Hybrid Filter Blending to Maintain Facial Expressions in Rendered Human Portraits

Mahdi Rezaei*, J. Lin, and R. Klette

Department of Computer Science
The University of Auckland, Auckland, New Zealand
E-mail: m.rezaei@auckland.ac.nz
*Corresponding author

Abstract: Artistic rendering of human portraits is different and more challenging than that of landscapes or flowers. Issues are eye, nose, and mouth regions (i.e. facial features) where we need to represent their natural emotions. Shades or darkness around eyes, or shininess at nose tips may negatively impact the rendering result if not properly dealt with. Similarly, a lighter colour around the mouth region caused by lighting might produce some disturbing artefacts. The proposed computerized method attempts to be adaptive to those sensitive areas by utilizing a face analysis module. First, the program detects main facial segments and features. Then it utilizes a blending of various filtering parameters aiming at an adequate final portrait that represents the subject's original facial expression, while still supporting a non-photorealistic artistic rendering as the perceived impression.

Keywords: Non-photorealistic rendering; artistic filter; pointillism; Glass pattern; curved-strokes style; facial features detection

Reference to this paper should be made as follows: Mahdi Rezaei, Juan Lin, and Reinhard Klette, 'Hybrid Filter Blending to Maintain Facial Expressions in Rendered Human Portraits', *Int. J. Arts and Technology*, Vol. x, No. x, pp.xxx–xxx.

1 Introduction

"*Excellence is in the details. Give attention to the details and excellence will come.*"

Perry Paxton (1845 - 1933)

Photorealistic rendering aims at creating digital images that look like photographs taken by cameras. In contrast, *non-photorealistic rendering* (NPR) produces images of simulated worlds, in styles that are different to realism; see Mignotte (2003). An artistic filter implements a non-photorealistic rendering technique for transforming a given image with some 'artistic' effects. See Figure 1

Figure 1 A photo (*left*) and an example of a digital NPR (*right*) of this photo, as publicly available on `www.cs.rutgers.edu/ decarlo/abstract.html`.

for an example. Implemented effects aim at emulating styles followed by painters, such as impressionism or cubism.

Digital NPR brings together art and image processing; see Gooch and Gooch (2001). Nevertheless, a real painter is normally better at determining what visual style has the best artistic effect, since creativity itself cannot be easily (if at all) simulated by a computer.

NPR of a flower, a building, or a landscape is considerably easier than that of a human face. An NPR program easily fails on facial features. Human faces are 3-dimensional (3D) objects, and portrait renderings are 2-dimensional (2D). Determining a 'good' transformation from a 3D object into the 2D plane comes with many challenges (e.g. in terms of interpreting details). Paying attention to facial details can make resultant images unique and outstanding. Regarding the art emulation, generated output images should be 'as close as possible' to the real artwork (and defining a proper metric is an interesting task).

Paying attention to facial details in NPR programs is quite important as facial features are crucial to identify personalities. We would like to preserve each person's unique features, and at the same time we want to achieve artistic rendering effects. A program may have difficulties in understanding why a shade around one eye should not lead to a different representation compared to the other eye, or a specularity on a nose should not lead to bended 'brush strokes' around the specularity, or to preserve the general outline of the mouth.

The paper proposes an NPR approach that combines options for creative artistic rendering based on prior work by Rezaei (2011) and Valente and Klette (2010) with a new focus on human portrait particularities, resulting altogether in a particular rendering style.

The paper is organized as follows: Section 2 discusses recent research and related work in the field. Section 3 provides a detailed description for implementation of three painting styles, namely *Curved-and-Strokes*, *Pointillism*, and *Glass Patterns*. In Section 4 we discuss on Haar-like features for automated eye and nose detection. Section 5 focuses on painting considerations for the mouth region. Section 6 introduces a technique for non-uniform bordering around the canvas. In Section 7 we apply our filter blending technique to gain our final outcome of painting. Section 8 provides insights into experimental results. Section 9 concludes.

2 Related Works

A variety of painterly rendering techniques was developed in the early 1990s, starting with pioneering work by Haeberli (1990), who introduced computerized paintings defined by an ordered collection of strokes described by sizes, shapes, colours, and directions. The proposed method tried to simulate an impressionistic style.

Hertzmann et al. (2001) described a two-phase framework called *image analogies*. In the first phase (the *design phase*), the authors used a pair of images, one original image and one filtered image as the training data. Then, during the second phase (the *application phase*), followed by texture synthesis, a learned filter is applied to new target images to produce similar filtered images.

Chen et al. (2004) proposed a system that used example-based methods to emulate artistic sketching of human portraits. The system was divided into two subsystems, one focused on the face and the other one focused on the hair part. Attention to facial details was achieved during the global face phase. A global face model was designed to analyse how artists place face elements within the canvas when starting their sketches.

Brooks (2007) presented a technique to render portraits by using mixed media NPR painting filters. The input image is first segmented into regions based on the degree of local details and face detection, followed by processing each region using user-defined NPR filters. Filtered image sections are then combined into output images by taking advantages of gradient information in the original images.

DiPaola (2009) provided a parametrised approach to generate portrait images by taking advantages of approximating a knowledge domain for painterly rendering of human portraits. The knowledge domain uses fuzzy knowledge rules gained from interviews with a number of oil-portrait painters and traditional reference information. The data in the knowledge domain can be projected into an n-dimensional space and can be accessed at a different semantic level. The system would be able to generate a range of art style images by changing corresponding variables.

Papari and Petkov (2009) proposed to replace natural textures in an input image by synthetic painterly textures that were generated by means of a continuous Glass pattern, while the geometric structure of the rendered image was controlled by local gradient directions in the input image.

