Image Stylization by Interactive Oil Paint Filtering *

Amir Semmo [a,1] Daniel Limberger [a] Jan Eric Kyprianidis [b, c] Jürgen Döllner [a]

[a] *Hasso Plattner Institute, University of Potsdam, Germany*
[b] *TU Berlin, Germany*
[c] *HSHL, Germany*

Abstract

This paper presents an interactive system for transforming images into an oil paint look. The system comprises two major stages. First, it derives dominant colors from an input image for feature-aware recolorization and quantization to conform with a global color palette. Afterwards, it employs non-linear filtering based on the smoothed structure adapted to the main feature contours of the quantized image to synthesize a paint texture in real-time. Our filtering approach leads to homogeneous outputs in the color domain and enables creative control over the visual output, such as color adjustments and per-pixel parametrizations by means of interactive painting. To this end, our system introduces a generalized brush-based painting interface that operates within parameter spaces to locally adjust the level of abstraction of the filtering effects. Several results demonstrate the various applications of our filtering approach to different genres of photography.

Keywords: oil paint filtering, artistic rendering, colorization, image flow, interactive painting

1. Introduction

Image-based artistic rendering received significant attention in the past decades for visual communication, covering a broad range of techniques to mimic the appeal of artistic media [1]. Oil paint is considered to be among the most popular of the elementary media because of its qualities for subtle color blending and texturing [2]. Starting with the beginning of semi-automatic painting systems in 1990 [3], stroke-based techniques that align and blend primitives on a virtual canvas have been the predominant category to simulate oil paint [4]. While their example-based texturing approach is able to provide high-quality outputs of expressive nature and great opportunities for layering, however, stroke-based techniques are usually hard to parameterize to simulate paint with soft color blendings or no visible borders—e.g., as practiced in the Renaissance era (such as *sfumato* [5]) and prevalent in many figurative art works (Figure 1). To this end, image filtering is a promising alternative approach to produce painterly looks with more subtle color blendings—in particular with the recent advancements in shape-adaptive smoothing [1], such as anisotropic diffusion [6] and shock filtering [7]. Simulating the visual characteristics of oil paint via image filtering, however, is a difficult task with respect to three main issues:

- I1 The color distribution should be optimized to conform to a global color palette while preserving contrasts of important or prioritized features.
- I2 The paint texture should be oriented to the main feature curves to mimic the way an artist might paint with a brush.

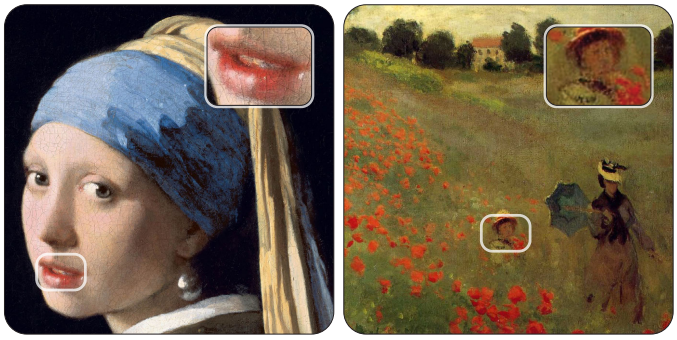

Figure 1: Oil paintings by J. Vermeer (1665) and C. Monet (1873). The artists use constrained color palettes with soft color blendings, two characteristics we simulate by our filtering technique.

- I3 The stylization process should be locally adjustable to enable creative control over the visual output.

In this work we present a technique for image stylization that employs (re-)colorization and non-linear image filtering to devise artistic renditions of 2D images with oil paint characteristics. Rather than attempting to simulate oil paint via aligned strokes [3, 8, 9, 10] or through physically-based techniques [11, 12], this work formulates I1 to I3 as sub-problems of image filtering (Figure 2). The first problem is solved by performing a recolorization, using the optimization-based approach of Levin et al. [13], with the dominant colors of the input image for quantization. This approach produces more homogeneous color distributions than local image filtering techniques and gives users more control in refining global color tones. The second problem is solved using the smoothed structure tensor [14], which is adapted to the feature contours of the quantized output, together with principles of line integral convolution [15] and Phong shading [16] to

[1] amir.semmo@hpi.de | http://www.hpi3d.de

This is the authors' version of the work. The definitive version will be published in Computers & Graphics, 2016. doi: 10.1016/j.cag.2015.12.001.

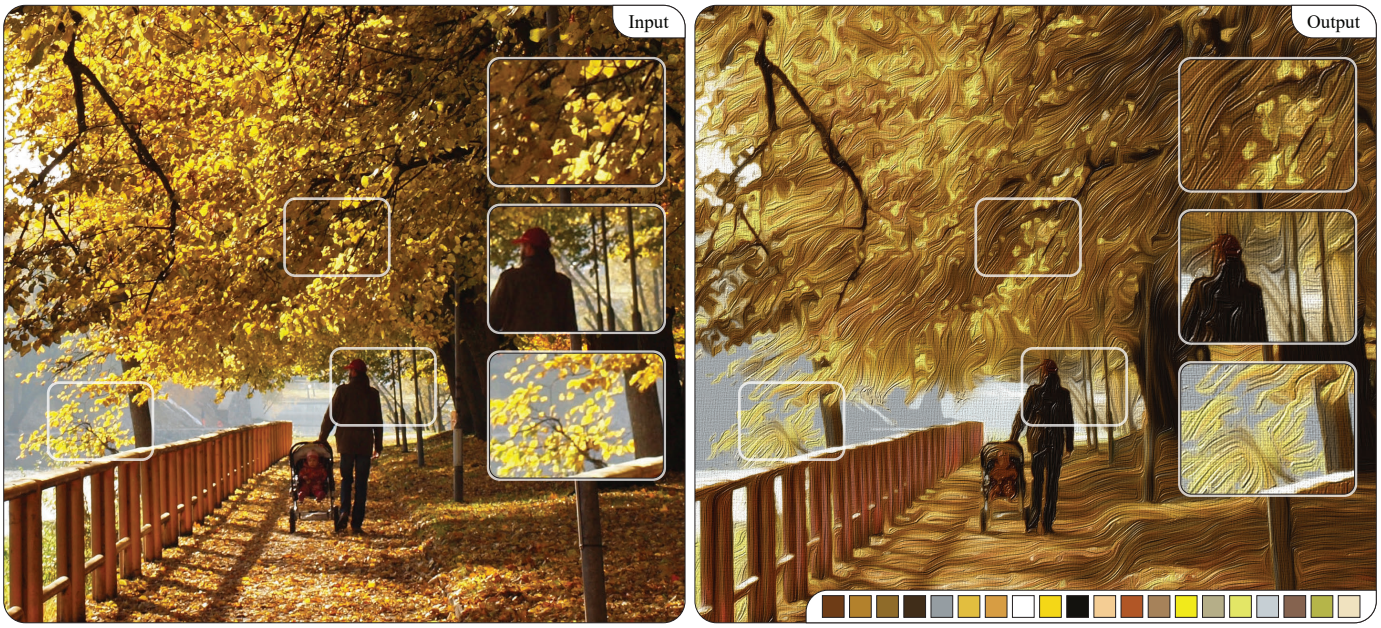

Figure 2: Exemplary application of our technique to automatically transform a color image (left) to a filtered variant with oil paint characteristics (right).

obtain a flow-based paint texture in real-time. Finally, the third problem is addressed by an interactive painting interface that implements GPU-based per-pixel parametrizations via virtual brush models to give users local control for adjusting paint directions, shading effects, and the level of abstraction. Our approach provides versatile parametrization capabilities to resemble paint modes that range from high detail to abstract styles.

This paper represents an extended journal version of the CAe 2015 paper by Semmo et al. [17]. Compared to the original paper, the major contributions are twofold: (1) we provide new methods to parametrize our local filtering effects according to image masks (e.g., derived from saliency-based metrics) and create outputs of varying level of abstraction, and (2) we expand our original algorithms towards an interactive painting system with brush tools for creative image editing. Accordingly, the remainder of this work has been restructured as follows. Section 2 reviews related work on image stylization and filtering, color quantization, and paint texture synthesis, now including topics such as brush-based painting and level of abstraction for stylized renderings. Section 3 presents the methods used for oil paint filtering, including extended methods to adjust the level of abstraction according to importance masks, e.g., using depth or saliency-based information. Section 4 proposes our interactive painting interface with brush tools to locally adjust paint configurations and the level of abstraction. Section 5 presents further results and implementation details, including comparisons to previous stroke-based techniques and an updated prospect on future work. Finally, Section 6 concludes this paper.

2. Related Work

Related work is found in the fields of image stylization and filtering, color quantization, paint texture synthesis, and brushed-based painting interfaces.

2.1. Image Stylization and Filtering

For artistic image stylization, three approaches can be distinguished: (1) stroke-based and example-based methods, (2) region-based techniques, and (3) image filtering [1]. A classical method for stroke-based stylization is to iteratively align brush strokes of varying color, size, and orientation according to the input image [3, 8, 9, 10, 18, 19, 20]. For an overview on this topic we refer to the survey by Hegde et al. [21]. Example-based rendering typically involves texture transfers by image analogies [22], a method previously used to create portraits with a painterly look [23, 24] and in neural networks to mimic painting styles [25], but which typically requires training data as input. An essential building block for region-based stylization is segmentation. Several methods based on a mean shift have been proposed for image abstraction [26, 27] and the simulation of artforms and fabrics, such as stained glass [28] and felt [29]. However, the rough boundaries of the segmented regions created by these methods would require elaborate post-processing to achieve color blending characteristics of oil paint.

To substantially modify areas or image regions, local image filtering that operates in the spatial domain may be used, which is often based on anisotropic diffusion [6]. A popular choice is the bilateral filter, which works by weight averaging pixel colors in a local neighborhood according to their distances in space and range [30], e.g., for image-based abstraction [31]. Flow fields have been used to adapt bilateral filtering [32, 33] and particle-based techniques [34] to local image structures. In this work, quantized color outputs are smoothed by flow-based Gaussian filtering to provide smooth interpolations at curved boundaries, however, we restrain from weighting in the color domain to achieve firmer color blendings.

Additional filter categories include morphological operations using dilation and erosion (e.g., for watercolor rendering [35]), and global optimization schemes for image decomposition, such

as weighted least squares [36], local extrema for edge-preserving smoothing [37], and L_0 gradient minimization [38]. However, these techniques are less suited for rendering with a constrained color palette, which instead requires filtering schemes found in color quantization.

2.2. Color Image Quantization and Abstraction

The typical goal of color quantization is to approximate an image with a relatively small number of colors while minimizing color abbreviations. Popular approaches are based on the median-cut algorithm [39], clustering using octrees [40], k-means [41], and adaptive segmentation via perceptual models [42] or roughness measures [43]. However, these algorithms may absorb colors because of their global optimization scheme, or only operate in the color space. Other approaches also consider the spatial space via local luminance mapping and thresholding [31], and in its optimization scheme to preserve image details [44], but are mainly self-organizing with respect to the derived color palette. By contrast, we propose a technique that derives colors from local image regions using a scoring system for optimal distribution, and uses the derived color palette for image quantization. At this, the optimization framework of Levin et al. [13] is parametrized to propagate seed pixels—using colors of the derived palette—to the remaining pixels at the premise that pixels in space with similar intensities should have similar colors, with an additional pass for luminance quantization. A related optimization scheme was proposed by Kim et al. [45] to seamlessly stitch uniformly-colored regions, but for the application of dequantization. The output produced by our method is then post-processed using a flow-based Gaussian filter to provide firm color blendings.

