Erschienen in: Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging. - Aire-la-Ville : Eurographics Association. - S. 25-33. - ISBN 978-1-4503-1584-5

Feedback-guided Stroke Placement for a Painting Machine

Oliver Deussen, Thomas Lindemeier, Sören Pirk, Mark Tautzenberger

Department of Computer and Information Science, University of Konstanz, Germany

Abstract

In this paper we present and evaluate painterly rendering techniques that work within a visual feedback loop of eDavid, our painting robot. The machine aims at simulating the human painting process. Two such methods are compared for different objects. One uses a predefined set of stroke candidates, the other creates strokes directly using line integral convolution. The aesthetics of both methods are discussed, results are shown.

1. Introduction

This paper aims at presenting two things: an introduction to eDavid, our painting machine that works with visual feedback. Over the last two years we built this set-up for developing painting styles for the artistic representation of given input images. In the main part of this paper, we compare two of these styles and try to evaluate them with respect to their aesthetic properties.

eDavid is a one-arm industry robot that we modified and equipped for painting purposes (see [Deu12] and Figure 1). A camera observes the canvas, their images are compared with a given target function (usually an image, but could be a more complex representation such as a 3-d scene). New strokes are generated until the target function is sufficiently approximated by the paint on the canvas. eDavid can paint with different kinds of brushes, pencils and a number of physical colors reaching from ink to oil paint. This was our rationale to realize the set-up with an industry robot and not just a pen plotter.

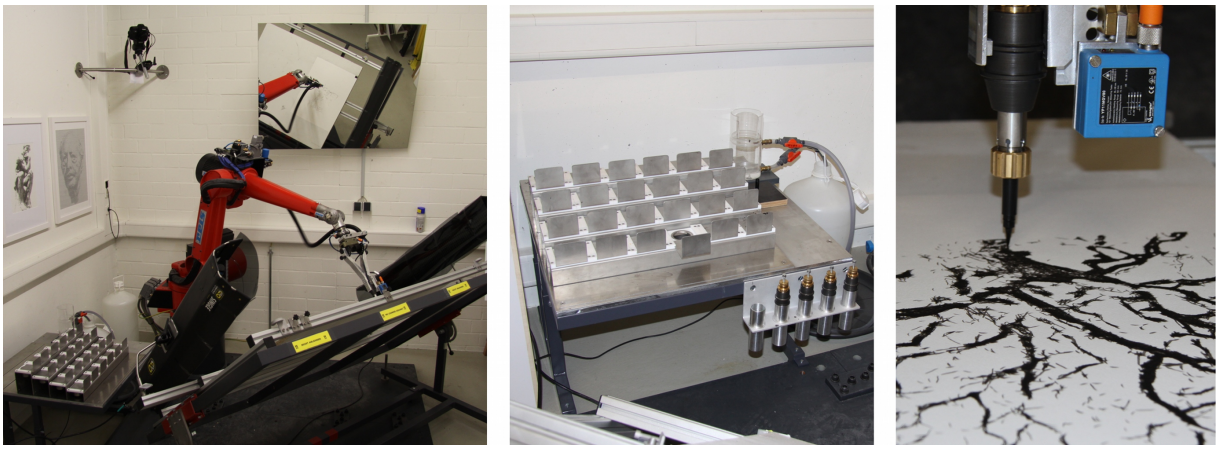
Furthermore, the feedback loop allows us to deal with the inaccuracies of brush-stroke rendering and the unpredictability of color interactions on a canvas. During our tests this behavior proved to be very important since it allows us not only to use quite inaccurate simulation techniques but can also easily be adapted to different strategies for placing strokes. This lets us realize various painting styles.

The whole eDavid experiment aims at approximating the manual painting processes by a machine, we want to find out to what extent we are able to produce artistically looking paintings. In art history it is also well known that physical limitations, e.g. interactions between ink and canvas, influence the formation of styles. We are looking for new forms of visual representations that are especially suited for painting machines; also we want to find out how to introduce high-level semantic information into the process. In recent years methods for image understanding developed a lot, so painting machines of the future could "know" what they draw and automatically adapt their painting strategy.