Valente and Klette (2010) developed a triangular graphical user interface (GUI) that allows the user to select a mix of different painting styles according to the position of cursor in the triangle; see Figure 2. (For an animated demonstration of this GUI, see `www.youtube.com/watch?v=EMLjhRbTOpk`.) Each corner of the triangle represents one style, and any point inside the triangle means a weighted combination of two painting style. An extension of this work has been briefly discussed in Pratthi (2010) by focusing on the artistic rendering of human portraits. It was noted that facial features, such as eyes and noses, are distorted if the artistic filtering is not further constrained.

Kim et al. (2010) proposed an automated system to emulate hedcut illustration drawing style, which appears mainly in the *Wall Street Journal*. Hedcut illustration drawing style tries to simulate the look of engraving and woodcuts as printed in old-style newspapers. The system uses isophotes for representing local shape and

Figure 2 Triangular GUI for filter blending by Valente and Klette (2010).

shading in hedcut illustrations, as well as a weighted constraint based on edges and isophotes for importance mapping and depth perception. The rendering process is quite time-consuming, taking about 3.7 minutes to process a 500×477 pixel image by a 2.4 GHz Core 2 processor.

Zhao and Zhu (2011) presented an example-based approach, to render human portrait paintings from photographs. The authors took brush strokes from portrait templates that were done by artists to 'paint' new human portraits. The system stores a dictionary of various types of faces as active portrait painting templates. The method was proposed for conquering potential issues caused by large brush strokes for sharp 3D structures and human beings' acute visual perception of human faces, and to improve 'painted' faces by using more appealing colours.

3 Painting Styles and Filter Design

First, we briefly recall the simulation of three painting styles, called *curved-strokes*, *pointillism*, and *Glass patterns*, and the work of Valente and Klette (2010). Then, in Section 4, we describe facial feature detection (i.e., of eye, nose, and mouth regions). In the next step, we apply a *smoothing* filtering technique for eye and nose regions to generate more pleasant and sensible artistic renderings. Finally, for the mouth region, we apply a contour strengthening approach to produce a more vivid and characteristic mouth.

3.1 Curves-and-Strokes Filter

This filter generates an output image with an appearance of hand-painted brushes. The method generates strokes on a canvas parametrised by an appropriate brush size, with the possibility of varying in different areas of an image. The algorithm performs rendering in three layers, and uses the following information from the input image:

- image size (width W_i and height H_i),
- face detection (centre location (x, y)) and detected face width (W_f), and
- size of detected face in proportion to the entire input image (in $p\%$).

Details of face detection and localization (supporting size calculation) are described in Section 4.

Based on the p value and inspired by Hertzmann et al. (2001), three brush sizes $R_1 = 0.02 \cdot W_i$, $R_2 = 0.01 \cdot W_i$, and $R_3 = 0.006 \cdot H_i$ are selected; those brush sizes ensure a satisfying appearance. The method starts by generating strokes on the canvas with the largest brush (R_1). From a painter's point of view, a painting would appear to be more artistic and professional if we use mainly larger brushes, and smaller brushes only for sensitive regions in order for addressing details. Therefore, at first, we render the whole image with the largest brush R_1. After applying a convolution of the created rendering with the source image, the algorithm determines the mean colour difference to the source image. If the difference is greater than a threshold T, then the next brush (i.e. smaller) also renders the canvas, otherwise it remains unchanged:

```
function paint(sourceImage, Rm ... Rn)
        {
           canvas = a new constant color image
           // paint the canvas
                for (i=n ; i > m ; i--)
                from largest brush to smallest do
                  {
                   // Apply Brush Ri
                   Canvas = sourceImage * I(Ri);
                   // Threshold in painted layer
                   Threshold (canvas, SourceImage, Ri, Ti)
                        If (Ti <= T) then
                            Exit (loop);
                  }
           return canvas;
        }
```

By applying the above algorithm, benefits are a more artistic appearance, as well as a logarithmic decrease in computational cost; see Figure 3 for a result. The overall output looks pleasant, however, the eyes seem abnormally deviated from their natural appearances, and the nose is also a bit bended.

There is a simple way to fix these problems: use a smaller brush size for the whole painting, to sustain more similarities to the reference image. This simple solution can solve the eye and nose problems, but the overall level of artistic impression will decrease. A better approach is actually to keep rendering the

Figure 3 Original portrait (*left*, photo in public domain) processed using curved brush strokes (*right*).

images at first with larger-size brushes, and then use smaller-size brush strokes *just* for eye regions, as well as applying some image enhancement techniques in the nose area. For the mouth region, in the resultant curved-strokes image we see a lack of features compared to the reference image.

In order to investigate reasons for previously described issues, we have to look closer into the brush-stroke generation process. During the rendering process, the reference image is first divided into multiple grids according to the size of brush strokes. For each grid square, the colour of a pixel in this square is compared with the colour at this pixel in the reference image, and a new stroke is added if the total sum of those differences is above a pre-defined threshold T.

Threshold T is the main factor to determine how close the finished painting will be compared to the reference image in terms of details. Around the mouth region, especially on contours, colours on these pixels are always much lighter than the pixels on the lips. As a result, the error value has a big chance to be less than the threshold T, and as a result, no new stroke will be added for smaller, subtle details. In Section 4 we provide more details.

3.2 Pointillism

For emulating pointillism, it is helpful to apply Seurat's theoretical work; see Yang and Yang (2008). There are two main reasons that make computerized pointillism to be attractive. The painting process of pointillism is highly exhaustive, so automating a major component of the (so far) manual painting task is very desirable. Secondly, it is not difficult to approximate the basic pointillism style by 'painting' simple point-like strokes. For emulating pointillism, inspired by Valente and Klette (2010), the image is generated in three layers, each of which distributes the points in different ways.