Artists are trained—beyond different painting styles and techniques—to enhance communication aspects of their emotions and ideas via principles of abstraction and highlighting. At this, the concept of *level of abstraction* (LoA) plays a major role for guiding a viewer's focus to certain image regions and improve the perception of information that are meant to be of particular interest [26, 46]. Here, a common method is to parametrize image filters according to image masks to select and seamlessly combine different LoA representations, e.g., by explicitly defined regions of interest [47, 48], saliency maps [49] and importance maps derived from edge-based hierarchies [50]. In this work, we also follow the approach of image masking to provide different LoA representations. Here, our main focus lies on the definition of parameter sets to locally control the filtering effects and their LoA, as well as the development of a modular interface to inject image-based metrics, such as feature semantics, image saliency, and the view distance derived from images with depth information (e.g., for foreground/background separation). Rosin and Lai [51] also use image salience and edge-based criteria to direct a color (de-)saturation and render images with spot colors, e.g., to make foreground objects stand out. Their technique also uses *hue* values for color quantization, however, they merely focus on three classes to guide the abstraction: dark (black), intermediate (gray) and light (white). By contrast, we seek an approach that guides the colorization via global and possibly refined color palettes that is subject to an optimization scheme to minimize color differences.

2.3. Interactive Paint Texture Synthesis

Contrary to physically-based paint modeling [12, 52], we separate the paint texture synthesis from color abstraction. Our computation is based on the smoothed structure tensor [14] and an eigenanalysis to obtain gradient and tangent information for directed smoothing similar to the work by Kyprianidis and Döllner [32]. The smoothing pass is adapted to the main feature contours retrieved via the flow-based Laplacian of Gaussian (FLoG) [53] to avoid ringing artifacts and provide outputs with adaptive detail. Similar to the work by Hertzmann [54], bump mapping via Phong-based shading [16] is used to synthesize a normal-mapped height texture that is aligned to the local feature orientations of the quantized image. The synthesis involves noise textures that are blurred in gradient flow direction to create a painting-like effect, an approach that is similar to line integral convolution [15] and is also followed in the texture-based design of tensor fields [55, 56]. By contrast, our computation is defined as a composition of ordered, parametrizable image processing steps and performs in real-time, thus it may also be injected by user-specified motions [9, 57] via painting [58].

The often tedious, error-prone process of manually tweaking image filtering effects is typically limited to a subset of global parameters. Most consumer photo-editing products only rudimentary support to adjust local parameters via masking and blending. Because our paint texture synthesis is local and fast, we propose a system for per-pixel parametrization that supports the modification and correction of (pre-)computed or intermediate filtering results. In contrast to placing dynamic or static rendering primitives [59], we extend the concept of specialized local parametrizations [60, 61] to a generalized brush-based painting within effect-parameter spaces. At this, our approach injects local parameters into image processing steps, a concept prominently used for WYSIWYG painting in the intensity [58] and gradient domain [62] to locally adjust stylized renderings [63]—which is often coupled with user-defined textures that can be interactively merged, intersected and overlapped [64]. Our generalized approach enables to aggregate single filter operations and parameters (e.g., brush sizes, smoothing kernels) to high-level brush tools to explicitly control the LoA of the filtering effects, which is exemplified for local orientation and shading corrections of paint textures (e.g., to simulate the glossiness of water features). For a compendium on the synthesis of brush strokes we refer to the work by DiVerdi [65].

3. Method

An overview of our stylization technique is shown in Figure 3. The processing starts with extracting a user-defined number of dominant colors from the input image (Section 3.1). Next, a recolorization based on the optimization approach by [13] is performed to quantize the chrominance and luminance channels using the derived color palette (Section 3.2). In parallel, contour lines are extracted from the input image via difference-of-Gaussians (DoG) and Laplacian-of-Gaussians (LoG) filtering, where the latter is used for parametrizing the computation of a flow field (Section 3.3). The computation is based on the structure tensor [14], which is adaptively smoothed according to the

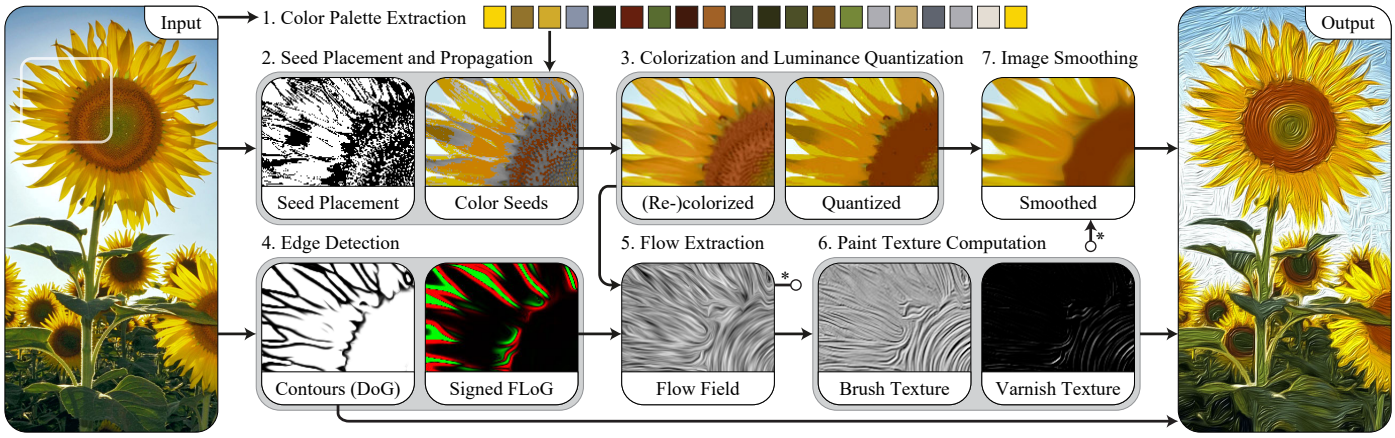

Figure 3: Overview of the different stages performed for the proposed stylization technique that automatically transforms color images (left) to filtered variants with an oil paint look (right). For a detailed description of the filtering stages the reader is referred to Section 3.

derived contour lines to avoid filtering across feature boundaries. An eigenanalysis is then used to provide gradient and tangent information for paint texture synthesis. The synthesis performs in real-time, is based on bump mapping and Phong shading [16], and produces outputs that are blended with the quantized image to compose the final result. The filtering stages are presented in more detail in the following sections, each followed by a discussion as well as methods for dynamic parametrization to achieve content-based LoA effects.

3.1. Dominant Color Extraction

To synthesize the way artists paint with a limited number of base colors, dominant (and contemporary) colors need to be derived from the input image. Common approaches use global optimization schemes that cluster similar colors and determine the base colors from the largest clusters, such as the median-cut algorithm [39]. These approaches, however, may produce false colors or absorb contemporary colors.

Our approach computes a color palette $\mathcal{P}$ with colors derived from local image regions of image I. A scoring system is applied to classify image regions according to their normalized entropy by penalizing the extraction from features with similar colors. The color extraction is performed incrementally, starting with an empty palette. Extracted colors are inserted after each iteration.

To find a new color for a given color palette $\mathcal{P}$, we seek for an image region R of minimal region score

$$\mathcal{S}(R) = \frac{\frac{\eta(R)}{|R|}}{\frac{\bar{\omega}}{|R|} \cdot \frac{\bar{L}}{|R|}} = \frac{\eta(R)}{\bar{\omega} \cdot \bar{L}} \cdot |R| \,. \tag{1}$$

$|R|$ denotes the area of the region and is used to normalize the weights: image entropy $\eta(R)$, lightness $\bar{L}$, and color distance $\bar{\omega}$. The entropy of the image is computed from the probabilities $p_R(c)$ that a binned color c appears in the image region R:

$$\eta(R) = -\sum_{c \in R} p_R(c) \log_2 p_R(c) \,. \tag{2}$$

The entropy weight is used to favor the color extraction from regions with constant color tones that preferably belong to a single image feature. At this, the probability functions are discretized using color bins of constant size. For all examples in this paper the bin size is set to 256. To favor vivid colors, the region score is weighted according to the lightness $\bar{L} = \sum_{p \in R} L(p)$ — using sRGB gamma corrected intensities. Finally, to favor colors not yet present, the region score is weighted according to the minimum color distance to colors of the current palette $\mathcal{P}$:

$$\bar{\omega} = \sum_{p \in R} \omega(p)^2 \quad \text{with} \quad \omega(p) = \min_{c \in \mathcal{P}} \Delta E(I(p), c) \,. \tag{3}$$

This way, more priority is given to extracting palettes with diverging color tones. In Equation 1, the respective weights are divided by $|R|$ to yield the normalized entropy, average lightness, and average color distance.

To find a (rectangular) region with minimum score we proceed heuristically, using the following algorithm. The steps are performed iteratively in CIE-Lab color space, n-times in total for a target palette size of n:

1. **Color Difference Mask**: To avoid recomputations in the subsequent steps, ω is precomputed via the minimal color distance (ΔE) for pixels in I to $\{c \in \mathcal{P}\}$, and the results are buffered in a color difference mask.
2. **Region Search**: Regions are computed for the horizontal and vertical scanlines of the input image. The color is extracted from the region with the better score (Figure 4):
   (a) **First Pass**: $\mathcal{S}(R)$ is computed for all regions of width $\tau > 0$ along the horizontal/vertical scanlines. The scanline $\mathcal{M}_x$ with minimal $\mathcal{S}(R)$ is determined. For all examples in this paper we set $\tau = 3$.

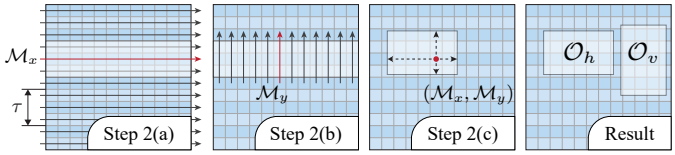

Figure 4: Schematic overview of the region selection for the horizontal pass: (a) horizontal and (b) vertical scanlines with optimal score are determined, followed by (c) region growing to select $\mathcal{O}_h$—next to $\mathcal{O}_v$ from the vertical pass.

(b) **Second Pass**: The first pass is repeated for the orthogonal scanlines, bounded by the region that is defined by the scanline $\mathcal{M}_x$. Again, the scanline $\mathcal{M}_y$ with minimal $\mathcal{S}(R)$ is selected for further processing.

(c) **Area Computation**: $\mathcal{S}(R)$ is determined iteratively for growing horizontal and vertical boundaries around the pixel position $(\mathcal{M}_x, \mathcal{M}_y)$ until a minimum value is reached.

3. **Color Extraction**: Once a region with minimum score has been identified, a representative color is extracted from the region and inserted into the palette $\mathcal{P}$. Thereby, the representative color is computed by finding a mode in the box-filtered histograms of the chrominance channels.

An example of the iterative computation of dominant colors, color difference masks, and optimal region selection is given in Figure 5. In a final step, the colors in C are sorted by their weighted count in the input image (i.e., thresholding ΔE) and color difference to previously sorted colors.