Since eDavid approximates the target by a sequence of strokes, we have to adapt methods for painterly rendering to work within a feedback environment: based on the target function and the currently distributed paint on the canvas a sequence of new brush strokes has to be computed that, if realized by the robot, creates a better approximation of the target by the canvas. In this sense greedy optimization is performed.

In this paper we compare two techniques for such stroke placements. In both cases we assume the target function to be a gray-scale image, thinned black ink serves as paint. An in depth exploration of color (and other media) is left for future works; however, a first result is shown in the last section. Using black ink simplifies the setup and the needed comparisons between target and canvas: color calibration can be simplified and we do not have to take care about the 3-d structure of the paint layers. Nevertheless, our framework is already able to work with color; this is why we mention color issues at some places.

The visual quality of the two algorithms depends on the target image. For some objects a set of short and straight strokes (we call them **stroxels**) is better suited due to their ability to create different visual appearances than a set of curved strokes and vice versa. After reviewing related works and describing our painting machine we compare their ability to represent different classes of objects. Results are given and discussed, future works are outlined.



(a) (b) (c)

Figure 1: *System setup of the painting machine: a) robot with canvas and camera; b) repository for colors and brush tools (front row), in the back brush washing facility; c) brush tool and distance sensor.*

2. Related Work

The vast majority of publications in the field of non-photorealistic rendering and computational aesthetics work on *simulating* the painting process. Since we deal with a real machine, we firstly cite methods for painterly rendering that are applicable within a feedback loop. Then we refer to drawing machines and robots that artists use for producing different kinds of paintings.

Painterly rendering was introduced in 1990 by Haeberli [Hae90] who created artistic images using the mouse as a virtual brush. Hertzmann [Her98] later published a general approximation scheme for painterly rendering that starts with a rough approximation of a given image that is refined by smaller and smaller brush strokes in subsequent steps. We use this idea for representing images with a set of predefined stroke directions and lengths (stroxels). In contrast to Hertzmann we therefore limit ourselves to a fixed set of lengths and widths.

While the above algorithm just represents the colors of the image, for many styles it is also important to conserve salient image features such as edges or object boundaries. Hays and Essa [HE04] introduce a set of layers that refine a painterly representation at important places. Collomosse and Hall [CH02] use a mapping of strokes to geometric elements of different width according to the image salience. Both algorithms work well for NPR in general but are not easy to implement within our sequential feedback loop since they reorder strokes.

Kang et al. [KLC07] compute image gradients, smooth the resulting vector field and use it to track important image features by Line Integral Convolution (LIC). The method enables them to create smooth outlines even for low quality input. We use their method to produce image-guided strokes, in contrast to their approach we do not just draw outlines but try to represent the whole grey-scale content of the image by such strokes.

An important aspect, especially when colors are used, is the ordering of colors. Northam et al. [NIK10] explore different strategies such as painting lighter colors over darker ones or vice versa to represent features optimally. Since we currently use colors only to a limited extend, we do not alter the order but will take this into account in the future.

In the past, a number of machines were built to create drawings mechanically. After early attempts in the 19th century, Jean Tinguely (1925-1991, see [Wik12c]) created a number of sophisticated machines that were able to create complex stroke patterns. In contemporary art, Harold Cohens Aaron (see [Coh12]) is the most famous robot painting project, ongoing for many decades now. However, the artist did not focus on faithfully representing image content but to realize abstract computer-generated artworks. In the early times he built and used a sophisticated painting machine, but recently he moved to ordinary printers.

Early pioneers of computer graphics used plotters (e.g. Frieder Naake, cf. [Wik12a]) and also robots (e.g. Ken Goldberg, [Wik12b]) to create art works, numerous others could be listed here. Today, a number of artists uses such machines, but typically their main purpose is to create abstract and artistic paintings. Ben Grosser [Gro12] and Holger Baer [Bae12] are typical representatives. Zanelle [Arm12] by v. Armin is a specialized plotter to create pop-art like paintings and also somewhat more realistic portraits. An interesting painting machine is Vangobot [KM12], also a specialized plotter, that uses a paint mixing hardware to create color variations. To our knowledge, none of the mentioned approaches uses a feedback loop for optimization.

In contrast to most of the above approaches we also do not primarily aim at creating art but to explore the algorithms behind human painting and to describe art processes by means of visual optimization. Therefore an important aspect of our system is the visual feedback loop that is described in the next section.