The background layer fills up the canvas with large points which have minimal colour distortion. The middle layer adds smaller dots of stronger colours to areas where the brightness of the picture differs from the original by a threshold T_p. The final layer provides some edge enhancement to prevent details so far hidden by the larger dots. At each layer, dots are painted in a random order by using a Z-buffer, where each point's Z-value is initialized by a random number. As an advantage, compared to the curved-strokes method, this filter for eyes and noses uses only two parameters that need to be altered. Values that need to be defined are the

Figure 4 Original image (*left*, photo in public domain) and a naive emulation of a pointillistic style (*right*) showing eye and nose issues (e.g. 'holes').

brush size R and a decreasing rate of the size for each layer (default $d = 25\%$). For the mouth region, as introduced above, the middle layer is painted based on the algorithm that finds the error between the canvas and the reference image, and then adds new brush strokes at locations where the error is large enough. Here, we use the same technique for the correction of the mouth region as in curved-strokes renderings. Furthermore, the algorithm for the foreground layer detects the brightest and darkest points in a neighbourhood of some size. If there are more points closer to dark than to bright, a new stroke will be painted on the brightest point. Our technique is to blend some percentages of grey-scale pixels in the mouth region, adjusting the stroke distribution for detailed areas.

For simple rendering, uniform circles, each of constant colour, are used in the implementation. The filter emulates Seurat's painting style by breaking the image into a series of points, restricting the colour palette, and incorporating the idea of divisionism. Figure 4 shows a pointillistic style rendering. In this sample we have multiple faces, so we consider W_f to be the average width of all the detected faces.

3.3 Glass Patterns

This filter generates impressionistic images in the style of Van Gogh; inherent geometric structures are modelled as Glass patterns and transformed into the target image via a random point set. The geometric transformation uses information about the image's local gradients to create an effect of impressionistic whirls around contours in the image. The used algorithm is a modified version of the work by Papari and Petkov (2009). Similar to curved-strokes and pointillism, the implementation is based on OpenCV and defines two parameters of Glass patterns (GP), the step size h and the standard deviation, based on the proportion of the face size to the image size ($\rho = \frac{W_f}{W_i}$), and a grid histogram deviation, respectively. The initial step in the Glass pattern method is edge preserving smoothing (EPS), which eliminates small textures from the input image while maintaining the main object contours.

We denote the output image of EPS by I_{EPS}. The next step is the generation of synthetic painterly textures (SPT) which simulate oriented brush strokes. Assume $\nabla_\sigma I_{EPS}$ to be the scale-dependent gradient map of I_{EPS}, which is created by the convolution of I_{EPS} with the gradient of the Gauss function g_σ:

$$\nabla_\sigma I_{EPS} = I_{EPS} * \nabla_{g_\sigma}$$

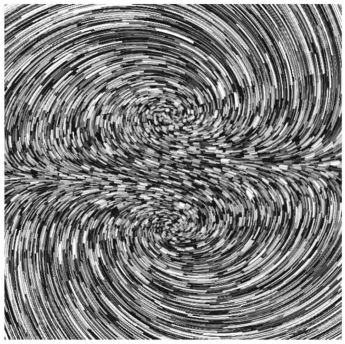

Figure 5 Sample of a whirled Glass pattern.

Figure 6 Original portrait (*left*, photo in public domain). Naive Impressionistic-style emulation (*right*) showing distortion in the eye regions.

Finally we apply a typical Glass pattern such as shown in Figure 5 at random locations of the image with different strengths of a *whirl parameter* S between 0 to 1, and a *whirl value* θ_w that can be a random or a fixed number; in the latter case we chose equals to $\pi/4$, $\pi/2$, $3\pi/4$, or π. The implemented filter gives a nice look of an impressionistic oil painting by shifting pixels around the canvas in a pattern of impressionistic whirls. The Glass pattern is successful in providing 'pleasant' visual effects for small images, but, because colour values rotate around some fix points, it does not apply well on large portraits, and again some refinements in eye areas are necessary. See Figure 6 *right*.

For the mouth region, the technique we described above does not work well for Glass pattern emulation. The reason is that the sampled Glass pattern is placed randomly, and gradients of Glass patterns do not work with the gradients of original features, thus, sometimes the mouth contour can look different to the reference one, depending on the random placement of the Glass patterns. One approach to make the mouth look closer to the reference could be to crop-out the mouth region, and then save the gradients on mouth-contour pixels. When placing the Glass patterns, the system can match the Glass pattern that has the minimum difference between gradients of Glass patterns and gradients in the original image onto the mouth areas. However, this idea has not yet been fully explored. So far we only demonstrate a partial solution for mouth improvements in experiments for the Glass pattern method.

4 Haar Classifiers for Face, Eye and Nose Detection

For the facial component detection, we apply Haar-like feature matching based on Viola and Jones (2001) and our previous work in Rezaei and Klette (2011). Figure 7 shows a cascaded structure of weak classifiers, each one maps a few Haar wavelet features (less than 10) into a 2D image processing concept, in correspondence with common grey-scale representation.

