ABSTRACT

The depiction of motion in static representations has a long tradition in art and science alike. Often, motion is depicted by spatio-temporal summarizations that try to preserve as much information of the original dynamic content as possible. In our approach to depicting motion, we remove the spatial constraints and generate new content steered by the temporal changes in motion. Applying particle steering in combination with the dynamic color palette of the video content, we can create a wide range of different image styles. With recorded videos, or by live interaction with a webcam, one can influence the resulting image. We provide a set of intuitive parameters to affect the style of the result, the final image content depends on the video input. Based on a collection of results gathered from test users, we discuss example styles that can be achieved with FlowBrush. In general, our approach provides an open sandbox for creative people to generate aesthetic images from any video content they apply.
we provide a means of converting motion patterns into new, ex-

The basic idea of our approach is that for each pixel in the input, 

The display of motion in static pictures has a long tradition in 

video-based graphics with the goal of artistic presentation, although 

video visualization. The authors provide a classification of existing 

video source based on optical flow and particle creation, a general 

visual music [Garcia and McGraw 2016]. Wii Motion controls [Lee 

the painting process interactively. However, at some point in time, 

composition of motion either as a new video, or as a new picture. 

from any video content, FlowBrush can be deployed for creative 

created with the technique. With the potential to create pictures 

discussion of the parameter space, and a gallery of diverse artwork 

flow for creating particle paths steered by displacement vectors. 

We introduce FlowBrush, a new technique that is based on optical 

cal flow for creating particle paths steered by displacement vectors. 

We aim to abstract the result from its input, therefore our approach 

style is transferred, is necessary. 

Our general approach can be interpreted as a transformation 

temporal coherence into spatial coherence. This concept is also 

commonly applied in scientific flow visualization. In particular, tech-

niques that combine the idea of Line Integral Convolution [Cabral 

and Leedom 1993] with image compositing techniques include pro-

cessing steps similar to FlowBrush (e.g., Jobard et al. [2002]; van 

Wijk [2002]). Aliasing artifacts resulting from large integration 

step sizes are often handled with appropriate counter-strategies 

[Weiskopf 2009]. In contrast, we see such artifacts, resulting from 

large step sizes and imprecision in the calculated flow, as an addi-

tional stylistic means, similar to single hairs on a brush that do not 

follow a perfect path along the main direction.

A subtopic in flow visualization considers the illustrative render-

ing of flow data by emphasizing important features (e.g., Brambilla 

et al. [2012]; Browning et al. [2014]; Jones and Ma [2010]; Li and 

Shen [2007]), often applying stylistic means from art. In contrast, 

our spatial representation of the optical flow is not intended to 

support analytical reasoning. With the provided parameter interac-

tion, we rather want to provide means of transforming motion and 

colors into a new, abstracted representation of the input.

Artistic applications of flow visualization can be found in the 

work by Forbes et al. [2013] and Vehlow et al. [2014]. Both ap-

proaches incorporate fluid simulations that can be influenced by pa-

rameter adjustments and control point interaction. We also provide a 

similar, interactive drawing process, however, since our approach 

is based on optical flow, the artist can additionally influence the 

output by the video input.

When using optical flow in the creation of artistic images, the 

work by Ruder et al. [2016] sticks out. It transfers the style of one 

image to the contents of a video sequence and is a multi-frame 

extension of the work of Gatsys et al. [2015], who already applied 

style transfer to single images. While the optical flow of the input 

video is not the source of the artistic features, it is important to 

maintain style consistency between adjacent frames of the output 

video. In contrast to our approach, a second input, the image whose 

style is transferred, is necessary.

1http://www.iflong.com/texts/lists/slit_scan/
3 TECHNIQUE

The technical procedure to create an image with FlowBrush is based on three steps, depicted in Figure 2: (1) calculation of the optical flow, (2) particle steering, and (3) compositing. Trajectories for individual video pixels are directly rendered on the canvas, the artist can influence the compositing by an additional filter step and by adjusting parameters to change the depiction of the trajectories.

Our application prototype is implemented in C++ and OpenCV with CUDA support for real-time processing of optical flow and bilateral filtering.

In the following, let $I^k$ be a series of input images at time steps $k$ and let $I^k(x_i)$ denote the color of pixel $i$ (with $i \in \{1, \ldots, N\}$) at location $x_i$ in the input coordinate system. Furthermore, let $J^k$ be a series of output images, where $J^1$ is a blank image, and let $T$ be a temporary image.

3.1 Optical Flow

For a video, the optical flow describes the spatial correspondence between pixels in consecutive video frames. This information can be used for numerous tasks including tracking. Although we do not track specific semantically coherent regions (e.g., objects), we make use of flow information to assemble specific motion paths for creating the output images. To calculate the optical flow between adjacent frames $I^t$ and $I^{t+1}$, we apply a dense variational method [Brox et al. 2004], provided by OpenCV with CUDA support.