3. System

As shown in Figure 1, eDavid consists of a number of components. The robot itself is equipped with a specialized picking device for grasping our brush tools. Five different brushes can be used in parallel by the system (see Figure 1(b), front row).

The colors are stored in a repository, so far we are able to use 24 different colors. The robot accesses a color container by mechanically opening the cover plate and dipping the brush into the color. A cleaning facility (Figure 1(b), background right) is needed when changing the paint color of a brush.

Since painting with brushes needs precise interaction we measure the surface characteristics of the canvas using a distance sensor which is mounted on the robot arm. Besides working with (slightly) curved canvases this allows us also to compensate for mechanical tolerances of the robot while moving the brush over the canvas. Such tolerances are in the sub-millimeter range but still visible for image styles with long strokes and fine brushes. Due to restrictions in the sensor (color dependency of the precision) we so far measure the distance only before we start painting for the entire canvas.

The visual feedback of the system is created by a Canon EOS 5D Mark II SLR with a 21MPixel sensor and a fixed 50mm lens. This provides us with a resolution of about 1mm on the canvas. Two specialized fluorescent tubes with polarization filters are used to illuminate the canvas, also the SLR is equipped with a polarization filter. Selecting the polarization direction perpendicular to the canvas avoids specular highlights of the paint color.

3.1. Software Setup

The robot is controlled by a assembler-like language, therefore we built a server application that controls the machine and accepts XML commands. Most commands are plotter-like instructions such as pen selection and drawing, but also specialized commands for measuring the canvas and the handling of brushes and colors have been implemented.

A second server is used for the camera. The Canon SDK allows us to control all necessary functions from the computer. Images are created in XYZ color space and are calibrated by using geometric calibration patterns and color sheets. The geometric calibration is within the range of a pixel, the color calibration so far in the range of 5%, which is sufficient for our current applications but has to be enhanced in the future.

4. Optimization via Visual Feedback

A typical application connects to the robot server and the camera server. Based on the target function and stroke placement strategy, the application creates a number of new strokes. The strokes are realized by sending them to the robot server; after painting, the camera server is invoked to obtain the canvas image.

For producing paintings, a number of practical constraints have to be taken into account. Colors react differently to overdrawing when they are still wet. To let them dry we avoid painting on the same place within a given time interval. Colors such as inks are filled into specialized brush pens that allow the robot to draw continuously without dipping the brush into an ink container. For others, such as oil color, only short strokes can be realized after dipping. These characteristics are stored by the robot server in brush and paint color profiles.

4.1. General Optimization Strategy

In each iteration, a number of brush strokes is generated to minimize the difference between canvas and target image. For the computation of the best approximation, the application has to anticipate the effect of painting a new stroke on the canvas. In our optimization program we implement this by using OpenGL graphics using a brush texture and opacity values that have been determined for each color. A stroke path is generated and the color application is simulated for all pixels under the stroke. Based on this simulation the quality of the stroke is computed, it determines how effective the stroke minimizes the difference between canvas and target. For a large number of stroke candidates the quality is computed and the best candidates are realized by the machine.

Predefined stroke candidates: A simple implementation of this strategy is to pre-define a set of stroke candidates. Such candidates are typically short stroke paths in different orientations. For a position within the image all candidates are translated to that position, the quality is is computed and the stroke with the highest local quality is selected and stored. For a large number of random positions this is repeated, the strokes with the globally highest quality are drawn within an iteration step.

Figure 2 shows the application of a static stroke set for the portrait of a woman. A set of 180 pre-defined stroke paths of the same length and width but different orientations was given. After optimization, the selected strokes approximate the given gray scale image quite well. In subfigure (c) the paper was purposely crimped to introduce an error to the realization by the robot. Strokes on the right side of the portrait are drawn with larger line width since the brush is closer to

(a) (b) (c)

Figure 2: *Painting with predefined stroke candidates: a) input image (original: Dominik Fusina, www.flickr.com) ; b) approximation with a pre-defined set of 180 strokes in different directions that are applied to positions in the painting that have the highest quality. c) approximation with introduced realization error (thicker strokes are painted on the right side of the face), the process adapts to this and reduces the stroke density within this area.*

the canvas here. Due to the visual feedback loop the machine adapts automatically to this error and the final gray-scale values are similar to the undisturbed version.