The detection process of a query object (e.g. a face) is in accordance with main characteristics of the object and the grey-value distributions in dark or light regions of a feature that models expected intensity distributions. For example, in

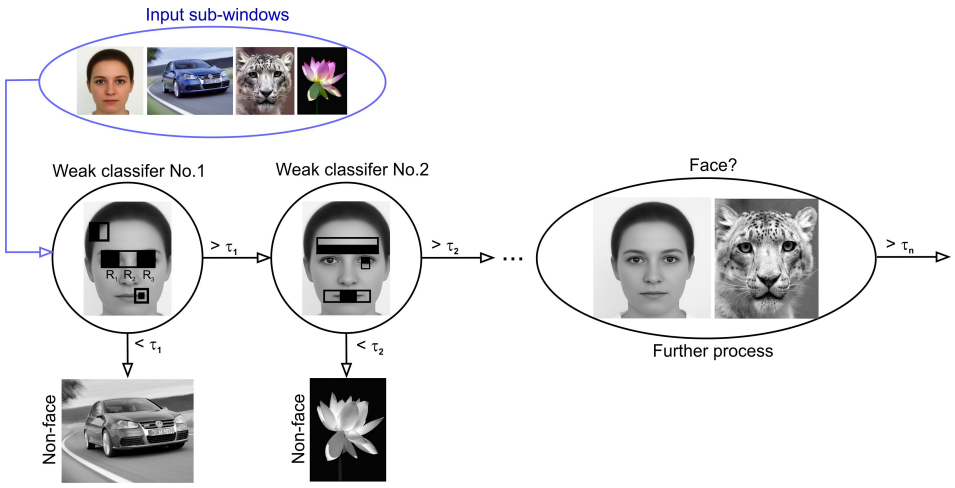

Figure 7 Haar-feature matching inside the weak classifiers.

Figure 7, left classifier, the middle feature relates to the idea that in all faces there are darker regions of eyes compared to the bridge of the nose.

4.1 Integral Image

We use *integral images* as introduced in Viola and Jones (2001) to confirm the presence or absence of given Haar-like features. For any pixel $p = (x, y)$, the *integral value* is defined as

$$I(p) = \sum_{1 \le i \le x \wedge 1 \le j \le y} P(i, j)$$

which is the sum of all pixel intensity values $P(q)$, where pixels $q = (i, j)$ are all pixels located above and left of p.

Consider the corners $p_1, \ldots, p_4$ of a rectangle R; the sum of all pixel values for R is equal to

$$I(R) = I(p_4) + I(p_1) - I(p_2) - I(p_3)$$

Applying positive and negative weights $\omega_i > 0$ for dark and light regions of Haar-features, we can define a final integral value. For example, for a feature defined by three regions (see Figure 7 on the left) we have that

$$I(F_k) = \omega_1 \cdot I(R_1) - \omega_2 \cdot I(R_2) + \omega_3 \cdot I(R_3)$$

4.2 Cascaded Classifiers

In order to classify object regions of a face such as eyes or the nose, an AdaBoost machine learning algorithm has been used to select appropriate Haar-like features among thousands of possible Haar-like features for each stage of the classifier. Then, a sliding window searches the image area to find appropriate matches in different image locations, for different size and shapes of Haar-like features. As shown in Figure 7, feature matching with a similarity threshold greater than τ_k defines further progress to the next weak classifier.

Heavily weighted features come first to eliminate obvious non-face image regions as early as possible. The initial classifier simply rejects non-object (non-face) regions if the given fundamental features do not exist. For more information see Rezaei and Klette (2011).

5 Modifications at the Mouth's Contour

After face, eye and nose detection, by using the approach described above, the mouth region can already be estimated roughly just based on universal human face proportions. Although a mouth can also be detected by a trained Haar classifier, however, considering the other three detected facial segments, in conclusion, the mouth is expected in a defined area of the face. Thus, it is not necessary to implement an extra classifier as to save computation time.

5.1 Used Face Model

Pardew (2006) discusses in his book the anatomy of the head and head proportions, actually for beginner artists to have a 'construction guide' for head drawings. These face proportions are common facts, and can be utilized to crop-out the mouth region based on face detection. Figure 8 illustrates face-proportion analysis. As illustrated, painting beginners could start their human face paintings from a sketch of an ellipsoid. Eyes, nose, and mouth normally lay at certain positions on this ellipsoid. Focusing on the mouth region, the mouth's middle line is at 1/3 of the distance between the nose base and the chin. Vertical lines start at the centre of pupils, lined up with the corners of the mouth.

The centre point of the face is calculated by a cascaded classifier, and a radius is calculated according to a scale value. According to the geometrical face model, we define the cropped mouth region as *Region of Interest* (ROI), the top left point as (x, y), the centre point as (c_x, c_y), and the radius as r. The mouth region can now be obtained roughly as follows:

$$x = c_x - \frac{2r}{5} \quad \text{and} \quad y = c_y + \frac{r}{3}$$
$$W_m = r \quad \text{and} \quad H_m = \frac{r}{2}$$

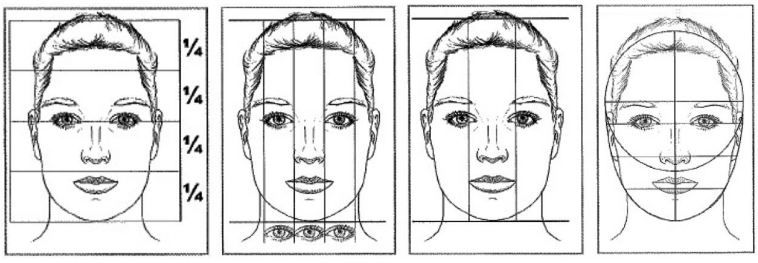

Figure 8 Construction guide for placing facial features, as published by Pardew (2006). *From left to right*: The face is divided vertically into four quarters. The eyes are about one eye-width apart. The mouth is usually inside the width between the pupils.

for mouth width W_m and mouth height H_m.

Since mouths may have different states (e.g., open, or nearly closed), the width and height defined above are actually larger than those of a scaled mouth window, in order to crop a general mouth region even for irregular mouth shapes (or states).