We denote the resulting displacement for each pixel $i$ by $\vec{w}^t(x_i) = (u^t(x_i), v^t(x_i))^T$, where $u^t(x_i)$ is the horizontal displacement and $v^t(x_i)$ is the vertical displacement. Before computing the optical flow, we resize any input video to a height of 200 pixels and adjust the width in order to preserve a 4:3 aspect ratio for webcam feeds. For one calculation step, 5 inner and outer fixed point iterations and 10 solver iterations were performed. This parameter setting allows us to achieve interactive frame rates for calculations even on non-high-end systems. On a computer with 3.6 GHz Intel i7 CPU and an NVIDIA GeForce GTX 660 Ti, our painting algorithm performed with an average rate of 11 frames per second. Reducing the number of iteration steps can improve the performance but results in a less accurate flow.

3.2 Particle Steering

Each pixel $i$ from the input coordinate system is assigned a particle in the output coordinate system, i.e., we have $N$ particles. At the beginning, all particles reside in a common seed point $\vec{a}$. Afterward, particle $i$ is steered by the displacements $\vec{w}^k(x_i)$ that are estimated in each time step $k$ at the fixed location $x_i$. Please note that these displacements usually do not form the trajectory of any object in the input images but belong to different objects that move through location $x_i$ over time. The trace of particle $i$ in the output image at time step $t$ is the aggregation of the independent motions $\vec{w}^1(x_i), \ldots, \vec{w}^{t-1}(x_i)$ at location $x_i$ in the input image.

More formally: Using the seed point $\vec{a}$ in the output image, we set the origin $\vec{y}_i^1 := \vec{a}$ of all visualized traces of the particles $i$. For each time step $t$, the displacement vectors $\vec{w}^t(x_i)$ for all pixels $i$ in the input are calculated (see Section 3.1), and for each $i$, they are finally aggregated in an output pixel position buffer:

\[
\vec{y}_i^t := \vec{y}_i^{t-1} + \gamma \vec{w}^{t-1}(x_i) = \vec{y}_i^1 + \gamma \sum_{k=1}^{t-1} \vec{w}^k(x_i),
\]

where $\gamma$ is an amplification weight. The temporary image $T$ is initialized with $J^{t-1}$ and afterward the path increment of particle $i$ is rendered into $T$ by a line between the positions $\vec{y}_i^{t-1}$ and $\vec{y}_i^t$. The color of the line is determined by the color $I^{t-1}(x_i)$. The procedure
is depicted in Figure 3, where red boxes correspond to \( x_i \) and blue boxes correspond to \( y_i \) at the respective time steps \( k \).

Let us have a look at time step \( t = 2 \) in Figure 3. The temporary image \( T \) is initialized with the blank output image \( J^1 \). We now consider the motion \( \vec{w}_i \) between the first two input images at the pixel with the red box. As it moves upright and its color in the first image is blue, we render a blue line upright starting at the seed point in the temporary image \( T \). Afterward, \( T \) is blended with \( J^3 \) giving \( J^4 \) (see Section 3.3). At the next time step \( t = 3 \), \( T \) is initialized with \( J^4 \). The red-boxed pixel is orange and moves left by 2 pixels (\( \vec{w}_i \)). Hence, we draw an orange line into \( T \) going to the left by 2 pixels and starting at the end of the last line. Finally, \( T \) is blended with \( J^5 \), creating \( J^6 \).

Please note that the displacements only affect the output image while the positions \( \vec{x}_i \) in the input remain unaltered (see red box in Figure 3). Since both coordinate systems are different, the resolution of the output image is independent from the input. Hence, our approach can generate high-resolution images from low-resolution input. We propose a resolution 4 times the size of the input for a canvas that can be used for both small and large step sizes of the trajectory painter.

### 3.3 Compositing

The compositing step blends the previous output image \( J^{t-1} \) with the adjusted temporary image \( T \) from the current step

\[
J^t = \alpha T + (1 - \alpha)J^{t-1}
\]

with a blending weight \( \alpha \in [0, 1] \). This iterative alpha blending approach is also applied for interactive vector field visualizations.

Only the previous and the current time step are required for computation, which makes this method efficient for real-time applications [Weiskopf 2009]. In order to modify the style of our image, a bilateral filter [Tomasi and Manduchi 1998] can be applied on the temporary image \( T \) before blending.

### 3.4 Parameters

The user can influence two numeric parameters, and three triggers to enable the bilateral filter, seed point randomization, and an alternative direction-to-color mapping. We choose those parameters because of their intuitive interplay:

**Blending.** Changing the blending parameter \( \alpha \) highly influences the style of the output. The user is free to adjust this parameter dynamically during the drawing process. Parameter \( \alpha \) can be interpreted as a metaphor for the amount of paint that is used to draw on a canvas. Higher values correspond to more color on the brush, whereas low values change the output in a more subtle way.