Dynamically generated strokes: For a given position in the image, a stroke can also be generated dynamically from the image content. As suggested by works on painterly rendering (cf. [Her98]) a strategy for directing strokes is to draw them perpendicular to the image gradient. For each position a path is created, the effect on the canvas is simulated and the corresponding quality is computed, candidates with highest quality are realized. Both methods will be described below.

4.2. Computing Stroke Quality

A number of factors influence the quality for a stroke candidate. These factors are not only responsible for the style of the result, but also for the painting strategy, i.e., the order in which strokes are painted to form the final image.

In general, strokes can only be applied at places where the color difference between target image and canvas can be lowered by applying a brush stroke with the selected paint. For gray-scale input this is equivalent to the gray-scale difference of all pixels under the stroke being larger than the opacity of an additional layer of paint, in the chromatic case this has to be measured for all color channels.

We compute the difference $d(T,C)$ between canvas image C and target image T and select the regions with sufficiently high image differences for the given color i. These regions, we call them $d_i(T,C)$, are our candidate regions for adding new brush strokes.

As mentioned above, for the practical realization and for implementing different styles, some additional factors have to be considered:

Overdrawing: Often it is not intended to paint wet-on-wet, the colors may mix in an unpredictable way or the paper might even crimp due to too much ink. Therefore regions where strokes have been applied are avoided for a predefined and color-specific time duration. This is done by intersecting $d_i(T,C)$ with a map R that shows recently painted strokes. The resulting map $M_i = R \cap d_i(T,C)$ specifies the regions where new strokes can be placed.

Homogeneity: The pixels under a stroke candidate sc within M should have a large homogeneity $h(sc)$ or a small color variance, rsp., to be represented effectively by a single color. This automatically selects strokes perpendicular to the image gradient since in this direction there is a higher probability of having small color variances. Stroke directions of many painting styles are set according to this rule.

Orientation: Many painting styles furthermore prefer a uniform orientation of strokes, at least in areas where the image gradient is not too large. Examples are crosshatchings as well as impressionist or expressionist styles. The orientation of the strokes can therefore be weighted by a function $a(cs)$ that, e.g., prefers strokes in horizontal and vertical direction.

Values for orientation and proximity are typically in conflict and have to be balanced against each other. For some styles a direction field could be given that lets strokes follow a pre-defined pattern.

Proximity: The color (the amount of ink) that is needed to alter the pixels under a stroke candidate towards the intended color in T should be approximated optimally by one of the given paint colors. So we compute the average color of those pixels and determine the closest paint color. The comparison is done on the basis of the color hue, the difference in hue $c_i(sc)$ is an additional factor for the quality of the stroke.

Our standard method to determine stroke quality in a color i is to combine the above-mentioned factors and find a path within M_i that maximizes:

$$q(sc) = w \cdot h(sc) + (1-w) \cdot a(sc) + c_i(sc), \qquad (1)$$

with $w \in [0..1]$ being a weight factor. More complex functions could perform a non-linear balancing between $h(sc)$ and $a(sc)$. In the following section we describe two methods that rely on this function.

5. Realization of two Painting Styles

As mentioned above, in this paper we compare two styles to represent a target image: pre-defined stroxels and dynamically generated longer strokes. After we introduced the two styles in the previous section we now describe their practical realization to produce robot paintings. Both have different representational abilities and are well suited for different objects. We will discuss this in the next section.