Discussion. We empirically determined $\xi = \Delta E^*_{ab} = 7$ as a good default value for thresholding, using the CIE76 formula

$$\Delta E^*_{ab} = \sqrt{(L^*_2 - L^*_1)^2 + (a^*_2 - a^*_1)^2 + (b^*_2 - b^*_1)^2} \qquad (4)$$

to compute the difference between two colors, where $\Delta E^*_{ab} \approx 2.3$ corresponds to a just noticeable color difference [66]. Our algorithm is compared to the median-cut algorithm in Figure 6. Notice how colors of single features are accurately represented and not merged with colors of other features (e.g., the butterfly in Figure 6a, the eagle's beak in Figure 6c, the guitar in Figure 6e, red tones in Figure 6f). Further, we noticed that our approach is more resistant to noise as one can observe by the green background in Figure 6c, where a clustering approach may also derive false colors. Figure 7a demonstrates this stability by an image that has been artificially corrupted with Gaussian and impulse noise, where only small changes for the derived contemporary colors are noticeable when using our approach. In addition, Figure 7b demonstrates the stability for different image resolutions, where a down-sampled image still leads to plausible palettes with stable estimates for the pre-dominant colors. Finally, we observed that the number of color extractions significantly affects if image features are accurately represented or not. To this end, one could use a metric to control the color coverage, e.g., to derive colors until the maximum and/or mean of the color difference mask (ω) falls below a given threshold. This way, more colors could be derived for colorful images without the need for content-dependent parameter adjustments. Here, we also believe that the accuracy of our algorithm can be further improved when using a generalized region growing technique to derive colors from feature-aligned (non-rectangular) regions.

3.2. Image Quantization using Colorization

Our goal is to quantize the input image I using the extracted dominant colors. We formulate this task as an optimization problem, performing a (re-)colorization [13]: Given the intensity of the input image and a number of colored seed pixels, the colors should be propagated to the remaining pixels such that pixels with similar intensities have similar colors.

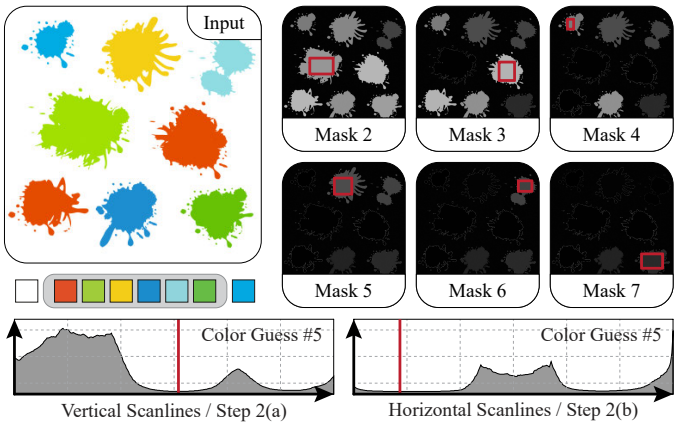

Figure 5: Color difference masks computed for an image with 8 colors (including white). The overlaid rectangles indicate the respective regions with minimum score used for the color extraction. The region scores for the fifth color guess are shown at the bottom.

Figure 6: Comparison of the median-cut algorithm to our approach for dominant color extraction. The ten dominant colors are sorted from left to right. With our approach, colors of single features are more accurately represented.

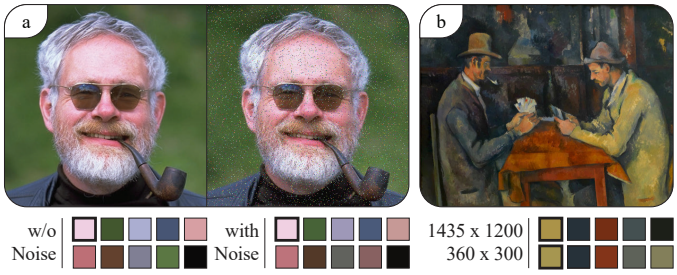

Figure 7: Stability tests of our color extraction: a) applied to an image corrupted with 2% Gaussian and 5% impulse noise, b) applied to a down-sampled image of the painting *The Cardplayers* by P. Cézanne (1892). In all cases, plausible palettes with stable pre-dominant colors are derived from the images.

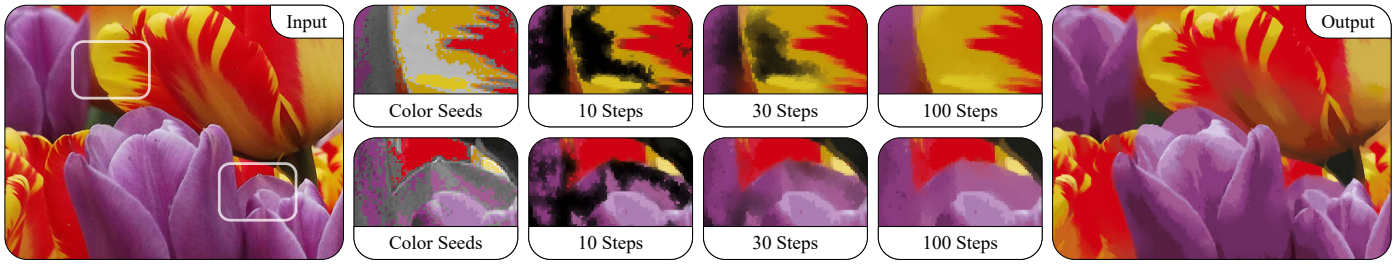

Figure 8: Image quantization using the algorithm described in Section 3.2 with a palette of 26 colors and $\alpha = 5.5$ for automatic seed placement. The optimization problem was iteratively solved with the "generalized minimum residual" method [67].

Figure 9: Comparison of our quantization method with the median-cut algorithm. Notice how our approach preserves feature contrasts (e.g., the beard) at a better scale. The smoothed output is based on a flow-based Gaussian filter [32].

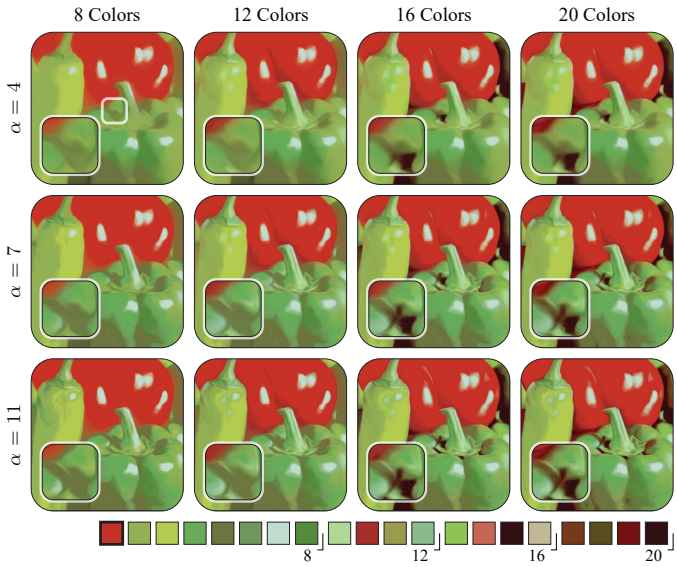

Figure 10: Example for adjusting the seed placement threshold α and the number of colors for the derived palette to adjust the LoA. The results include a post-processing step by flow-based Gaussian smoothing.

1st Pass: Colorization. The optimization is performed with the constraint that *seed pixels* $c(\mathbf{r_i})$ are set to the respective color component of the dominant color c_p in $\mathcal{P}$ with minimal distance:

$$c(\mathbf{r_i}) = \arg\min_{c_p \in \mathcal{P}} \Delta E(c_p, I(\mathbf{r})) \tag{5}$$

if and only if the minimal color distance falls below a threshold $\alpha > 0$. This replaces the interactive placement of "scribbles" described in [13]. For a given color channel C, the recolorized channel C' is then computed via the objective function

$$\arg\min_{C} \sum_{\mathbf{r}} \Big(C(\mathbf{r}) - \sum_{\mathbf{s} \in \mathcal{N}(\mathbf{r})} w_{\mathbf{rs}}\, C(\mathbf{s}) \Big)^2, \tag{6}$$
$$\text{subject to } C(\mathbf{r}_i) = c(\mathbf{r}_i) \text{ for } i \in \text{Seeds}$$

where $w_{\mathbf{rs}}$ denotes the squared difference between the luminance values of the pixels $\mathbf{r}$ and $\mathbf{s}$, and $\mathcal{N}(\mathbf{r})$ being the 8-connected neighborhood of $\mathbf{r}$. The objective function then yields a large sparse system of linear equations that is solved for both chrominance channels of the input image in CIE-Lab color space using the "generalized minimum residual" method [67] (Figure 8).

2nd Pass: Luminance Quantization. We introduce a second pass for luminance quantization, where the objective function is used to recompute the luminance channel. To this end, we reformulate the weighting function $w_{\mathbf{rs}}$ to compute the squared difference between the two *hue* values at pixels $\mathbf{r}$ and $\mathbf{s}$ of the recolorized image. The *hue* value is obtained by converting the recolorized image to CIE-LCh color space—as cylindrical version of the CIE-Lab color space. The recomputed luminance channel is then combined with the color channels of the first pass to yield a recolorized image with potential color transitions.

3rd Pass: Flow-based Gaussian Smoothing. The quantized image is post-processed by a Gaussian filter with standard deviation σ_q that is adapted to the local structure of image features, derived from the smoothed structure tensor [32]. This creates smoothed outputs at curved boundaries and, in general, a more painterly look (Figure 9).

Discussion. Examples using our method are presented in Figure 8, Figure 9 and Figure 10. Contrary to schemes based on global optimization methods, we observed that our approach produces outputs with better feature contrasts but higher variances

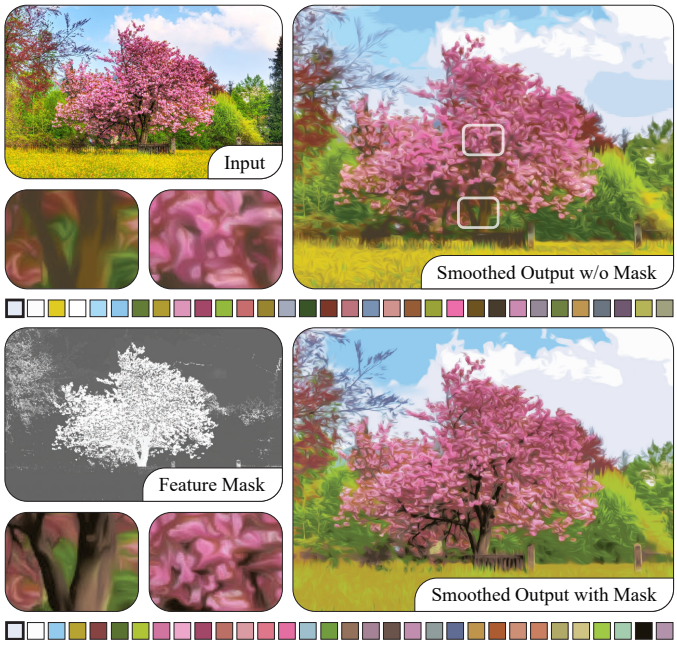

Figure 11: Using image masks to weight the color extraction according to important or salient image features. Notice how the derived color palette for the modified weighting (bottom) represents the tree more accurately than the standard procedure (top), while the environment becomes more abstract. The salient region detection is based on the algorithm of Cheng et al. [68].

to the original images (e.g., Figure 9). This behavior can be explained by deriving colors from local image regions—e.g., instead of prioritizing a global variance minimization—and the usage of color differences as weighting factors in the scoring system introduced in Section 3.1. The threshold for automatic seed placement should be set greater or equal to the threshold used for the color extraction to use all colors of the derived palette. Here, we empirically determined $\alpha = \xi = 7$ as a good default value. The threshold may also be set lower to initially place fewer color seeds and thus induce more color blendings, or may be set higher to result in more details and crisp boundaries between image features. In addition, the thresholding may be combined with reducing the number of colors in the derived palette to further adjust the LoA. Figure 10 illustrates the mutual impact of these two parametrization possibilities.