**Step Size.** This parameter represents a multiplier \( \gamma \in [1, 20] \) for adjusting the length of the displacement vector \( \vec{w}_i \), used in the context of particle steering:

\[
y_i(t) = y_i(t-1) + \gamma \vec{w}_i(t-1)(\vec{x}_i)
\]

Increasing \( \gamma \) amplifies the motion and distributes the pixels faster over the output image. This parameter can be interpreted as a metaphor for the stroke length of a brush. We will further discuss the influence of this parameter in Section 4.1.

**Bilateral Filter.** Without any additional filtering, we call the style of our output images the plain style, drawing all trajectories straight into the image, with individual traces clearly visible. By applying a bilateral filter [Tomasi and Manduchi 1998] to the temporary image \( T \), we provide a second style that smooths areas into homogeneous regions while preserving edges, an artistic effect similar to artificially generated oil paintings. Its parameters \( (\sigma_{\text{color}} = 30, \sigma_{\text{space}} = 20) \) are chosen empirically and kept fixed. The remaining choice is to enable or disable it.

**Direction to Color.** Generally, the color of painted trajectory segments depends on the currently visible colors in the input. As an alternative, we provide a direction-to-color mapping as it is commonly applied in illustrations of dense, optical flow. Since every direction is assigned to another color, this feature can be applied to influence the result by grouping motion in similar directions together in the composition to achieve more homogeneous colored regions. Additionally, not all environmental settings might provide a satisfying color palette (e.g., low light settings) and the alternative color mapping is not influenced by this factor. Figure 4 shows examples of the color coding for four example videos that will be discussed in detail in Section 4.1.

**Random Seed.** Without the random seed enabled, the pixels begin to distribute over the output, creating a noise pattern that conveys less motion patterns than at the beginning. Therefore, all pixels have to be reset to a new seed point. This seed point can either be set manually, by clicking on an image position with the mouse, or randomly, after a specific number of processed frames.
We discuss our results on two different levels. First, we investigate the parameter space and how changes in the aforementioned parameters influence the resulting image. Second, we provide examples of how different video input results in various images of different style and expression.

4 RESULTS

We provide our tool to a testing group, asking them to try out the application and feel free to experiment with the creation of images. We asked them to send us back resulting images along with a questionnaire on how they created the results. As a result, we received 34 images. Combined with our own experimental results, we want to provide a glimpse into the vast amount of creative possibilities that can be realized with FlowBrush.

**Filtered Images.** The bilateral filter can be applied to add an alternative artistic style to the result. Depending on $\alpha$, this can either resemble oil paintings (Figure 6a, $\alpha > 0.3$) or aquarelles (Figure 6b, $\alpha < 0.1$).

**Space-Filling Images.** Small $\alpha$ values and large step sizes $\gamma$ are suitable to fill the output image completely while smoothing the transition between individual trajectories. The resulting images (Figures 6c and 6d) cover the canvas completely. This style could also be used to create backgrounds for a painting.

**Nebular Structures.** Values of $\alpha < 0.05$ lead to subtle changes in the result, similar to the dry-brush painting technique. Fast motion leads to nebular structures. If a motion is abruptly ended and the input stands still, the pixels at the end positions become visible, creating edges and points in the image (Figures 6e and 6f).

**Images Colored by Direction.** Results applying the direction-to-color mapping were rare. However, one test user claimed that his setting at home did not provide good colors, so he switched to the alternative mode (Figures 6g and 6h).

**Repetitive Motion Images.** As we presented in the parameter series (Figure 5), repetitive motion can result in images resembling the source video. As an additional example, Figure 7a shows the result of a video from a waterfall.

**Seed Point Compositions.** Some users did not rely on the seed point randomization. They manually set seed points on the canvas to compose an image from the flow. The source was either additionally influenced from a webcam (Figure 7b) or a pre-recorded video (Figures 7c and 7d).

**Structure Preserving Images.** Starting from an initial seed point, zooming motion can create a pixel distribution that partially resembles the original video content, allowing the artist to include real-world content into the motion patterns. Figures 7e and 7f show examples with faces, composited into the resulting image. Both images were created with $\alpha < 0.3$, and step size $\gamma < 3$. Figure 7f was created with additional bilateral filtering.

![Image 54x621 to 558x708](image_url)

Figure 4: Direction-to-color mapping for four example videos.