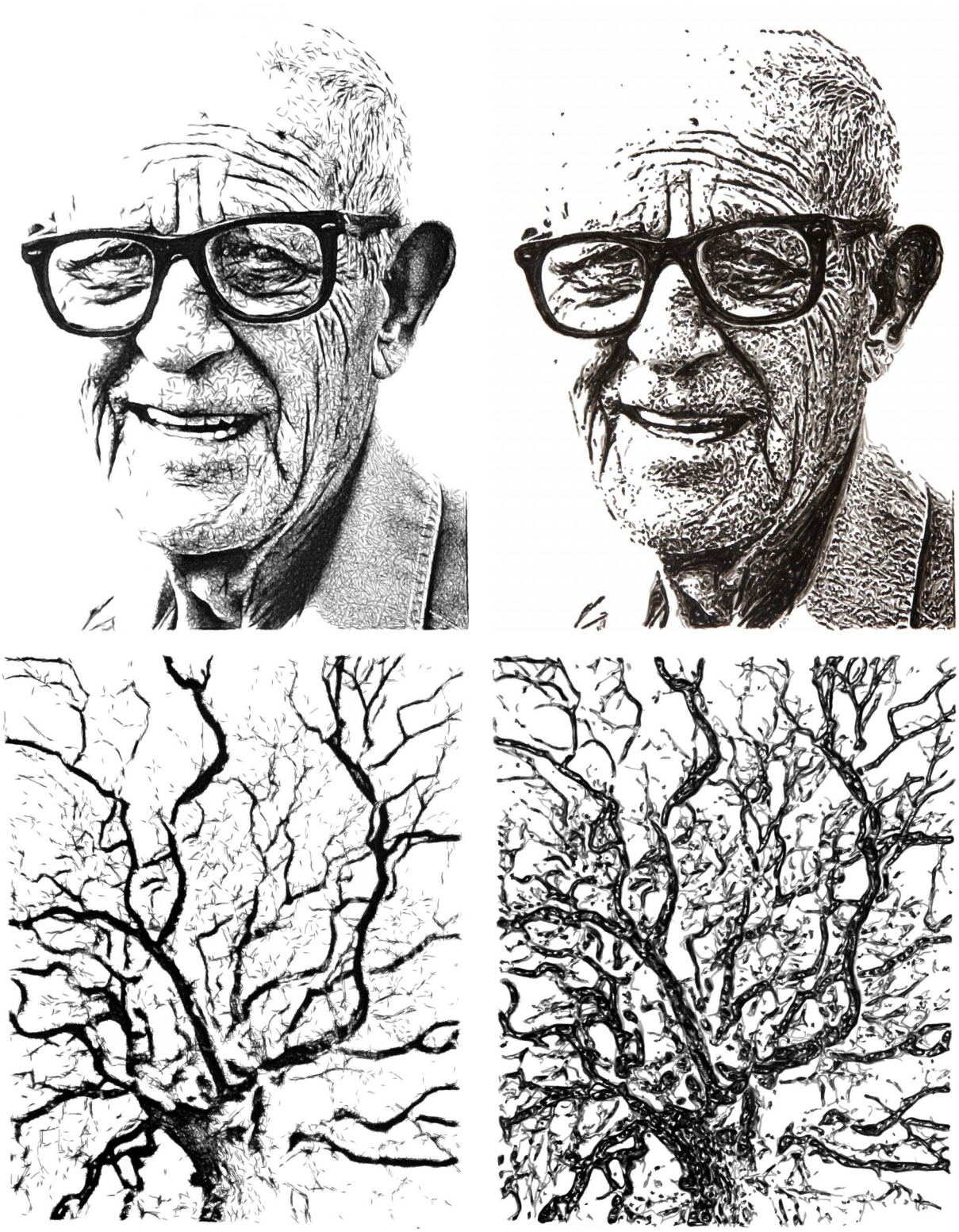


Figure 3: *Objects drawn with pre-defined paths (left) and dynamically generated strokes.*

