conference on Computer Graphics* (Washington, DC, USA, 2001), CGI '01, IEEE Computer Society, pp. 47–. 2

[HHD03] HILLER S., HELLWIG H., DEUSSEN O.: Beyond stippling-methods for distributing objects on the plane. *Computer Graphics Forum 22*, 3 (2003). 5, 6

[HKL*99] HOFF III K. E., KEYSER J., LIN M., MANOCHA

(a) (b) (c)

Figure 9: *A tower (Figure 7(a)) painted using our robot: (a) Painted without visual feedback but using layers.; (b) Painted with visual feedback but without layers. The color differences to (c) are due to daylight changes when taking the photography and a slightly different color balance when creating the painting.; (c) painted with visual feedback and layers. In this result the reflections in the water look more realistic.*

(a) (b)

Figure 10: *Paintings after the Thinker by Auguste Rodin (Figure 7 (b)). Slight changes in paint colors results in a new but yet not too different painting due to the feedback mechanism. Note: here we did not do the Voronoi optimization for the background strokes, this results in a less regular distribution of the strokes.*

Figure 11: *A winter scene painted using layers and a blue color palette (see Figure 7(e) for the input image). In this painting, we intentionally highlight the bridge by using a detail brush for the foreground.*

Figure 12: *A desert scene painted using layers and stroke dithering with a palette with four opaque paints (mixed from krapp dark, carmine red, cadmium red, indian yellow, sand, black and white) (see Figure 7(f) for the input image).*

D., CULVER T.: Fast computation of generalized voronoi diagrams using graphics hardware. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 277–286. doi:10.1145/311535.311567. 6

[KK11] KYPRIANIDIS J. E., KANG H.: Image and Video Abstraction by Coherence-Enhancing Filtering. *Computer Graphics Forum 30*, 2 (Apr. 2011), 593–602. doi:10.1111/j.1467-8659.2011.01882.x. 5

[KKM09] KALAIDJIAN A., KAPLAN C. S., MANN S.: Automated landscape painting in the style of bob ross. In *Proceedings of the Fifth Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging* (Aire-la-Ville, Switzerland, Switzerland, 2009), Computational Aesthetics'09, Eurographics Association, pp. 115–122. doi:10.2312/COMPAESTH/COMPAESTH09/115-122. 9

[KM13] KELLY L., MARX D.: Vangobot. http://vangobot.com, 2013. January 10th, 2013. 3

[LFT05] LI F.-F., , FERGUSON R., TORRALBA A.: Recognizing and learning object categories. *A short course at ICCV 2005* (2005). 2

[Lit97] LITWINOWICZ P.: Processing images and video for an impressionist effect. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 407–414. doi:10.1145/258734.258893. 4, 5

[Llo82] LLOYD S.: Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (Mar. 1982), 129–137. doi:10.1109/TIT.1982.1056489. 2

[LPD13] LINDEMEIER T., PIRK S., DEUSSEN O.: Image stylization with a painting machine using semantic hints. *Computers & Graphics 37*, 5 (Aug. 2013), 293–301. doi:10.1016/j.cag.2013.01.005. 3, 7

[LSTS04] LI Y., SUN J., TANG C., SHUM H.: Lazy snapping. *ACM Transactions on Graphics (ToG)* (2004), 303–308. URL: http://dl.acm.org/citation.cfm?id=1015719. 2, 3

[OBW*08] ORZAN A., BOUSSEAU A., WINNEMÖLLER H., BARLA P., THOLLOT J., SALESIN D.: Diffusion curves: A vector representation for smooth-shaded images. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 92:1–92:8. doi:10.1145/1360612.1360691. 5

[PKD05] PITIE F., KOKARAM A., DAHYOT R.: N-dimensional probability density function transfer and its application to color transfer. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on* (Oct 2005), vol. 2, pp. 1434–1439 Vol. 2. doi:10.1109/ICCV.2005.166. 2

[PKD07] PITIÉ F., KOKARAM A. C., DAHYOT R.: Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding 107*, 1-2 (July 2007), 123–137. doi:10.1016/j.cviu.2006.11.011. 2

[Poy03] POYNTON C.: *Digital Video and HDTV Algorithms and Interfaces*, 1 ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. 5

[RAGS01] REINHARD E., ASHIKHMIN M., GOOCH B., SHIRLEY P.: Color transfer between images. *IEEE Comput. Graph. Appl. 21*, 5 (Sept. 2001), 34–41. doi:10.1109/38.946629. 2

[Sec02] SECORD A.: Weighted Voronoi stippling. *Proceedings of the second international symposium on Non-photorealistic animation and rendering - NPAR '02* (2002), 37. doi:10.1145/508535.508537. 5

[SY00] SHIRAISHI M., YAMAGUCHI Y.: An algorithm for automatic painterly rendering based on local source image approximation. In *Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering* (New York, NY, USA, 2000), NPAR '00, ACM, pp. 53–58. doi:10.1145/340916.340923. 2

[TD07] TSCHUMPERLÉ D., DERICHE R.: Anisotropic Diffusion PDE's for Multi-Channel Image Regularization: Framework and Applications. *hal.inria.fr* (2007). 5

[TFL12] TRESSET P. A., FOL LEYMARIE F.: Sketches by paul the robot. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging* (Aire-la-Ville, Switzerland, Switzerland, 2012), CAe '12, Eurographics Association, pp. 17–24. 3

[TFL13] TRESSET P. A., FOL LEYMARIE F.: Portrait drawing by paul the robot. *Computers & Graphics 37*, 5 (2013), 348 – 363. doi:http://dx.doi.org/10.1016/j.cag.2013.01.012. 3

[VBTS07] VANDERHAEGHE D., BARLA P., THOLLOT J., SILLION F. X.: Dynamic point distribution for stroke-based rendering. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (Aire-la-Ville, Switzerland, Switzerland, 2007), EGSR'07, Eurographics Association, pp. 139–146. doi:10.2312/EGWR/EGSR07/139-146. 2

[Wik12] WIKIPEDIA: Frieder nake. http://de.wikipedia.org/wiki/Frieder_Nake, 2012. December 19th, 2012. 3

[Wik13] WIKIPEDIA: Tinguely art machines. http://en.wikipedia.org/wiki/Tinguely, 2013. Januar 10th, 2013. 2

[WND97] WOO M., NEIDER J., DAVIS T.: *OpenGL Programming Guide*. Addison-Wesley, 1997. 6

[YY06] YANG H., YANG C.: A Non-Photorealistic Rendering of Seurat's Pointillism. *Advances in Visual Computing 4292* (2006), 760–769. 2

[ZHT07] ZHANG E., HAYS J., TURK G.: Interactive tensor field design and visualization on surfaces. *IEEE Transactions on Visualization and Computer Graphics 13*, 1 (Jan. 2007), 94–107. doi:10.1109/TVCG.2007.16. 5

[ZZ10] ZHAO M., ZHU S.-C.: Sisley the abstract painter. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2010), NPAR '10, ACM, pp. 99–107. doi:10.1145/1809939.1809951. 2, 3

[ZZ11] ZHAO M., ZHU S.-C.: Customizing painterly rendering styles using stroke processes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2011), NPAR '11, ACM, pp. 137–146. doi:10.1145/2024676.2024698. 2, 3

[ZZXZ09] ZENG K., ZHAO M., XIONG C., ZHU S.-C.: From image parsing to painterly rendering. *ACM Transactions on Graphics 29*, 1 (Dec. 2009), 1–11. doi:10.1145/1640443.1640445. 2, 3