Adaptive Image Quantization. An advantage of our quantization scheme becomes apparent when the LoA should be adjusted according to image contents, e.g., based on feature semantics, image saliency, or a foreground/background separation. According to Figure 10, we see two possibilities for local-based adjustments: on the one hand, user-defined weights can be injected into Equation 1 to adapt the color extraction, and on the other hand the seed threshold α can be adaptively computed. We experimented with both adjustments by using importance masks—explicitly defined prior to processing—to guide the color extraction to features of interest. Figure 11 shows a result where more contemporary colors are derived from a salient image feature to depict it with more details—using the algorithm of Cheng et al. [68] for saliency estimation—without changing the

Figure 12: Using adaptive thresholds for the placement of color seeds to control the level of abstraction for image features. Top: static thresholds $\alpha = 8$, bottom: adaptive thresholds $\alpha \in [3, 11]$ selected according to an importance mask to increase the LoA for features in the background.

number of color extractions. To this end, the score computation defined in Equation 1 is changed to:

$$\mathcal{S}(R) = \frac{\eta(R)}{\bar{\omega} \cdot \bar{L} \cdot \bar{\Omega}} \cdot |R| \,, \tag{7}$$

where $\bar{\Omega}$ refers to the importance for pixels in region R as defined by a normalized input mask. In addition, Figure 12 shows a result where seed thresholds $\alpha \in [\alpha^+, \alpha^-]$ are linearly interpolated according to an importance mask to increase the level of abstraction in the background regions of an image.

3.3. Paint Texture

Oil painting is a time-consuming process that often comprises multiple layers of paint and drying phases. During finishing, thin protective layers (*varnish*) may be coated onto the paint for protection against dirt and dust, and to even out its final appearance. This yields two characteristics of oil paint textures: (1) reliefs of varying thickness according to the used brushes and applied number of layers with (2) a matt or glossy tint. The first effect may be perceived as subtle shading that is caused by external, off-image illumination, and the second effect as specular highlighting. To simulate both effects, first, a flow field is synthesized by using the local orientation information obtained from the smoothed structure tensor [32], afterwards, paint textures are synthesized by using the flow field for shading.

Flow Field Computation. Local orientation information is derived from an eigenanalysis of the smoothed structure tensor [14], a method that provides stable estimates and can be computed in real-time [32]. Line integral convolution [15] is then performed along the stream lines defined by the minor eigenvector field of the smoothed structure tensor. The obtained flow field, however, may contain singularities and blurred feature boundaries leading

to visual artifacts in the paint textures. To this end, we make use of the following enhancements (Figure 13):

1. *Relaxation*: The quantized image may provide large areas of solid color tones where gradient information are unreliable or undefined. To this end, structure tensors with low gradient magnitudes are replaced by inpainted information via relaxation (Figure 13 middle).
2. *Adaptive Smoothing*: The structure tensor is adaptively smoothed to avoid filtering over feature boundaries and to obtain more accurate results (Figure 13 bottom). Here, the main idea is to use the sign of the flow-based Laplacian of Gaussian (FLoG)—derived from the quantized color image—for thresholding: the Gaussian smoothing with standard deviation σ_s is adapted to exclude pixel values from weight averaging when the difference in the signed FLoG to the origin pixel reaches a given threshold, e.g., when the sign flips while crossing feature boundaries.

For a detailed description on the relaxation and FLoG computation we refer to the work by Kyprianidis and Kang [53].

Paint Texture Synthesis. Procedural monochromatic noise is blurred in gradient flow direction defined by the minor eigenvector field of the adaptively smoothed structure tensor. This results in flow images similar to those produced by line integral convolution [15], but using a flow-based Gaussian filter kernel with standard deviation σ_b to elongate the brush reliefs. The blurred noise is then interpreted as fine-grained height field with normals N and illuminated using a directional light source L.

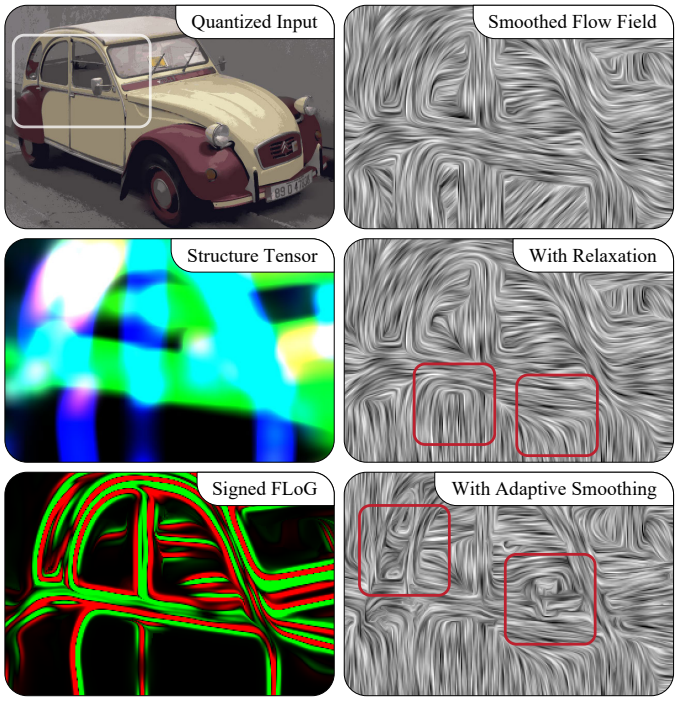

Figure 13: Enhancements for the smoothed structure tensor (here: $\sigma_s = 8$) to derive flow fields. Middle: visualized structure tensor (black regions refer to singularities) and relaxation to avoid singularities, bottom: visualized sign of the FLoG which is thresholded for adaptive smoothing of the structure tensor.

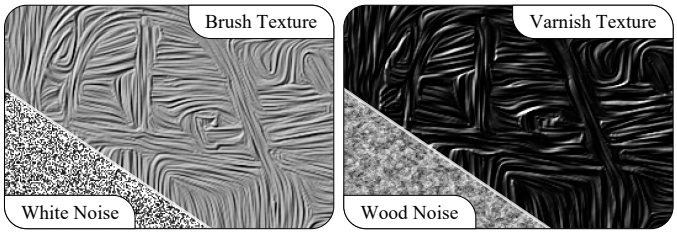

Figure 14: Paint textures computed for the flow field in Figure 13 using parameters $\sigma_b = 8.0$, $k_{scale} = 10.0$, $k_{specular} = 3.0$, $k_{shininess} = 8.0$.

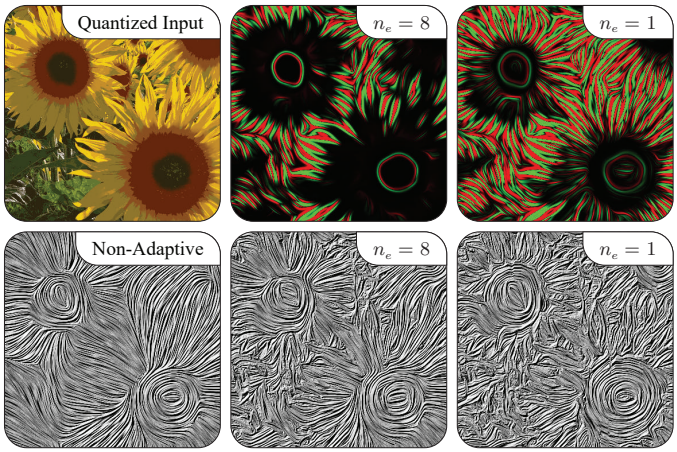

Figure 15: Varying the number of iterations for orientation-aligned bilteral filtering (n_e) of the quantized image, used for FLoG filtering. Top: visualized sign of the FLoG filter, bottom: brush texture with and without adaptive smoothing.

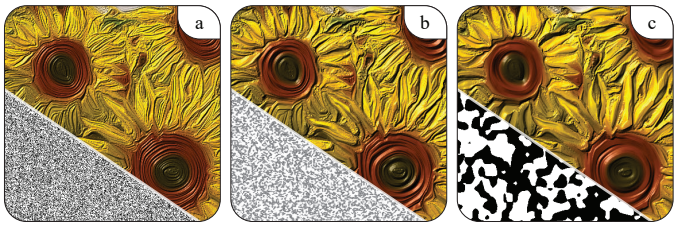

Figure 16: Different noise frequencies and amplitudes to adjust the virtual brush. Constant parameters: $\sigma_b = 20.0$, $k_{scale} = 20.0$, $k_{specular} = 0.5$, $k_{shininess} = 20.0$.

Here, principles of Phong shading [16] are used to render a brush texture $\mathcal{T}_B$ and a varnish texture $\mathcal{T}_V$ with pixels p:

$$\begin{aligned} \mathcal{T}_B(p) &= 0.5 + N(p) \cdot L, \\ \mathcal{T}_V(p) &= k_{specular} \cdot (N(p) \cdot L)^{k_{shininess}}. \end{aligned} \tag{8}$$

At this, $k_{specular}$ and $k_{shininess}$ are used to adjust the varnish texture. An additional factor k_{scale} scales the height field to control the relief strength (Figure 14). The standard deviation σ_b used for noise filtering is a key parameter to control the LoA of the paint textures, where we observed that our enhancement for adaptive smoothing of the structure tensor is crucial to preserve salient feature curves. Figure 15 shows how to further adjust the LoA when pre-processing the quantized image—used as input for FLoG filtering—by an orientation-aligned bilateral filter [32] with a varying number of iterations n_e. This approach has also proven to be effective in cartoon-like filtering [31]

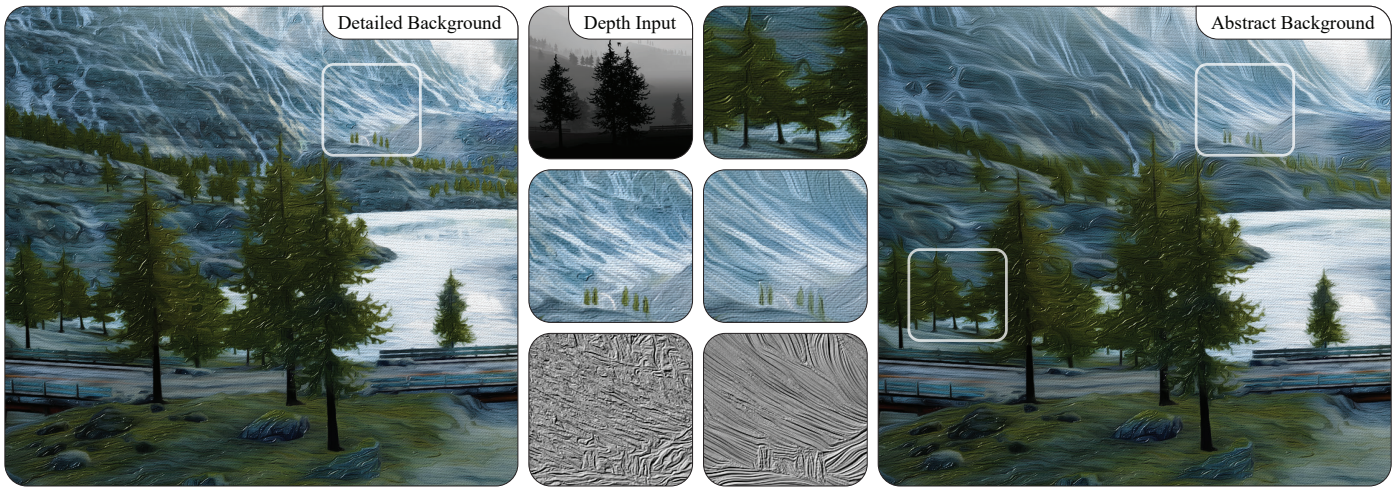

Figure 17: Depth-dependent synthesis (color and depth input rendered with Unreal Engine 4) of paint textures to vary the LoA in the background region of a rendered image. Foreground: $\mathcal{K}_Q$ = (NA, NA, 4.0, 20.0) and $\mathcal{K}_\mathcal{T}$ = (0, 20.0, 10.0, 0.8, 20.0), background (*left*): $\mathcal{K}_Q$ = (NA, NA, 1.0, 8.0) and $\mathcal{K}_\mathcal{T}$ = (0, 8.0, 6.0, 0.8, 20.0), background (*right*): $\mathcal{K}_Q$ = (NA, NA, 10.0, 20.0) and $\mathcal{K}_\mathcal{T}$ = (2, 20.0, 10.0, 0.8, 20.0). The visualized brush textures do not include additional filtering by saturation.