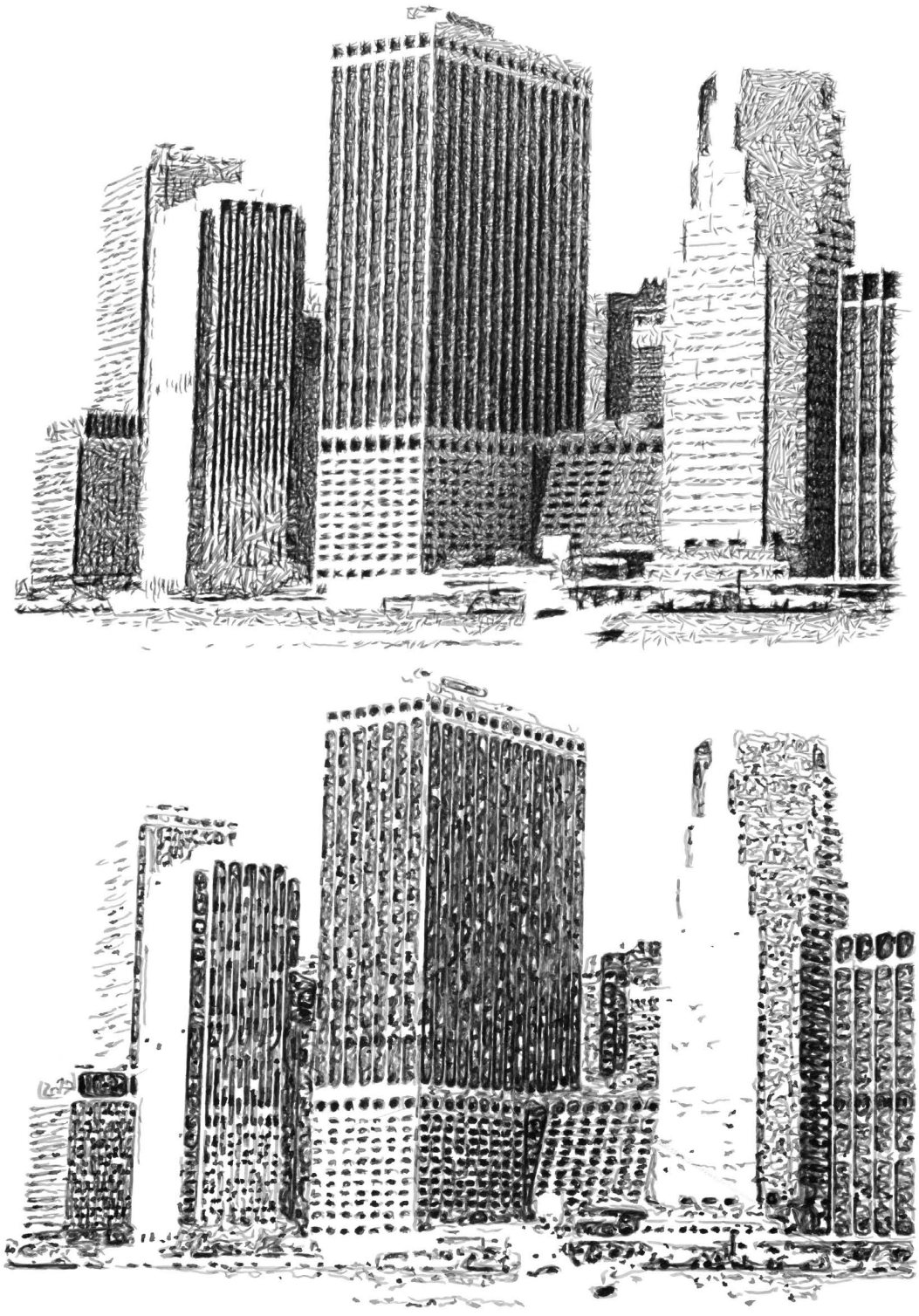


Figure 4: *More technical scene drawn with pre-defined paths (upper line) and dynamically generated strokes.*

5.1. Pre-defined Strokes

Our pre-defined set of paths for this style has three different lengths (5,10 and 20mm) and three widths (2mm, 3mm, 4mm) in 60 different directions each. This results in 540 paths that we test for thousands of random positions in M_i. The stroke candidates with the highest quality are selected and painted. We take care that they do not overlap by using a paint map R.

Upper and lower left painting of Figure 3 show objects painted with this method. Within each iteration 300 strokes were painted. The strokes are determined in sets of five strokes each. These sets are the top stroke candidates from 1000 random positions according to Eq. (1) with $w = 0.5$ and $a(cs) = 0$ (no orientation preference). Sixty of these sets are painted in a single iteration. If a stroke was selected, its simulation is added to the maps R and M_i to prevent the ongoing simulation from overdrawing strokes within the current iteration.

5.2. Dynamically generated Strokes

In a second method we compute strokes dynamically on the basis of the image information and its gradient. The image gradient is computed using a Sobel operator, subsequently it is filtered to allow larger and smoother strokes (cf. Kang et al. [KLC07]). This gradient is used to produce stroke paths that later guide the brush strokes.