5.2 General Description of the Used Procedure

As discussed above, a total error E can be used for reducing numbers of strokes, producing a less detailed rendering. For example, we may use

$$E = \sum_{k=1}^{m} |P_{original}(k) - P_{processedImage}(k)|$$

for all the m relevant pixels $P(k)$ in the region of interest, and to minimize the total error value E. For example, the region of interest might be defined by the pixels near to the mouth's contour.

To achieve this, at first the mouth region is converted into a grey-scale image, then it is smoothed by using a Gaussian blur, then we apply a Laplace operator for detecting the mouth's contour at zero-crossings.

To retain the characteristics of the resulting edges in the rendering process, edge enhancement is done on these pixels by using morphological operations. Before weighted edge pixels being blended into the original image, noise is reduced by using a median filter. The weighted pixels that are added into the original image, before the actual artistic rendering, are actually hard to identify visually. The role for those added pixel values is to put more weights on the error function E as described before for influencing the creation of new rendering strokes. A heuristic method is applied to control the percentage of blended pixels, just to make sure that there are not too many strokes added around the mouth's contour.

5.3 Gaussian Blur and Median Filter

Gaussian blur is a common technique that is used in image processing, typically for reducing image noise or to reduce details (e.g., as a preprocessing step, or for calculating a scale space in computer vision). Mathematically, Gaussian blur is a convolution of the image with the Gauss function

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-(\frac{x^2+y^2}{2\sigma^2})}$$

We use Gaussian blur as a preprocessing step for reducing noise around the mouth, thus also reducing the brightness of extremely light pixels.

The median image operator is a non-linear image filter, widely used for preprocessing in order to remove noise, and preserving edges at the same time. The median filter considers a local neighbourhood around a pixel and replaces its image value by the median value in the neighbourhood (e.g., it performs very well in removing salt-and-pepper noise). We apply the median filter after running an edge operator for detecting the mouth's contours, thus eliminating small artefacts but retaining the mouth's contour. Figure 9 shows examples of Gaussian blur and median filtering in the mouth region.

Figure 9 The original image is as in Figure 3. *Top, left to right*: Cropped mouth region, its grey-scale image, and the grey-scale mouth region after Gaussian blur. *Bottom, left to right*: Result of applying a Laplacian edge operator, resulting contours after bit-wise inversion and histogram equalisation, and contours after applying the median filter.

5.4 Edge Detection

There are different edge detection techniques available in image processing, for example the Laplacian-of-Gaussian, or the Canny edge operator (both available in OpenCV). The Canny edge operator depends on two user-defined parameters, used as thresholds for the hysteresis procedure. Due to variations in brightness and image details, those parameters need to be selected adaptively for given images.

Figures 10 and 11 show edges detected by the Canny edge operator for different threshold values. We see that those edges are not consistent and suffer from lack of details. The Laplacian edge operator, on the other hand, demonstrated a more robust and better performance on mouth regions in this case, also in cases where pixel values only change gradually from dark to bright. Let G_σ be the Gauss function. The Laplacian of G_σ is then defined by

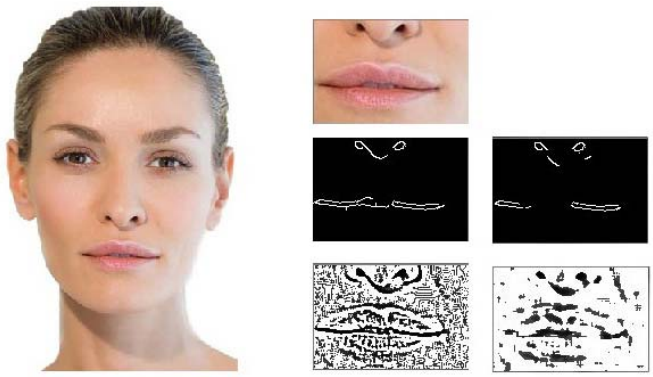

Figure 10 *Left*: Reference image. *Top to bottom, left to right*: Cropped mouth region. Canny edge detection with thresholds 60 and 195, or 120 and 165. Mouth contours obtained with LoG operator and subsequent median filter.

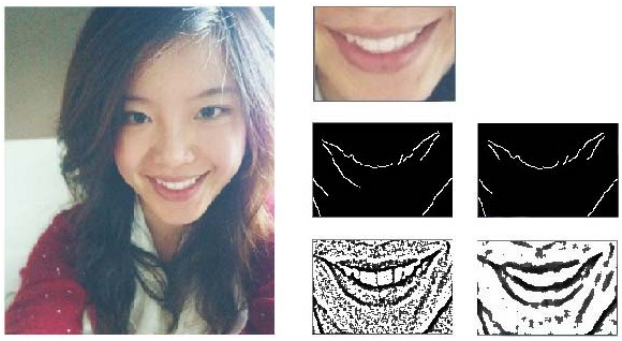

Figure 11 *Left*: Reference Image. *Top to bottom, left to right*: Cropped mouth region. Canny edge detection with thresholds 65 and 155, or 75 and 155. Mouth contours obtained with LoG operator and subsequent median filter.

$$\begin{aligned}\Delta G_\sigma(x,y) &= \frac{\partial^2}{\partial x^2}G_\sigma(x,y) + \frac{\partial^2}{\partial y^2}G_\sigma(x,y)\\ &= \frac{x^2+y^2-2\sigma^2}{\sigma^4}e^{-(\frac{x^2+y^2}{2\sigma^2})}\end{aligned}$$

The Laplacian-of-Gaussian (LoG) edge detector is a convolution of an image region with ΔG_σ.

Obviously, the shape of a mouth may vary a lot. For example, the shape of a 'big smile' is absolutely different from a densely compressed mouth; we would like to have an operator that works well with any shape of a mouth. The LoG operator appears to be a reasonable choice. Figures 10 and 11 also show edges detected with the LoG operator.