Table 1: Overview of parameters with value ranges used within this paper to adjust the color image quantization and paint texture synthesis. Left-end values typically refer to a low LoA, whereas right-end value refer to a high LoA.

	Parameter	Value Range	Effect
Quantization	n	36 – 8	Size of color palette
	α	11.0 – 4.0	Color seed threshold
	σ_s	2.0 – 20.0	Structure tensor smoothing (std. dev.)
	σ_q	0.0 – 20.0	Quantized image smoothing (std. dev.)
Paint Textures	n_e	0 – 10	Iterations of bilateral filtering for FLoG
	σ_b	2.0 – 20.0	Noise smoothing (std. dev.)
	k_{scale}	20.0 – 0.0	Relief strength
	$k_{specular}$	5.0 – 0.3	Specularity for varnish
	$k_{shininess}$	30.0 – 10.0	Shininess for varnish

to adjust contour lines. We also experimented with different noise implementations and observed that high frequency noise simulates brush characteristics quite naturally (Figure 14), but may also be based on lower frequencies and amplitudes to adjust the brush size (Figure 16).

Image Composition. Finally, the brush texture is multiplied with the smoothed color image, and the intermediate result is blended with the varnish texture using *linear dodge* as blend mode. Optionally, contour lines are enhanced by using a flow-based DoG filter [33] and a canvas texture is blended with the output to further enhance the sensation of depth. For the latter, the paint textures may also be filtered in image regions of low saturation to imitate layering at certain feature boundaries (e.g., Figure 2).

Adaptive Paint Texture Synthesis. The paint texture synthesis is local and fast (Section 5), and thus may also be adaptively computed according to user interaction or image contents. Table 1 summarizes the parameters used to control the LoA on a per-pixel basis, where $\mathcal{K}_Q = (n, \alpha, \sigma_s, \sigma_q)$ adjusts the image quantization and abstraction, and $\mathcal{K}_\mathcal{T} = (n_e, \sigma_b, k_{scale}, k_{specular}, k_{shininess})$ adjusts the brush and varnish textures. Again, we experimented with image masks to parametrize the paint textures. Figure 17 demonstrates a depth-dependent linear interpolation between two parameter sets to depict fewer or more details in the background regions of an image. In addition, Figure 18 shows an approach where image saliency is used to direct the granularity of the brush and varnish textures. Additional effects may be easy to implement, e.g., based on lightfield data to produce stylized depth of field effects [69] or feature semantics with qualitative parameter sets for content-aware filtering.

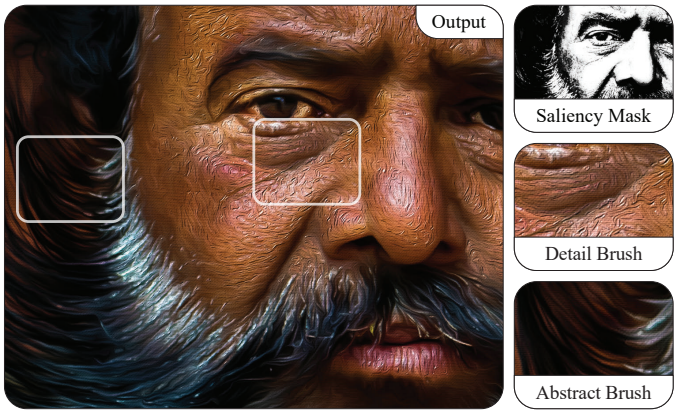

Figure 18: Saliency-based filtering output of a portrait with detailed paint textures in image regions with facial skin. High saliency: $\mathcal{K}_Q$ = (NA, NA, 4.0, 6.0) and $\mathcal{K}_\mathcal{T}$ = (0, 6.0, 7.0, 0.8, 20.0), low saliency: $\mathcal{K}_Q$ = (NA, NA, 4.0, 16.0) and $\mathcal{K}_\mathcal{T}$ = (8, 16.0, 2.0, 0.3, 20.0). For the computation of the saliency mask, the algorithm of Cheng et al. [68] was used.

4. Interactive Painting

When tweaking the output of an image filter, users typically strive for a global parametrization-trade-off that corresponds to multiple visual requirements in different image regions. Recently, consumer photo-editing products started to extend the concept of non-destructive per-pixel parametrizations from alpha masking to a small set of image computations (adjustments)

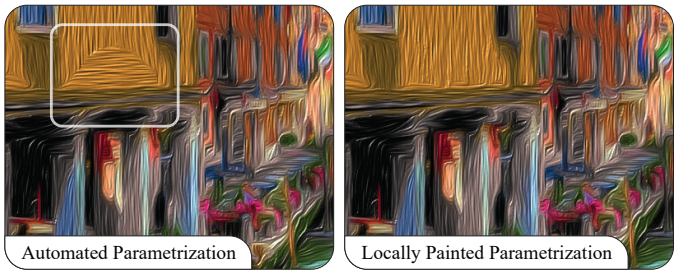

Figure 19: Failure case of a filtering result and its manual correction using our painting interface for per-pixel parametrization.

by means of parameter masks. We extended this approach by exposing filter parameters as well as intermediate filtering results as parameter maps that can be manipulated via painting metaphors. This enables (1) artistic control over the stylization process, and (2) the modification of intermediate filter outputs of inadequate local parametrizations (Figure 19).

4.1. Local Parameter Painting

Our method extends the concept of a specialized, locally computed parametrization [61] to a generalized configuration by brush-painting within parameter spaces of local image filters. At this, parameters are encoded as *parameter maps* and adjusted via *virtual brush models* according to well-defined action sequences. Thereby, we denote the following system:

- A *parameter map* is a typed map that substitutes either a uniform filter parameter or an intermediate computational result. These maps are usually aligned with the input image, but might also cover sub-regions and have different image resolutions.
- An *action* defines locally typed computations on parameter maps, e.g., *replace*, *add*, *scale*, or *blur*. It is filter-independent and can be assigned to equally-typed parameter maps. The type usually depends on the number of a parameter's coefficients.
- A *brush shape* specifies rules and parameters for the dynamic creation of two-dimensional weight masks. Thereby, atomic shapes—either by functional definition or shapes from vector graphics—are (randomly) placed and encoded as distance maps while satisfying specific constraints (e.g., softness, jittering, scattering).
- A *brush stroke* is derived from a set of sequential user inputs, e.g., attributed paths with interpolated position, orientation, and pressure values. Here, a temporal input mask is created by computing dynamic brush shapes along the path, enabling per-pixel weighting of actions.
- A *brush* maps a sequence of actions to parameter maps. While drawing, these actions are applied and weighted by the temporal input mask (brush stroke).

This system provides a generic interface and is used to locally adjust filter parameters and intermediate filter results through

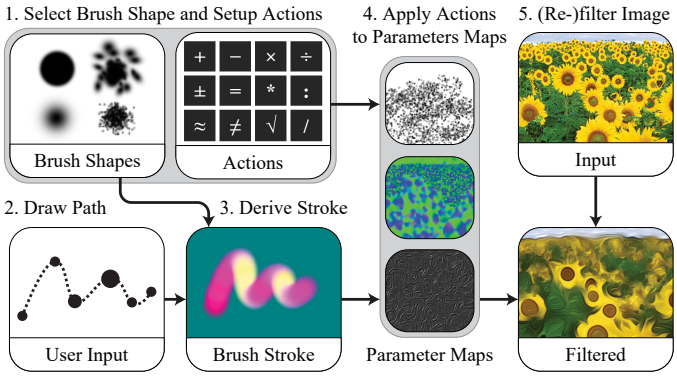

Figure 20: Schematic overview of our per-pixel parametrization interface: brush strokes are derived from drawn paths and applied to selected parameter maps. The maps' modified local parameters are then used to (re-)filter the image.

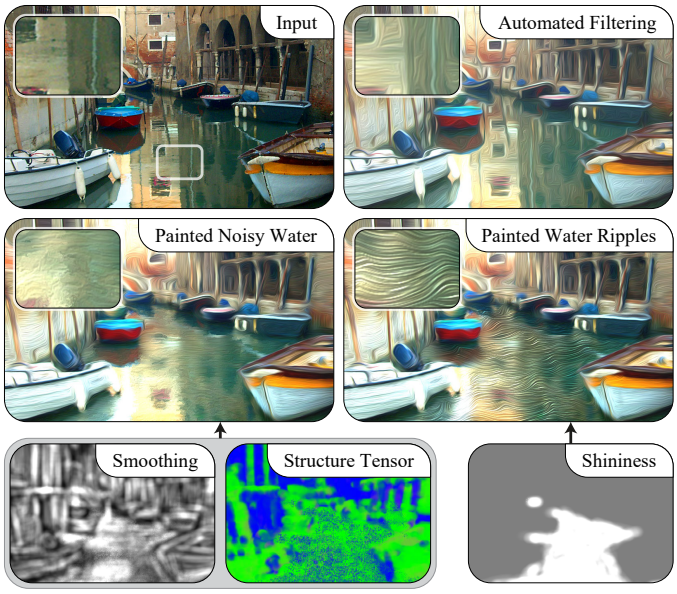

Figure 21: Example of manual corrections made for parameter layers to adjust the paint textures: smoothing of the flow field, flow direction, and shininess.

painting (schematized in Figure 20). The brush implementation is hardware-accelerated and is used, in particular, to locally adjust the parameters defined by $\mathcal{K}_Q$ and $\mathcal{K}_\mathcal{T}$. Examples are given in Figure 21 and the accompanying video.