In each iteration of the visual feedback loop, a large number of initial random positions is generated. For each position a path is generated using Line Integral Convolution (LIC) of the gradient vector field (see Cabral et al. [CL93]). This lets the strokes follow what Kang et al. call "edge tangent flow". We use a Runge-Kutta integration method of fourth order and a step size of one pixel to produce the paths. Uniform regions can be filled with pre-defined directions or by computing a distance transform [ST94].

Since with this method many paths are directed towards the edges of the image, we store the already drawn paths in a separate file. We test if a new path overlaps existing ones and if it overlaps it will be clipped. If the remaining stroke is too short it will be deleted.

In a second step the remaining part of the path is tested for rendering. We take M_i and compute the stroke quality for a stroke with maximum width. Typically the stroke will overlap with already drawn parts and therefore possibly extend outside of M_i, which is penalized in the quality function. The stroke width is gradually reduced and the quality is determined again. The width with maximum quality is added to a candidate list. The robot realizes the candidates with the highest quality. The method automatically prevents the machine from overdrawing, therefore no additional overlap test is applied here.

5.3. Practical Considerations

Both styles were drawn with a specialized brush pen, a mixture between a brush and a pen. Its fiberglass brush is very robust and thus convenient for our application. We extended its ink cartridge to be able to draw many thousand strokes without refilling it. This allows us to paint 2000-3000 strokes per hour. Unfortunately, higher speeds are prohibitive since the acceleration may cause unintended spurting of ink on the canvas and reduces the accuracy.

The used ink was thinned by a factor of 1:15 to have the ability of gradually darkening the canvas. Interestingly, we had many problems with highly thinned ink because it has the tendency to dissolve and also to block the pen brush, which then dries out. All images have a size of 40-60cm and where drawn with approx. 30.000-40.000 brush strokes. Painting took between 10 and 15 hours depending on the type of brush strokes and the overall speed of the machine.

Using the set-up described so far, a number of practical constraints are imposed to the simulation and feedback loop of our active visual control:

Termination: For each iteration we record the number of unsuccessful stroke placements (no stroke with sufficiently high quality was found within 1000 random positions). If too many failures occur in a sequence we stop the iteration. This kind of stopping criterion seems to be much more stable than other methods such as root mean squared pixel errors (RMS) between target image and canvas. This is due to the fact that a lot of visual noise is created during our approximation leading to unstable image-based termination criteria.

Color Calibration: Since color calibration never works perfectly we have to take into account that the saturation on the canvas will possibly not reach the exact amount of our prediction. Therefore tolerances have to be added, otherwise the process would not terminate and the robot would repeatedly overdraw already saturated regions.

Camera Lens: Though we have a relatively high resolution camera, the feedback image is blurred. Small strokes are sometimes not shown exactly at the right position and thus can also introduce repeated overdraw. On the other hand the blur seems to stabilize the iteration. In his approximation scheme Hertzmann [Her98] also uses blur for distributing the introduced error. Maybe a similar effect is introduced here.

Paint Interaction: On the canvas, colors mix in a much more complex way than we can simulate on the computer. Therefore we use highly thinned colors to correct our prediction errors by repeated overdraw during the feedback iterations. Even in the gray-scale case such complex interactions have to be taken into account.

Figure 5: *First result with a color version of stroxel placement, we used ink jet colors yellow, magenta, and cyan for this realization. The input image is shown in the upper right (image: Elizabeth A. Day, Penn State SALA, Overland Partners).*

5.4. Visual Analysis

Both algorithms terminate with a valid visual representation of the object. It is hard to determine their quality by means of absolute measurements since their characteristics are too different and the approximations are too rough to use quantitative pixel-based error metrics such as RMS.

Dynamically generated strokes tend to create drawings that are too dark, this seems to be due to the fact that within stroke simulation the width is not approximated precisely enough. Furthermore, in the current version some tiny lines are not filtered out properly and cause dark spots. This has to be improved in the future.

In Figure 6 we compare a simulation of stroxels and their practical realization. Typically different strokes are drawn in our simulation and by the robot within the feedback loop; in general, however, the gray-scale values are simulated quite faithfully. Some differences can be seen though: strokes are darker than assumed, their width is larger and also their shapes do not really match our simulation. As mentioned above, our feedback loop allows the process to adapt to the so far created gray-scale values on the canvas. This limits the overall error.

