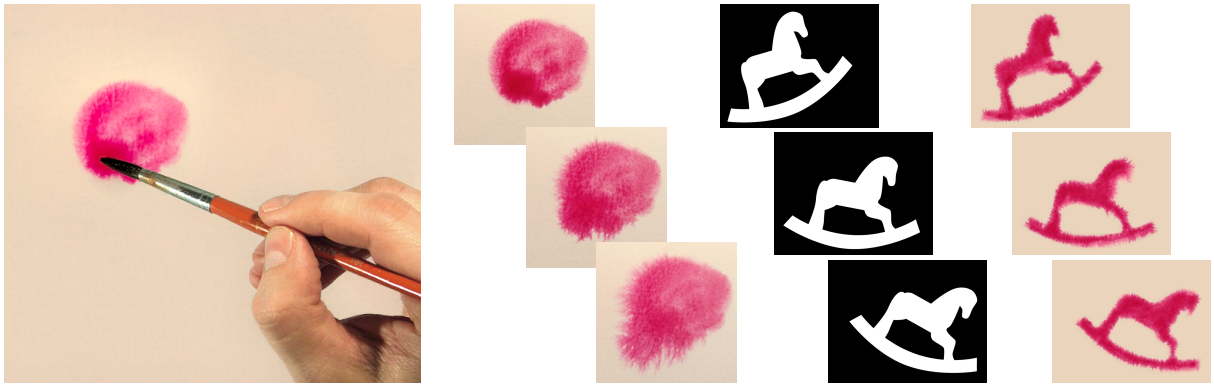
Pacific Graphics 2021
E. Eisemann, K. Singh, and F.-L Zhang
(Guest Editors)

Volume 40 (2021), Number 7

Fluidymation: Stylizing Animations Using Natural Dynamics of Artistic Media

A. Platkevič[1], C. Curtis[2], D. Sýkora[1]

[1]Czech Technical University in Prague, Faculty of Electrical Engineering, Czech Republic
[2]Google Research, USA



(a) capturing an exemplar (b) captured sequence (c) target animation (d) stylized output

Figure 1: *Fluidymation in action—an artist prepares a style exemplar by dropping some watercolor paint onto wet paper (a). We record this process in a video sequence that captures not only the appearance of the artistic medium but also its dynamic properties, e.g., pigment diffusion (b). The user provides a target animation (c) onto which we transfer the exemplar's appearance and dynamics. The resulting animated sequence (d) moves as if the artistic medium diffuses across the paper in the direction and speed of the prescribed target animation (see our supplementary video).*

Abstract

We present Fluidymation—a new example-based approach to stylizing animation that employs the natural dynamics of artistic media to convey a prescribed motion. In contrast to previous stylization techniques that transfer the hand-painted appearance of a static style exemplar and then try to enforce temporal coherence, we use moving exemplars that capture the artistic medium's inherent dynamic properties, and transfer both movement and appearance to reproduce natural-looking transitions between individual animation frames. Our approach can synthetically generate stylized sequences that look as if actual paint is diffusing across a canvas in the direction and speed of the target motion.

CCS Concepts
• ***Computing methodologies*** → ***Non-photorealistic rendering;***

1. Introduction

Example-based style transfer to video sequences has seen remarkable advancements recently, both in terms of visual quality [FJS*17, JvST*19] and reduction of computational overhead [FCC*19, KSM*19, TFF*20]. Unlike still images, when stylizing a video, one needs to take temporal continuity into account. Processing the frames independently usually causes the output sequence to flicker [FLJ*14]. A similar effect is common for traditional hand-colored animations created manually in a frame-by-frame fashion [Wel19]. Due to the limited control over physical artistic media, it is usually impossible for the artist to achieve a perfect continuity in time. This limitation is commonly understood as

an important artistic feature of hand-colored animations. However, watching such flickering animation may cause eye strain, and the viewer can become tired after a while [KSHTB*03]. On the other end of the spectrum there are video stylization approaches which guarantee high temporal consistency [SED16,RDB18]. Their drawback is that the stylized content looks glued on the moving objects, which breaks the impression of being painted frame by frame. Other techniques [BCK*13,FLJ*14,JvST*19] let the user fine-tune the amount of temporal flickering to balance between the two extremes. However, the synthetically generated flicker they produce usually does not convincingly reproduce temporal dynamics seen in actual hand-colored animations.