4.2. Brushes

The shape and actions of a brush can be parametrized by constant values or dynamically mapped to user inputs, e.g., the pressure and orientation of a digital pen, or gestures for touch-enabled devices. Technically, our implementation enables users to create and customize brushes at runtime, but which demands a detailed understanding of the underlying filtering stages. To this end, we provide a number of predefined brushes:

- A *relief brush* increases or decreases the relief strength by applying a multiply action to the height scale k_{scale}.
- A *varnish brush* allows to adjust the specularity and shininess of the varnish texture. It applies two multiply actions

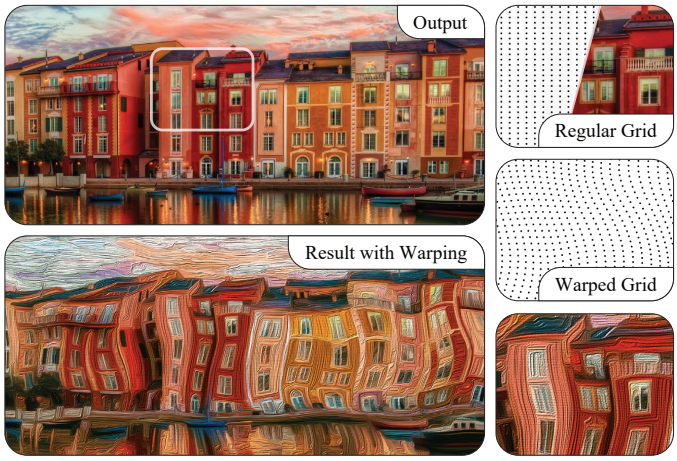

Figure 22: Image warping using grid-based resampling, parameterized by a virtual brush tool. The warping is performed prior to paint texture synthesis.

Figure 23: Touchscreen and user interface for the proposed painting system.

to the parameter maps of $k_{shininess}$ and $k_{specular}$.

- Two *LoA brushes*: one to adjust the structure tensor smoothing σ_s and another to perform bilateral filtering and apply unsharp masking effects.
- A *flow brush* to adjust the tangential information of the structure tensor, which is especially helpful to exaggerate or fix inadequate stroke directions.
- A *colorization brush* that allows to fade between the color output and a grayscale version.
- An *eraser brush* that reverts to initial painting states for all parameter maps or those of specific brushes.

Additional brushes, e.g., for painting bristle structures by scaling the noise frequency or adjusting additional filter kernels for smoothing (σ_q and σ_b) are possible but not yet implemented. To simplify the mapping of actions to parameter maps, a single texture for every parameter or intermediate result is used. For large images, the application of actions is restricted to sub-regions of the parameter maps—according to the bounds of a brush stroke—to maintain a responsive painting system.

4.3. Image Warping

Artists often exaggerate figures and shapes to amplify the mood of their paintings. Prominent examples are art works from the Expressionism era—such as Edvard Munch's *The Scream* (1893–1910). Image filtering, however, is generally less suited for intentional shape abstraction. Here, one approach is to complement our technique by an interactive brush tool to perform local image warping. Starting with a virtual, regular grid, the user is able to shift local grid points by brush-based painting to create effects of local compression and distortion (see Figure 22 and the accompanying video). A similar approach was used before with facial constraints to create caricatures from photographs [70]. The adapted grid is then used for texture parametrization and to resample the quantized image by bilinear interpolation. Finally, the warped image serves as input for the paint texture synthesis—which may also be performed during warping for immediate visual feedback. Alternatively, image segmentation and mass-spring systems may be used to create outputs with more deliberate shape abstractions as demonstrated by Li and Mould [71].

5. Results

We have implemented the dominant color extraction using C++, the colorization and filtering stages on the GPU with CUDA, and the painting interface with Qt. All images were processed on an Intel® Xeon™ 4× 3.06 GHz and NVidia® GTX 760 GPU with 4 GByte VRAM. The painting system was tested with a 85" multitouch monitor with Ultra-HD resolution (Figure 23). A 800×600 pixel image is processed in 50 seconds for a palette with 25 colors. Here, the color extraction is currently the limiting stage, followed by the colorization, and the paint texture synthesis that performs in real-time for images with HD resolution. As demonstrated in Figure 7, the color extraction also provides stable estimates when processing downscaled images—up to the second pyramid level to speed up the processing by a factor of two. To enable interactive performance during painting, the stages after the image quantization are optimized to process only those image regions that require a recomputation. At this, the processing performs three steps per render cycle: 1) local regions defined by the virtual brush are buffered and extracted from the quantized image and parameter layers (stored in main memory) and are transferred to GPU memory, 2) the sub-images are processed according to the brush mode and the results are blit to the framebuffer for immediate visual feedback, 3) the filtered sub-images are transferred back to main memory. Using pitch linear memory on the described test system, this procedure enables to interactively adjust images up to 100 MP.

The proposed color quantization gives control to adjust the level of abstraction. A comparison to a non-quantized version of an image is shown in Figure 25, and demonstrates how the quantization is able to filter detail information and produce large areas of solid color tones. In particular, the latter effect yields varying scales of the paint texture, i.e., to simulate thick brushes. Figure 24 and Figure 26 show comparisons of the proposed technique to previous works. In contrast to stroke-based rendering,

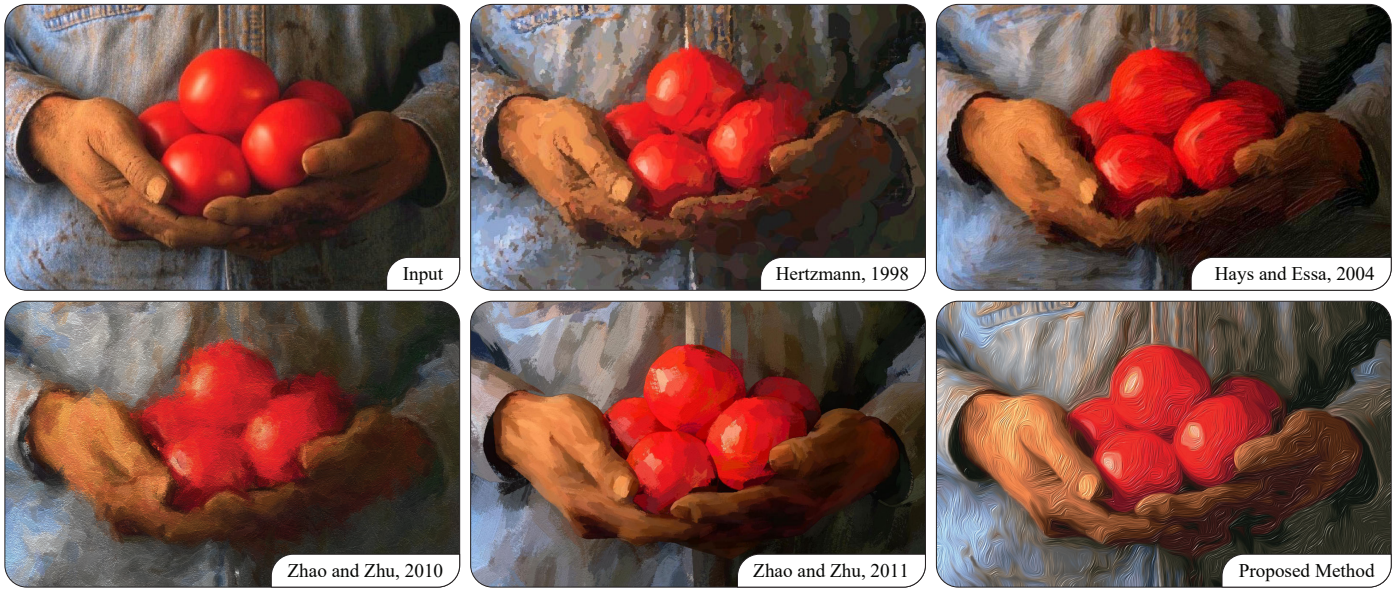

Figure 24: Comparison of the proposed method with stroke-based rendering techniques. LRTB: input, from Hertzmann [8], from Hays and Essa [9], from Zhao and Zhu [19], from Zhao and Zhu [72], and proposed method with $\mathcal{K}_Q = (34, 7.0, 6.0, 10.0)$ and $\mathcal{K}_\mathcal{T} = (4, 10.0, 3.0, 1.0, 10.0)$. Each method produces visually distinct outputs of varying expressiveness, texture, and feature alignment. For instance, the method of Hertzmann [8] aligns brush strokes in a color-matching scheme, but tends to overdraw features in regions of low contrast. Hays and Essa [9] use real brush stroke textures that are feature-aligned with globally interpolated orientations, yet their approach lacks complementary colors in neighboring strokes. This effect is simulated in the method of Zhao and Zhu [72] (2011) to explicitly emphasize feature contrasts. Zhao and Zhu [19] (2010) simulate the perceptual ambiguities known from abstract art, where shape-simplifying abstraction plays a major role. Finally, the proposed method produces soft color blendings with no visible borders between brush strokes, yet without the capability for explicit shape abstraction.

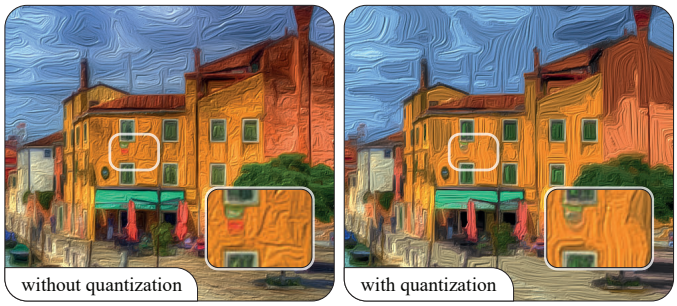

Figure 25: Two versions of a stylized image: (left) without prior quantization, (right) with prior quantization. The quantized version produces a more abstract look with respect to the color range and scale of the paint texture. Parameters used: $\mathcal{K}_Q = (40, 8.0, 3.0, 14.0)$ and $\mathcal{K}_\mathcal{T} = (0, 14.0, 5.0, 1.0, 16.0)$.

our approach produces outputs with more soft color blendings (Figure 24) but is also able to simulate reliefs of varying thickness and strong abstraction (e.g., the background and dress in Figure 26). In addition, the local parametrization capabilities are mandatory to provide artistic control over the depiction of single image features (e.g., the face in Figure 26), i.e., to provide an adaptive LoA that is similar to hybrid stylization techniques (e.g., the work of Zeng et al. [10], Figure 26). More images processed by our technique are shown in Figure 28, where the LoA is adapted to the image contents to have more distinct colors in colorful images (e.g., *Venice*) and wide filter kernels for Gaussian smoothing to obtain soft color blendings (e.g., the *landscape*). Moreover, the synthesized noise plays a major role for abstraction, e.g., wood noise can be used to simulate thick paint, and white noise is effective to simulate detail brushes.

5.1. Limitations

An inherent limitation of our technique is that it does not reach the qualities of shape abstraction as provided by top-down stroke-based rendering techniques (e.g., the method of Zhao and Zhu [19]). The proposed painting system with complementary warping gives users some local control to adjust the LoA. Further, image features that should be represented with soft color blendings may be filtered with (unwanted) hard transitions, depending on the seed color placement. Eventually, this requires manual effort to locally adjust the thresholds for the color quantization. Finally, the performance of the colorization currently does not enable interactive color refinements. One approach to alleviate this issue is to visualize intermediate results of the iterative solver for visual feedback, or to accelerate the colorization using fast intrinsic distance computations [73].

