Painting Per-Pixel Parametrization for Interactive Image Filtering

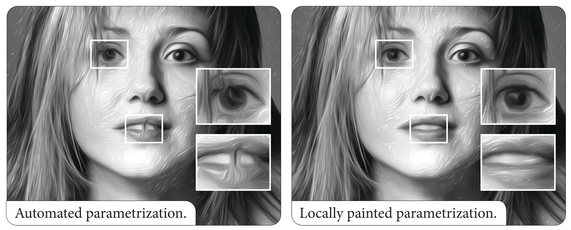
Daniel Limberger Jürgen Döllner
Hasso Plattner Institute, University of Potsdam, Germany

1 Abstract

We present a photo-editing method that enables per-pixel parameter manipulation of image filtering by means of interactive painting. Predefined as well as custom image filters are exposed to the user, as a parametrizable composition of image operations. Brushes, as a sequences of actions, mapping user input (in terms of brush shape, flow, pressure, etc.) to arbitrary functions or convolution operations, are used to draw within the parameter space. Our prototype *flowpaint* demonstrates that interactive painting can be used to, e.g., locally tweak inadequate parametrization and, furthermore, provides a blueprint for an open photo-editing platform.

2 Introduction

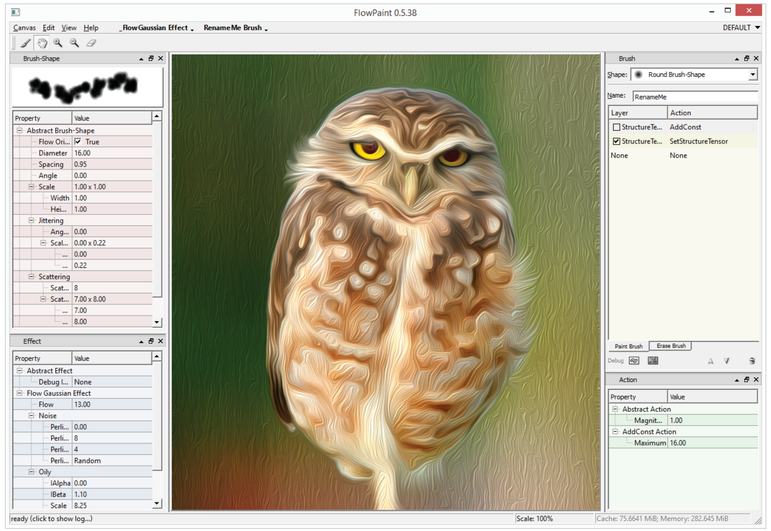
Computational, artistic stylization of images can be defined as a composition of ordered, highly parametrizable image processing steps [Kyprianidis et al. 2013]. The often tedious, trial-and-error prone process of manually tweaking an effect's outcome is typically limited to a subset of parameters of global scope. Local effect-parameters, e.g., intermediate computation results, are usually not accessible. Although, most consumer photo-editing suites apply a paint-brush metaphor (imitating real-life painting for, e.g., masking, coloring, and selecting), local parameter manipulation by painting is rudimentary supported (i.e., composing via masking and blending).

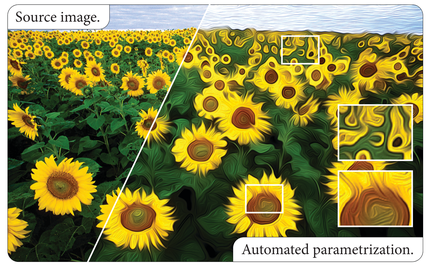

Automated parametrization.

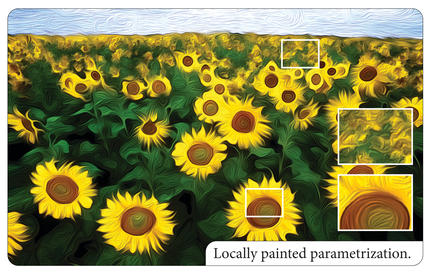
Locally painted parametrization.

Our method is targeting manipulation of local, per-pixel parametrization to support for (1) freestyle, artistic stylization and (2) modification and correction of (pre-)computed or intermediate results. In contrast to placing dynamic or static rendering primitives [Schwarz et al. 2007; Schmid et al. 2011] our method further extends the concept of specialized, local computation parametrization [Todo et al. 2007] to a generalized, user-configurable brush-painting within effect parameter-spaces.
We present an interactive, photo-editing prototype, (*flowpaint*), that uses hardware accelerated image processing and implements customizable, per-pixel parameter painting. For it, effects (*image-filters*) with exposed *parameter layers* and *brushes* as sequences of well-defined *layer-action* pairs are used. In that way, *flowpaint* represents a first step towards photo-editing ecosystems where image-filters are either user-created at run-time or distributed, shared, and customized through public image-filter repositories.

3 Method

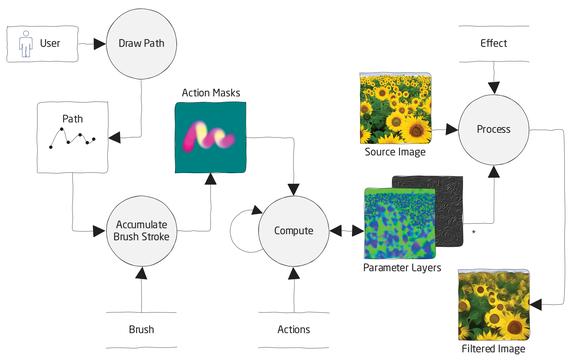




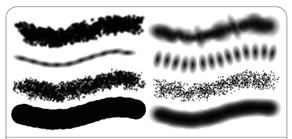

Locally painted parametrization.