4.1 Parameter Space Exploration

Figure 5 shows parameter series for the blending parameter $\alpha$ and the step size $\gamma$. The presented examples are from four different videos with continuous motion patterns. For each video, a sample of 700 frames was processed to create an output image for seven seed points, distributed over the canvas. Additionally, the results for the bilateral filter are included for $\alpha = 1.0$ for direct comparison.

Figure 5a results from a video with a fireplace. The continuous upward motion of the flames creates an output that also resembles a fire. Figure 5b results from a ship’s propeller filmed under water. The same video was also applied to create the image in Figure 1. Figure 5c was created with a video from a rotating spiral. The resulting image reflects this motion pattern. Figure 5d shows the result of a water vortex, filmed from top. As in Figure 5b, this video alters between different flow velocities, resulting in different shapes at the seed points (see $\gamma = 1$).

The blending parameter influences how fast new time steps will become visible in the output image. No or small motion leaves pixels at their old positions and the pixels become better visible over time for a small $\alpha$ value. We suggest applying small values to create diffuse background textures, or nebular structures that begin to show details if the motion stops abruptly (see Section 4.2). Vice versa, a high $\alpha$ value composes new pixels and their trajectories directly into the image, analog to a brush with much paint on it.

Increasing the step size $\gamma$ results in an amplified displacement, so that even small motion has a significant influence on the output image. In combination with increasing $\alpha$ values, individual trajectories appear more prominent. Larger step sizes lead to more chaotic results, which can be an interesting aspect for artistic expression.

The bilateral filter shows the smallest effect on Figure 5c. This is mainly due to the fact that the spiral motion overwrites results of old time steps faster and the compositing requires some iterations to fully incorporate the filter results. For the other three series, the filter effect becomes clearly visible, especially for small step sizes ($\gamma = 1$). Regions of similar color become homogeneous and salient edges remain in the image, preventing it from becoming blurred.
Figure 5: Parameter series for different video sources. A set of 7 seed points, each switched after 100 frames, was used to depict the influence of the blending parameter and the step size.
Figure 6: Examples of images created with FlowBrush.
Figure 7: Examples of images created with FlowBrush.
We presented a new technique to depict motion from a video source. The source code is available on our website. The CUDA-based implementation of the relevant calculation steps allows a deployment of the application for live performances (Figure 8) and interaction with an audience, for example in art galleries, open house events, or as individual art projects in general. The source code is available on our website. For systems without CUDA support, we provide an alternative, CPU-based calculation method for optical flow.

Future extensions could incorporate more computer vision support. Spatial image segmentation could be applied to use FlowBrush as a coloring book, allowing one to draw only in specific segments at a time. Temporal segmentation, for example by scene detection, could be applied to generate a sequence of output images each covering consistent content. Object detection could help steer specific particles directly. For example, detection and tracking of individual hands could be applied to map the hand motion to specific particles. This would result in a painting process similar to other virtual painting devices. Furthermore, the proposed approach for particle steering is only one of many possible computational models. Future work could evaluate how different approaches (e.g., based on pathlines) influence the result.

We see FlowBrush as a conversion tool that transforms motion input into artistic images. What the artist chooses for video input (e.g., particles directly. For example, detection and tracking of individual hands could be applied to map the hand motion to specific particles. This would result in a painting process similar to other virtual painting devices. Furthermore, the proposed approach for particle steering is only one of many possible computational models. Future work could evaluate how different approaches (e.g., based on pathlines) influence the result.

We see FlowBrush as a conversion tool that transforms motion input into artistic images. What the artist chooses for video input (e.g., particles directly. For example, detection and tracking of individual hands could be applied to map the hand motion to specific particles. This would result in a painting process similar to other virtual painting devices. Furthermore, the proposed approach for particle steering is only one of many possible computational models. Future work could evaluate how different approaches (e.g., based on pathlines) influence the result.

We see FlowBrush as a conversion tool that transforms motion input into artistic images. What the artist chooses for video input (e.g., particles directly. For example, detection and tracking of individual hands could be applied to map the hand motion to specific particles. This would result in a painting process similar to other virtual painting devices. Furthermore, the proposed approach for particle steering is only one of many possible computational models. Future work could evaluate how different approaches (e.g., based on pathlines) influence the result.

We see FlowBrush as a conversion tool that transforms motion input into artistic images. What the artist chooses for video input (e.g., particles directly. For example, detection and tracking of individual hands could be applied to map the hand motion to specific particles. This would result in a painting process similar to other virtual painting devices. Furthermore, the proposed approach for particle steering is only one of many possible computational models. Future work could evaluate how different approaches (e.g., based on pathlines) influence the result.

ACKNOWLEDGEMENTS

We thank all participating artists for their contributions and the German Research Foundation (DFG) for financial support within project B01 and B04 of SFB/Transregio 161.

REFERENCES


3 http://go.visus.uni-stuttgart.de/flowbrush