5.2. Future Work

We see multiple directions for future work. A major strength of our technique is that the stylization uses global color palettes that may be easily refined interactively. Here, we plan to implement the transfer of color moods to adapt a derived palette to a target palette or image. Second, the color extraction may be improved to support feature-aligned (non-rectangular) regions for a more robust extraction, e.g., via a generalized region growing. Third, the adaptive filtering approaches could be extended to support image semantics and stylize certain features with different parameter sets, e.g., by image parsing [10]. Fourth, the extension of our technique to video is of particular interest. Here, the colorization could be extended by a temporal constraint as proposed by Levin et al. [13], together with an optical flow to

Figure 26: Comparison of the proposed method with the image filtering technique of Winnemöller et al. [31] and the stroke-based rendering technique of Zeng et al. [10]. Parameters: $\mathcal{K}_Q = (38, 7.0, 16.0, 16.0)$, base: $\mathcal{K}_\mathcal{T} = (0, 16.0, 5.0, 1.0, 16.0)$ and skin: $\mathcal{K}_\mathcal{T} = (0, 16.0, 1.0, 1.0, 10.0)$ with no color quantization in facial regions.

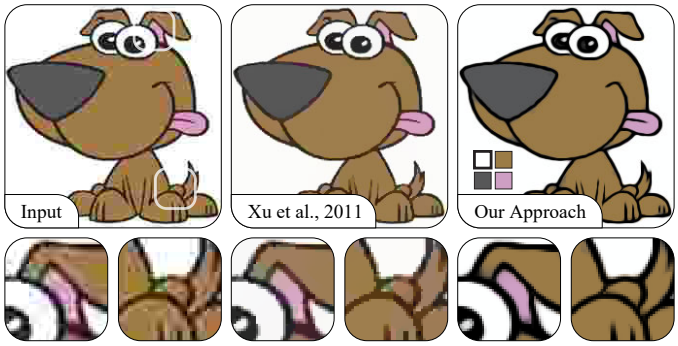

Figure 27: Comparison of image smoothing via L0 gradient minimization [38] with our quantization technique, combined with the output of a DoG filter, for JPEG artifact removal.

stabilize the paint texture synthesis. Finally, we believe that the palette-based quantization and colorization are quite generic in their application and could be applied to further problems. For instance, we experimented using our methods for JPEG artifact removal of clip-arts (Figure 27), where our approach produces accurate results and may also be used to easily redefine single colors.

6. Conclusions

We have presented an approach for transforming images into filtered variants with an oil paint look. The proposed color extraction and colorization methods enable to quantize color images according to their dominant color palette. Results show that our quantization scheme is able to represent selected image features accurately and provide homogeneous outputs in the color domain. The flow-based image abstraction and proposed paint texture synthesis perform in real-time to enable interactive refinements, and facilitate per-pixel parametrizations to direct the level of abstraction to user-defined or salient image regions. Several results demonstrate the manifold application of our approach to different genres of photography and to simulate paint with soft to moderate color blendings.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and Holger Winnemöller for his support on the *flowpaint* research project. This work was partly funded by the Federal Ministry of Education and Research (BMBF), Germany, within the InnoProfile Transfer research group "4D*n*D-Vis" (`www.4dndvis.de`), and was partly supported by the ERC through grant ERC-2010-StG 259550 (XSHAPE).

References

[1] Kyprianidis, J.E., Collomosse, J., Wang, T., Isenberg, T.. State of the 'Art': A Taxonomy of Artistic Stylization Techniques for Images and Video. IEEE Trans Vis Comput Graphics 2013;19(5):866–885. doi:10.1109/TVCG.2012.160.

[2] Scott, M.. Oil Painter's Bible. Chartwell Books; 2005.

[3] Haeberli, P.. Paint by Numbers: Abstract Image Representations. SIGGRAPH Comput Graph 1990;24(4):207–214. doi:10.1145/97880.97902.

[4] Hertzmann, A.. A survey of stroke-based rendering. IEEE Computer Graphics and Applications 2003;(4):70–81.

[5] Earls, I.. Renaissance art: a topical dictionary. ABC-CLIO; 1987.

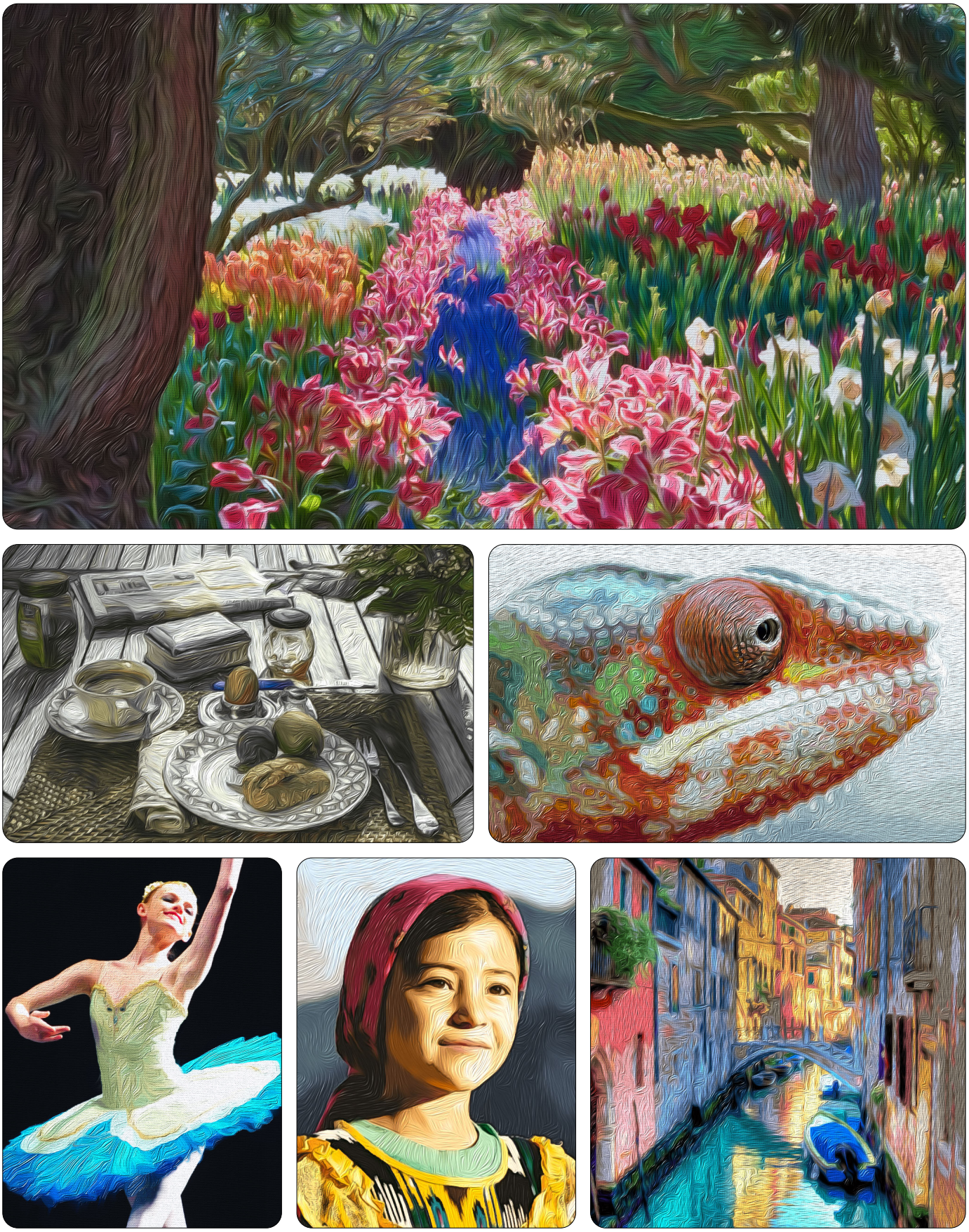

Figure 28: Image stylization results produced with the proposed oil paint filtering technique.

[6] Weickert, J.. Anisotropic diffusion in image processing; vol. 1. Teubner Stuttgart; 1998.

[7] Kang, H., Lee, S.. Shape-simplifying Image Abstraction. In: Computer Graphics Forum; vol. 27. 2008, p. 1773–1780.

[8] Hertzmann, A.. Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In: Proc. ACM SIGGRAPH. ACM; 1998, p. 453–460. doi:10.1145/280814.280951.

[9] Hays, J., Essa, I.. Image and Video Based Painterly Animation. In: Proc. NPAR. ACM; 2004, p. 113–120. doi:10.1145/987657.987676.

[10] Zeng, K., Zhao, M., Xiong, C., Zhu, S.C.. From Image Parsing to Painterly Rendering. ACM Trans Graph 2009;29(1):2:1–2:11. doi:10.1145/1640443.1640445.

[11] Baxter, W., Wendt, J., Lin, M.C.. IMPaSTo: A Realistic, Interactive Model for Paint. In: Proc. NPAR. ACM; 2004, p. 45–148. doi:10.1145/987657.987665.

[12] Lu, J., Barnes, C., DiVerdi, S., Finkelstein, A.. RealBrush: Painting with Examples of Physical Media. ACM Trans Graph 2013;32(4):117:1–117:12. doi:10.1145/2461912.2461998.

[13] Levin, A., Lischinski, D., Weiss, Y.. Colorization Using Optimization. ACM Trans Graph 2004;23(3):689–694. doi:10.1145/1015706.1015780.

[14] Brox, T., Boomgaard, R., Lauze, F., Weijer, J., Weickert, J., Mrázek, P., et al. Adaptive Structure Tensors and their Applications. In: Visualization and Processing of Tensor Fields. Springer Berlin Heidelberg; 2006, p. 17–47. doi:10.1007/3-540-31272-2_2.

[15] Cabral, B., Leedom, L.C.. Imaging Vector Fields Using Line Integral Convolution. In: Proc. ACM SIGGRAPH. ACM; 1993, p. 263–270. doi:10.1145/166117.166151.

[16] Phong, B.T.. Illumination for Computer Generated Pictures. Commun ACM 1975;18(6):311–317. doi:10.1145/360825.360839.

[17] Semmo, A., Limberger, D., Kyprianidis, J.E., Döllner, J.. Image Stylization by Oil Paint Filtering Using Color Palettes. In: Proc. CAe. The Eurographics Association; 2015, p. 149–158.

[18] Gooch, B., Coombe, G., Shirley, P.. Artistic Vision: Painterly Rendering Using Computer Vision Techniques. In: Proc. NPAR. ACM; 2002, p. 83–ff. doi:10.1145/508530.508545.

[19] Zhao, M., Zhu, S.C.. Sisley the Abstract Painter. In: Proc. NPAR. ACM; 2010, p. 99–107. doi:10.1145/1809939.1809951.

[20] Lu, J., Sander, P.V., Finkelstein, A.. Interactive Painterly Stylization of Images, Videos and 3D Animations. In: Proc. ACM I3D. ACM; 2010, p. 127–134. doi:10.1145/1730804.1730825.

[21] Hegde, S., Gatzidis, C., Tian, F.. Painterly rendering techniques: a state-of-the-art review of current approaches. Comp Anim Virtual Worlds 2013;24(1):43–64. doi:10.1002/cav.1435.

[22] Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.. Image Analogies. In: Proc. ACM SIGGRAPH. ACM; 2001, p. 327–340. doi:10.1145/383259.383295.

[23] Zhao, M., Zhu, S.C.. Portrait Painting Using Active Templates. In: Proc. NPAR. ACM; 2011, p. 117–124. doi:10.1145/2024676.2024696.

[24] Wang, T., Collomosse, J., Hunter, A., Greig, D.. Learnable Stroke Models for Example-based Portrait Painting. In: Proc. British Machine Vision Conference. BMVA; 2013, p. 36.1–36.11.

[25] Gatys, L.A., Ecker, A.S., Bethge, M.. A Neural Algorithm of Artistic Style. CoRR 2015;URL: http://arxiv.org/abs/1508.06576.