6 Non-uniform Borders of the Canvas

Paintings done by artists usually have non-uniform borders. Non-uniform borders may contribute to a 'relaxed' or 'more pleasant' appearance. There are many options for making non-uniform borders. For example, we may add extra pixels around the rectangular image for padding borders and adding strokes on that area to emulate the randomness. As another approach, we may remove some of the painted strokes close to the border area of the image. Tests show that some important details might be lost when random pixels are removed around the border. Therefore, we prefer adding of extra border pixels (border strokes) around the original image area. For explaining our technique, we use the following two offset parameters

$$B_1 = \frac{1}{15} * W_i \quad \text{and} \quad B_2 = \frac{1}{4} * B_1$$

where W_i is the width of the given image. For rendering, we use a canvas C_1 which extends the original $W_i \times H_i$ canvas of the given image by adding B_2 all around, thus a new size of $(W_i + 2B_2) \times (H_i + 2B_2)$. Now, strokes or dots are also drawn in the extended canvas. We extend the C_1 canvas again; this new canvas C_2 extends now the original canvas by B_1 all around. The rendering of C_1 is now

Figure 12 Non-uniform borders around a rendered human portrait.

a sub-rectangle of the C_2 canvas, and we render the border area with the given rendering algorithm using the white pixels as input. Therefore, we also create white strokes into the C_1 canvas. These padded borders produce an irregular border. Figure 12 shows an example.

7 Filter Blending

In this section, we describe the process of replacement of facial features (filter blending) and image enhancements in sensitive segments of the portrait as rendered initially. The steps of the process are illustrated in Figure 13.

Consider an input portrait $P(x, y)$ with $0 < x \leq W_i$ and $0 < y \leq H_i$. Before applying an artistic filter, we run face, eye, and nose detection as discussed in Section 4. Then we save an unchanged copy of these detected rectangular regions into windows

$$E_{1L}(x_1, y_1, w_1, h_1),\ E_{1R}(x_2, y_2, w_2, h_2),\ \text{and}\ N_1(x_3, y_3, w_3, h_3)$$

as original left eye, right eye, and nose, respectively, where x_k and y_k identify the upper left corner of the windows, and w_k and h_k are width and height. Similarly, we define the original mouth region as

$$M_1(x_4, y_4, w_4, h_4)$$

by following the approach in Section 5.

Next, we apply artistic rendering filters using a default parameter set A to the original image $P(x, y)$; we obtain the new image $Q(x, y)$. Now we apply the artistic rendering filter with the modified parameter set B (as discussed above) on those three regions of $E_{1L}(x_1, y_1)$, $E_{1R}(x_2, y_2)$, and $N_1(x_3, y_3)$, as well as contours and edge enhancements in mouth regions. The resultant regions which are smoothed eyes, nose, and mouth will be saved in $E_{2L}(x_1, y_1)$, $E_{2R}(x_2, y_2)$, $N_2(x_3, y_3)$, and $M_2(x_4, y_4)$. Parameters in set B cause less distortion than parameters in set A.

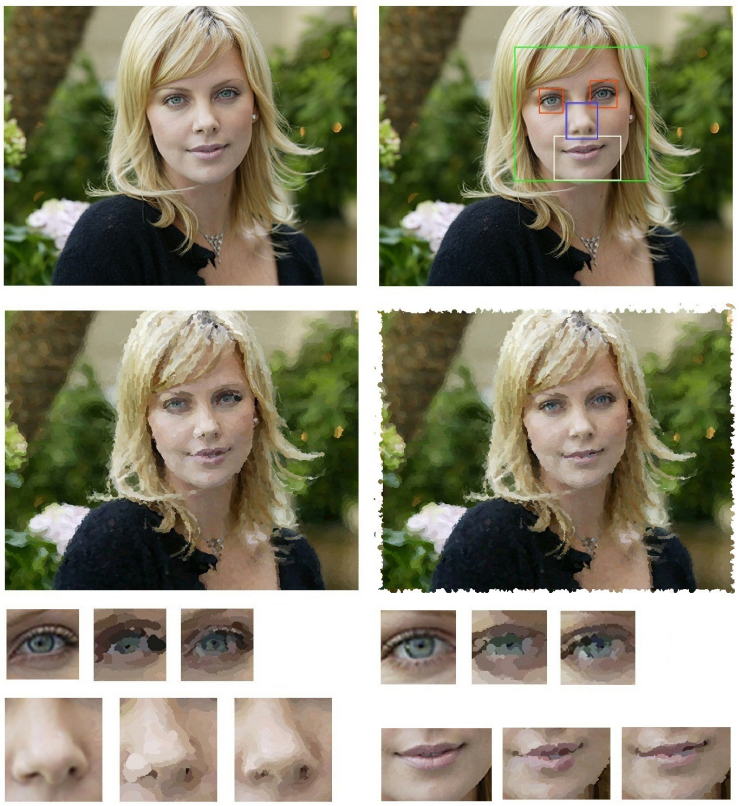

Figure 13 Curves and strokes rendering. *Top to bottom, left to right*: Input image. Detected face, eye, and mouth regions, and blurred nose region using a Gaussian aperture width of $p = 11$. Initial output with max brush size $R_M = 7$ and min brush size $R_n = 3$. Final rendering result with non-uniform borders. Eye and nose modifications with $R_{f_M} = 0.85R_M$ and $R_{f_n} = 0.33R_n$. Mouth improvement.

Finally, we replace the regions of eyes, the nose, and the mouth in $Q(x, y)$ by convolving the smoothed ones at the same locations defined by (x_k, y_k), for $k = 1$ to 4. Image $R(x, y)$ is the final result.