A very informal investigation of members of the university did not yield a clear preference for one of these styles (please refer to [Her10] for a review of evaluation methods based on user studies). While the stroxels look more "technical", the dynamically generated strokes have more curly and "expressionistic" look that might be better suited for organic forms. However, also the buildings in Figure 4 seems to be approximated quite nicely this way.

Both styles incorporate enough "artistic" attributes to be considered "beautiful". If pre-defined paths are not restricted in their direction, the textures of technical objects such as the building are sometimes rendered too arbitrary, orientation preferences could help here.

Figure 6: *Simulation of stroxels (left) and their practical realization.*

6. Conclusion and Future Work

We presented a mechanical painting system that works with visual feedback and an optimization loop. An industry robot is used to move a brush over the canvas, a set of specialized tools was developed to enable experiments with different brushes and paint colors. Two painting styles were investigated and discussed, a number of results were presented.

A methodically interesting question for the future would be a numerical evaluation of the painting results. Appropriate statistical means have to be found to compute the quality of different artistic representations.

In the future we want to further extend our experiments to colors, a first result is shown in Figure 5. First trials with oil colors showed us that this medium is quite difficult to handle, different colors have different viscosity, the handling of brushes is much more complex. Furthermore, the mixing of colors has to be improved, a solution similar to VangoBot [KM12] could help here. Nevertheless, we consider our set-up as a good test bed for many experiments, in the future we will also invite artists to explore the space of creative possibilities opened by eDavid.

References

[Arm12] ARMAN P. V.: Zanelle. http://www.vanarman.com/, 2012. March 13th, 2012. 2

[Bae12] BAER H.:. http://www.holgerbaer.com/, 2012. March 13th, 2012. 2

[CH02] COLLOMOSSE J. P., HALL P. M.: Painterly rendering using image salience. In *20th Eurographics UK Conference* (June 2002), Eurographics Assoc., pp. 122—128. 2

[CL93] CABRAL B., LEEDOM L. C.: Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), SIGGRAPH '93, pp. 263–270. 7

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

[Deu12] DEUSSEN O.: eDavid, a Painting Robot. http://graphics.uni-konstanz.de/, 2012. 1

[Gro12] GROSSER B.:. http://bengrosser.com/, 2012. March 13th, 2012. 2

[Hae90] HAEBERLI P. E.: Paint by numbers: Abstract image representations. In *Computer Graphics (Proceedings of SIGGRAPH 90)* (Aug. 1990), pp. 207–214. 2

[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. 2

[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. 2, 4, 7

[Her10] HERTZMANN A.: Non-photorealistic rendering and the science of art. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2010), NPAR '10, ACM, pp. 147–157. 8

[KLC07] KANG H., LEE S., CHUI C. K.: Coherent line drawing. In *ACM Symposium on Non-Photorealistic Animation and Rendering (NPAR)* (Aug. 2007), pp. 43–50. 2, 7

[KM12] KELLY L., MARX D.: Vangobot. http://vangobot.com, 2012. April 30th, 2012. 2, 9

[NIK10] NORTHAM L., ISTEAD J., KAPLAN C. S.: Brush stroke ordering techniques for painterly rendering. In *Computational Aesthetics 2010 Eurographics Workshop on Computational Aesthetics in Graphics Visualization and Imaging Victoria British Columbia Canada May 2830 2009* (2010), Eurographics Association, pp. 59–66. 2

[ST94] SAITO T., TORIWAKI J.-I.: New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications. *Pattern Recognition 27*, 11 (1994), 1551 – 1565. 7

[Wik12a] WIKIPEDIA: Frider Naake. http://de.wikipedia.org/wiki/Frieder_Nake, 2012. March 13th, 2012. 2

[Wik12b] WIKIPEDIA: Ken Goldberg. http://en.wikipedia.org/wiki/Ken_Goldberg, 2012. March 13th, 2012. 2

[Wik12c] WIKIPEDIA: Tinguely art machines. http://en.wikipedia.org/wiki/Tinguely, 2012. March 13th, 2012. 2