[26] DeCarlo, D., Santella, A.. Stylization and Abstraction of Photographs. ACM Trans Graph 2002;21(3):769–776. doi:10.1145/566654.566650.

[27] Wen, F., Luan, Q., Liang, L., Xu, Y.Q., Shum, H.Y.. Color Sketch Generation. In: Proc. NPAR. ACM; 2006, p. 47–54. doi:10.1145/1124728.1124737.

[28] Mould, D.. A Stained Glass Image Filter. In: Proc. EGRW. 2003, p. 20–25. URL: http://dl.acm.org/citation.cfm?id=882404.882407.

[29] O'Donovan, P., Mould, D.. Felt-based Rendering. In: Proc. NPAR. ACM; 2006, p. 55–62. doi:10.1145/1124728.1124738.

[30] Tomasi, C., Manduchi, R.. Bilateral Filtering for Gray and Color Images. In: Proc. ICCV. IEEE; 1998, p. 839–846. doi:10.1109/ICCV.1998.710815.

[31] Winnemöller, H., Olsen, S.C., Gooch, B.. Real-Time Video Abstraction. ACM Trans Graph 2006;25(3):1221–1226. doi:10.1145/1141911.1142018.

[32] Kyprianidis, J.E., Döllner, J.. Image Abstraction by Structure Adaptive Filtering. In: Proc. EG UK TPCG. The Eurographics Association; 2008, p. 51–58. doi:10.2312/LocalChapterEvents/TPCG/TPCG08/051-058.

[33] Kang, H., Lee, S., Chui, C.K.. Flow-Based Image Abstraction. IEEE Trans Vis Comput Graphics 2009;15(1):62–76. doi:10.1109/TVCG.2008.81.

[34] Yoon, J.C., Lee, I.K., Kang, H.. Video Painting Based on a Stabilized Time-Varying Flow Field. IEEE Trans Vis Comput Graphics 2012;18:58–67. doi:10.1109/TVCG.2011.47.

[35] Bousseau, A., Kaplan, M., Thollot, J., Sillion, F.X.. Interactive Watercolor Rendering with Temporal Coherence and Abstraction. In: Proc. NPAR. ACM; 2006, p. 141–149. doi:10.1145/1124728.1124751.

[36] Farbman, Z., Fattal, R., Lischinski, D., Szeliski, R.. Edge-Preserving Decompositions for Multi-Scale Tone and Detail Manipulation. ACM Trans Graph 2008;27(3):67:1–67:10. doi:10.1145/1360612.1360666.

[37] Subr, K., Soler, C., Durand, F.. Edge-preserving Multiscale Image Decomposition based on Local Extrema. ACM Trans Graph 2009;28(5):147:1–147:9. doi:10.1145/1661412.1618493.

[38] Xu, L., Lu, C., Xu, Y., Jia, J.. Image Smoothing via L_0 Gradient Minimization. ACM Trans Graph 2011;30(6):174:1–174:12. doi:10.1145/2070781.2024208.

[39] Heckbert, P.. Color Image Quantization for Frame Buffer Display. SIGGRAPH Comput Graph 1982;16(3):297–307. doi:10.1145/965145.801294.

[40] Gervautz, M., Purgathofer, W.. A Simple Method for Color Quantization: Octree Quantization. In: New Trends in Computer Graphics. Springer Berlin Heidelberg; 1988, p. 219–231. doi:10.1007/978-3-642-83492-9_20.

[41] Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.. An efficient k-means clustering algorithm: Analysis and implementation. IEEE Trans Pattern Anal Mach Intell 2002;24(7):881–892. doi:10.1109/TPAMI.2002.1017616.

[42] Chen, J., Pappas, T.N., Mojsilovic, A., Rogowitz, B.. Adaptive Perceptual Color-Texture Image Segmentation. IEEE Trans Image Processing 2005;14(10):1524–1536. doi:10.1109/TIP.2005.852204.

[43] Yue, X., Miao, D., Cao, L., Wu, Q., Chen, Y.. An efficient color quantization based on generic roughness measure. Pattern Recognition 2014;47(4):1777–1789. doi:10.1016/j.patcog.2013.11.017.

[44] Yu, J., Lu, C., Sato, Y.. Sparsity-based Color Quantization with Preserved Image Details. In: SIGGRAPH Asia 2014 Posters. ACM; 2014, p. 32:1–32:1. doi:10.1145/2668975.2668999.

[45] Kim, T.h., Ahn, J., Choi, M.G.. Image Dequantization: Restoration of Quantized Colors. Comput Graph Forum 2007;26(3):619–626. doi:10.1111/j.1467-8659.2007.01085.x.

[46] Santella, A., DeCarlo, D.. Visual Interest and NPR: an Evaluation and Manifesto. In: Proc. NPAR. ACM; 2004, p. 71–150. doi:10.1145/987657.987669.

[47] Cole, F., DeCarlo, D., Finkelstein, A., Kin, K., Morley, K., Santella, A.. Directing Gaze in 3D Models with Stylized Focus. In: Proc. EGSR. The Eurographics Association; 2006, p. 377–387. doi:10.2312/EGWR/EGSR06/377-387.

[48] Cong, L., Tong, R., Dong, J.. Selective Image Abstraction. Vis Comput 2011;27(3):187–198. doi:10.1007/s00371-010-0522-2.

[49] Collomosse, J., Hall, P.. Genetic Paint: A Search for Salient Paintings. In: Applications of Evolutionary Computing; vol. 3449. Springer Berlin Heidelberg; 2005, p. 437–447. doi:10.1007/978-3-540-32003-6_44.

[50] Orzan, A., Bousseau, A., Barla, P., Thollot, J.. Structure-preserving Manipulation of Photographs. In: Proc. NPAR. ACM; 2007, p. 103–110. doi:10.1145/1274871.1274888.

[51] Rosin, P.L., Lai, Y.K.. Non-photorealistic Rendering with Spot Colour. In: Proc. CAe. ACM; 2013, p. 67–75. doi:10.1145/2487276.2487280.

[52] Chu, N., Baxter, W., Wei, L.Y., Govindaraju, N.. Detail-preserving Paint Modeling for 3D Brushes. In: Proc. NPAR. ACM; 2010, p. 27–34. doi:10.1145/1809939.1809943.

[53] Kyprianidis, J.E., Kang, H.. Image and Video Abstraction by Coherence-Enhancing Filtering. Comput Graph Forum 2011;30(2):593–602. doi:10.1111/j.1467-8659.2011.01882.x.

[54] Hertzmann, A.. Fast Paint Texture. In: Proc. NPAR. ACM; 2002, p. 91–96. doi:10.1145/508530.508546.

[55] Zhang, E., Hays, J., Turk, G.. Interactive Tensor Field Design and Visualization on Surfaces. IEEE Trans Vis Comput Graphics 2007;13(1):94–107. doi:10.1109/TVCG.2007.16.

[56] Kagaya, M., Brendel, W., Deng, Q., Kesterson, T., Todorovic, S.,

Neill, P., et al. Video Painting with Space-Time-Varying Style Parameters. IEEE Trans Vis Comput Graphics 2011;17(1):74–87. doi:10.1109/TVCG.2010.25.

[57] Olsen, S.C., Maxwell, B.A., Gooch, B.. Interactive Vector Fields for Painterly Rendering. In: Proc. Graphics Interface. Canadian Human-Computer Communications Society; 2005, p. 241–247.

[58] Hanrahan, P., Haeberli, P.. Direct WYSIWYG Painting and Texturing on 3D Shapes. SIGGRAPH Comput Graph 1990;24(4):215–223. doi:10.1145/97880.97903.

[59] Schwarz, M., Isenberg, T., Mason, K., Carpendale, S.. Modeling with Rendering Primitives: An Interactive Non-photorealistic Canvas. In: Proc. NPAR. ACM; 2007, p. 15–22. doi:10.1145/1274871.1274874.

[60] Anjyo, K.i., Wemler, S., Baxter, W.. Tweakable Light and Shade for Cartoon Animation. In: Proc. NPAR. ACM; 2006, p. 133–139. doi:10.1145/1124728.1124750.

[61] Todo, H., Anjyo, K.i., Baxter, W., Igarashi, T.. Locally Controllable Stylized Shading. ACM Trans Graph 2007;26(3):17:1–17:7. doi:10.1145/1276377.1276399.

[62] McCann, J., Pollard, N.S.. Real-time Gradient-domain Painting. ACM Trans Graph 2008;27(3):93:1–93:7. doi:10.1145/1360612.1360692.

[63] Baxter, W.V., Lin, M.C.. A Versatile Interactive 3D Brush Model. In: Proc. Pacific Graphics. IEEE; 2004, p. 319–328.

[64] Ritter, L., Li, W., Curless, B., Agrawala, M., Salesin, D.. Painting With Texture. In: Proc. EGSR. The Eurographics Association; 2006, p. 371–376.

[65] DiVerdi, S.. A brush stroke synthesis toolbox. In: Image and Video-Based Artistic Stylisation; vol. 42. Springer London; 2013, p. 23–44. doi:10.1007/978-1-4471-4519-6_2.

[66] Mahy, M., Eycken, L., Oosterlinck, A.. Evaluation of Uniform Color Spaces Developed after the Adoption of CIELAB and CIELUV. Color Research & Application 1994;19(2):105–121. doi:10.1111/j.1520-6378.1994.tb00070.x.

[67] Saad, Y., Schultz, M.H.. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci and Stat Comput 1986;7(3):856–869. doi:10.1137/0907058.

[68] Cheng, M.M., Warrell, J., Lin, W.Y., Zheng, S., Vineet, V., Crook, N.. Efficient Salient Region Detection with Soft Image Abstraction. In: Proc. ICCV. IEEE; 2013, p. 1529–1536. doi:10.1109/ICCV.2013.193.

[69] Bousseau, A.. Non-Linear Aperture for Stylized Depth of Field. In: ACM SIGGRAPH Talks. ACM; 2009, p. 57:1–57:1. doi:10.1145/1597990.1598047.

[70] Gooch, B., Reinhard, E., Gooch, A.. Human Facial Illustrations: Creation and Psychophysical Evaluation. ACM Trans Graph 2004;23:27–44.

[71] Li, J., Mould, D.. Image Warping for a Painterly Effect. In: Proc. CAe. The Eurographics Association; 2015, p. 131–140.

[72] Zhao, M., Zhu, S.C.. Customizing Painterly Rendering Styles Using Stroke Processes. In: Proc. NPAR. ACM; 2011, p. 137–146. doi:10.1145/2024676.2024698.

[73] Yatziv, L., Sapiro, G.. Fast image and video colorization using chrominance blending. IEEE Trans Image Processing 2006;15(5):1120–1129. doi:10.1109/TIP.2005.864231.

Original photographs used in Figure 6a/d/e, Figure 7a, Figure 12, Figure 13, and Figure 28 (car) courtesy Phillip Greenspun. Photographs from flickr.com kindly provided under Creative Commons license by Anita Priks (Figure 2), Vincent van der Pas (Figure 3), Gulsen Ozcan (Figure 8), Akaporn Bhothisuwan (Figure 9), Valerija Fetsch (Figure 11), matthiashn (Figure 16), Rajarshi Mitra (Figure 18), Mark Pouley (Figure 28 / landscape), Jelto Buurman (Figure 28 / still life), Harclade (Figure 28 / ballerina), Christophe Chenevier (Figure 28 / girl), and Florence Ivy (Figure 28 / chameleon).