8 Experimental Results

Figures 14 to 18 illustrate three different styles of artistic rendering. We examined our method on different human portraits (differing by age, facial expression, skin colour, number of shown people, photo or painting), to understand the effectiveness and robustness of our approach for varying subjects. The generated images show in general vivid improvements in eye and nose regions, and often also in the mouth region, with some variations depending on rendering styles. The selected Figures 14 to 18 show original images, initial rendering results, and final results, also illustrating variations for facial features.

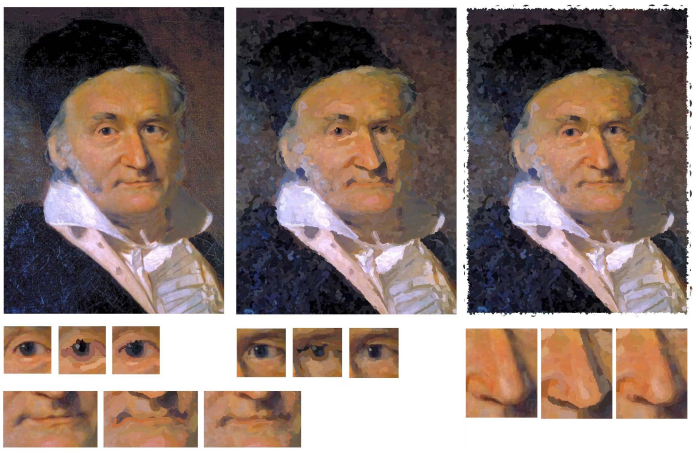

Figure 14 Curved-strokes rendering of a portrait of Carl Friedrich Gauss (image of this painting is in the public domain). *Left to right*: Original painting, initial output with maximum brush size $max = 27$ and minimum stroke strength $min = 3.5$, and final result with eyes after modification, with brush sizes $max = 6$ and $min = 2$, and nose modification with brush sizes $max = 6$ and $min = 2$ after blurring.

Figure 15 Pointillistic rendering. *Top to bottom, left to right*: Input image. Detected face, eye, nose, and mouth regions. Initial output with brush size $R = 5$ and stroke strength $s = 1$. Final rendering result with non-uniform borders. Two eye modifications with $b_e = 0.58R$ and $s_e = s$. Nose modification with $b_n = 0.7R$ and $s_n = 0.8s$. Mouth improvement.

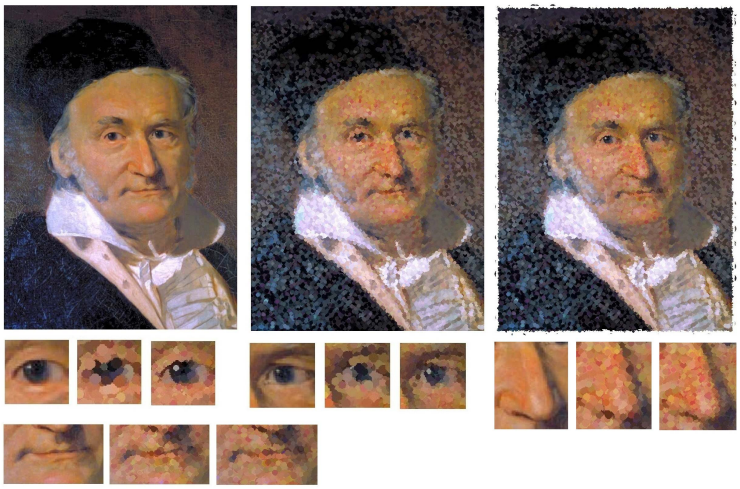

Figure 16 Pointillistic rendering on the portrait of Carl Friedrich Gauss. *Left to right*: Original portrait, initial output with brush size $R = 8.4$ and stroke strength $s = 0.4$, and final result, with eyes after modification with $b_e = 0.58R$ and $s_e = s$, and nose modification with $b_n = 0.7R$ and $s_n = 0.8s$.

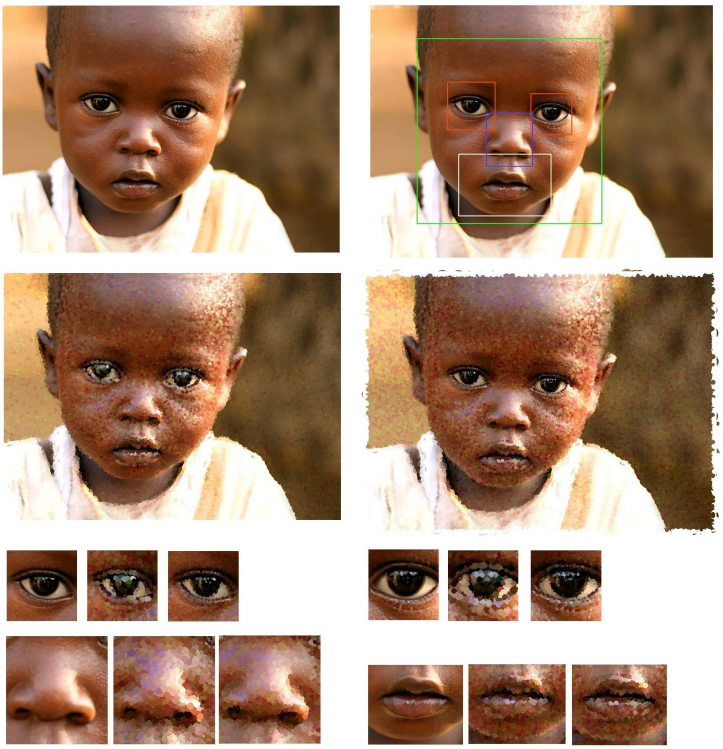

Figure 17 Pointillistic rendering. *Top to bottom, left to right*: Original image. Detected face, eye, nose, and mouth regions. Initial output with brush size $R = 5$ and stroke strength $s = 1$. Final result with non-uniform borders after modifications.