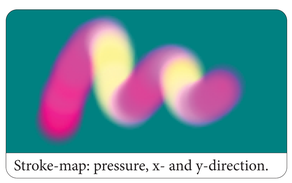
The GUI of *flowpaint* (as shown above) exposes, i.a., dynamic brush shape, brush, action, and effect configuration views. For initial processing and stylization, a library of common image convolutions and more complex effects is available (e.g., pixelizing, blurring, water coloring, oil-painting). This allows for use cases, ranging from simple brushes used to correct inadequate parametrizations, up to complex stylizations painted from scratch.

The following figure, depicts the interactive, hardware-accelerated processing pipeline used for customizable per-pixel parameter painting:



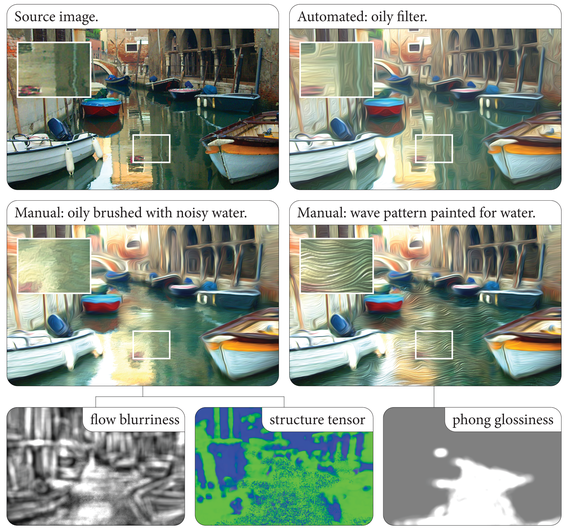
The processing of *flowpaint* is based on the following conceptual objects:
Effect: encapsulates a well defined sequences of image processing steps and exposes a set of named global or parameter layers.
Parameter Layer: describes a typed per-pixel parameter map, that can be manipulated by actions and is accessible for image-processing. A layer's extent might be specified independently of the source image extent, affecting the resolutions of parametrization.
Action: defines a computation (e.g., add constant, make zero, blur) that can be used to modify a parameter layer. The computation itself is effect independent, can have parameters, and can be assigned to any parameter layer of the computations supported type.
Brush: provides a sequence of actions mapped to parameter layers (layer-action pairs).
Brush Shape: is a distance transform used to map interpolated user-input (position, direction, pressure) to action compliant input masks (*brush strokes*).
Finally, a user drawn brush stroke is processed by iterating over all layer-action pairs and applying each action's parameter layer computation.


Brush stroke examples.


Stroke-map: pressure, x- and y-direction.

4 Results

The expressiveness of the proposed brush model renders the process of specifying brushes and effect-dependent presets highly complex, requiring image-processing expertise. The following figures depict variations of automated and manually parametrized oil-painterly stylizations, with different brush-painting techniques applied to different areas of the image;



5 Implementation and References

flowpaint is implemented using C++ and CUDA. Active parameter layers can be depicted as false-color images, allowing painting within parameter space (up to a closeup with the parameter values labeled).
Since our approach enables arbitrary, non-destructive photo-editing (similar to *constructive solid geometry*), we like to explore serialization strategies to make arbitrary effect-pipelines with their global and local parametrizations persistent and exchangable (e.g., *constructive filtered image*).

Kyprianidis, J. E., Collomosse, J., Wang, T., and Isenberg, T. 2013. State of the 'Art': A Taxonomy of Artistic Stylization Techniques for Images and Video. *IEEE Trans. Vis. Comput. Graphics 19*, 5, 866–885.

Schmid, J., Senn, M. S., Gross, M., and Sumner, R. W. 2011. OverCoat: An Implicit Canvas for 3D Painting. *ACM Trans. Graph. 30*, 4, 28:1–28:10.

Schwarz, M., Isenberg, T., Mason, K., and Carpendale, S. 2007. Modeling with Rendering Primitives: An Interactive Non-photorealistic Canvas. In *Proc. NPAR*, 15–22.

Todo, H., Anjyo, K.-i., Baxter, W., and Igarashi, T. 2007. Locally Controllable Stylized Shading. *ACM Trans. Graph. 26*, 3, 17:1–17:7.

The present work benefited from the input of Holger Winnemöller and Jan Eric Kyprianidis, who provided valuable comments and ideas to the undertaking summarized here. Original photographs kindly provided under Creative Commons license by Petras Gagilas, Wagner Machado Carlos Lemes, and Bruce Fritz.

Daniel Limberger, M.Sc.
daniel.limberger@hpi.de
+49(0)331 5509 3907

Computer Graphics Systems Group
Hasso Plattner Institute
www.hpi3d.de

Prof. Dr. Jürgen Döllner
office-doellner@hpi.de

Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
D-14482 Potsdam, Germany

www.hpi3d.de

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam
4DnD·Vis
INNOPROFILE UNTERNEHMEN REGION
Die BMBF-Innovationsinitiative Neue Länder
SPONSORED BY THE
Federal Ministry of Education and Research