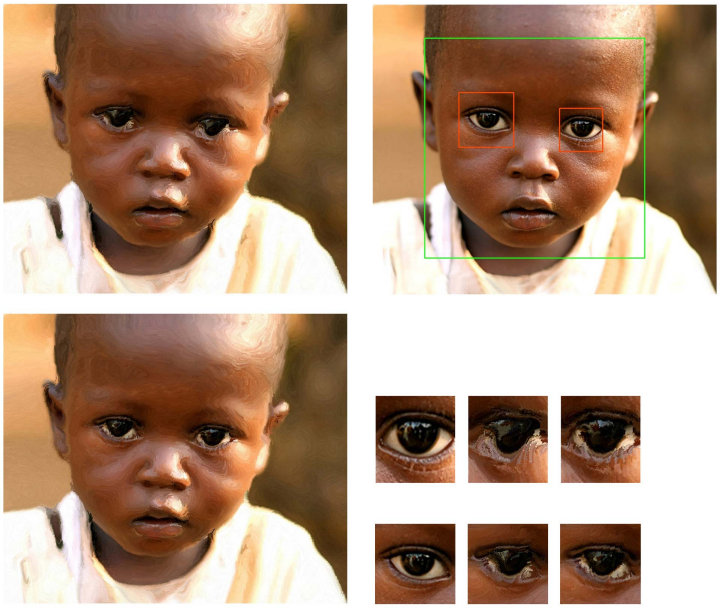

Figure 18 Glass pattern rendering. *Top to bottom, left to right*: Initial output with GP step size of $h = 0.4$, $\theta_w = \pi/2$, and $s = 0.5$. Detected face and eye regions. Final result (Note: no nose and mouth detection and modification is required for Glass pattern rendering). Eye modifications with $h_f = 0.3h$ and $s_f = 0.8s$.

9 Conclusions

The paper presents a hybrid approach of artistic rendering, facial feature detection, and final enhancements to emulate portrait painting in NPR style. Experimental results for three techniques (i.e. curved-strokes, pointillism, and Glass patterns) were satisfying in general in terms of eye, nose, and mouth improvements while keeping an overall artistic painting appearance.

For future work, such rendering and improvement methods could be expanded to other artistic styles (e.g. to cubistic style renderings). Also, the adjustment of the mouth region for Glass-pattern-based rendering still requires further work to be equally satisfying as all the other enhancement procedures.

Acknowledgments: We acknowledge availability of the NPR software provided by *Crystal Valente*, co-author of Valente and Klette (2010). Also useful information and guidance for the better understanding of artistic paintings was provided by oil painter *Mona Pouyan*, who advised on valuable details and techniques that made this research more realistic.

References

Brooks, S. (2007) 'Mixed-media painting and portraiture', *IEEE Trans. Visualization and Computer Graphics*, Vol. 13, No. 5, pp. 1041–1054.

Chen, H., Liu, Z., Rose, C., Xu, Y., Shum, H., and Salesin, D. (2004) 'Example-based composite sketching of human portraits', In Proc. *NPAR*, pp. 95–103.

DiPaola, S. (2009) 'Exploring a parametrized portrait painting space', *Int. J. Art Technology*, Vol. 2, No. 1-2, pp. 82–93.

Gooch, B., and Gooch, A. (2001) 'Non-Photorealistic Rendering.', *AK Peters Ltd. Publisher*.

Haeberli, P. (1990) 'Paint by numbers: Abstract image representations', In Proc. *SIGGRAPH*, pp. 207–214.

Hertzmann, A., Jacobs, C., Oliver, N., Curless, B., and Salesin, D. (2001) 'Image analogies', In Proc. *SIGGRAPH*, pp. 327–340.

Kim, S., Woo, I., Maciejewski, R., and Ebert, D. (2010) 'Automated hedcut illustration using isophotes', In Proc. *Smart Graphics*, pages 172–183.

Mignotte, M. (2003) 'Unsupervised statistical sketching for non-photorealistic rendering models', In Proc. *ICIP*, Vol. 3, pp. 573–576.

Papari, G., and Petkov, N. (2009) 'Continuous Glass patterns for painterly rendering', *IEEE Trans. Image Processing*, Vol. 18, pp. 652–664.

Pardew, L. (2006) 'Figure Drawing with Virtual Models: Getting the most out of Poser Figure Artist', *Course Technology PTR, online publisher*, pp. 59–63.

Pratthi, A. (2010) 'Artistic rendering of human portrait of photographs', Project report, Tamaki Transformation Program, The University of Auckland, Computer Science.

Rezaei, M. (2011) 'Artistic rendering of human portraits paying attention to facial features', In Proc. *ArtsIT 2011*, LNICST 101, pp. 90–99.

Rezaei, M., and Klette, R. (2011) '3D Cascade of classifiers for open and closed eye detection in driver distraction monitoring', In Proc. *CAIP*, pp. 171–179.

Valente, C., and Klette, R. (2010) 'Artistic emulation, filter blending for painterly rendering', In Proc. *PSIVT* , pp. 462–467.

Viola, P., and Jones, M. (2001) 'Rapid object detection using a boosted cascade of simple features', In Proc. *CVPR*, pp. 511–518.

Yang, C. K., and Yang, H. L.(2008) 'Realization of Seurat's pointillism via non-photorealistic rendering', *The Visual Computer*, Vol. 24, pp. 303–322.

Zhao, M., and Zhu, S.C. (2011) 'Portrait painting using active templates', In Proc. *NPAR*, pp. 117–124.