In this paper we propose a novel approach to generating temporally consistent stylization of video sequences which we call "fluidymation". The key idea is to transfer not only the texture from a style exemplar, but also the dynamic properties of the artistic medium being used. For example, instead of using a still image of watercolors, we use a video recording of watercolor paint diffusing across paper. The aim is to reuse this natural dynamic to convey the motion in the target sequence. In contrast to previous methods, which need to enforce temporal consistency explicitly, our approach leverages the dynamic properties of the artistic medium itself, to let the paint move naturally as it would in real life (see Fig. 1).

2. Related Work

A traditional approach to image stylization imitates the artwork creation process by overlaying a set of automatically distributed brush strokes (colored [Hae90] or textured [Her98, Her01]) to produce the final stylized image. In the case of video stylization, the strokes are displaced according to the motion in the scene, e.g., using optical flow estimation [HE04] or a 2D projection of the movement of 3D objects [Mei96]. The stroke aggregation process allows for a wide range of customization thanks to the possibility of altering the appearance of individual brushes [LBDF13].

Another important branch of stylization techniques uses procedural filtering [BLV*10, MSS*18] to enable artistic control via a manual tweaking of filtering parameters. Those approaches can mimic a wide range of styles, including watercolor, oil painting, or charcoal drawing. Bousseau et al. [BKTS06] proposed a filtering pipeline designed to simulate watercolor. To avoid the so-called *shower door effect* [Mei96], they employ temporal morphological filtering and texture advection [BNTS07]. Similarly, in [BSM*07], pre-defined 2D patterns are successively transformed in a shape-preserving manner to match the movement of objects in an animated 3D scene.

In example-based stylization [HJO*01], the concept of image analogies is used to change the appearance of the target image according to an example of a stylized source image. The analogy can be further extended using additional paired source and target guiding channels that encode other important features such as region boundaries [BCK*13] or illumination changes [FJL*16]. Those channels are then plugged into a guided variant [KNL*15,FJL*16] of patch-based synthesis [KEBK05, WSI07] to produce the final stylized image. Such an approach can also be extended into the video domain [BCK*13, FJS*17, JvST*19] where an additional guide is used to ensure temporal consistency.

Recently, neural style transfer became popular thanks to the seminal work of Gatys et al. [GEB16] showing that responses of a VGG network pre-trained on an image recognition dataset [SZ14] can be used to capture some aspects of artistic style. This technique was later extended to handle temporal consistency in video sequences [RDB18]. This success was later followed by image translation networks [IZZE17, TFF*20] that are able to match, or in some aspects even outperform the results of guided patch-based synthesis.

The techniques mentioned above have a common drawback. Although they can deliver temporally coherent stylized animations, their output feels rather artificial—the realistic transition phenomena typical for natural artistic media are not taken into account. In our work, we aim to perform example-based stylization of animations by transferring not only the appearance of natural media but also their dynamic properties. Thanks to this extension, we can mimic the impression that the paint propagates in unison with the motion of animated shapes.

Our approach is similar in motivation to the problem of appearance transfer to fluid simulations [BBRF14, JFA*15] where the style of an image or video exemplar is transferred to a target flow field. However, Browning et al. [BBRF14] use only a few hand-drawn images as a style exemplar, and thus cannot take into account the artistic medium's dynamic properties. The method of Jamriška et al. [JFA*15] can use video as a style examplar, but it does not manipulate the motion's speed and direction, and thus produces drifting and warping effects that break the illusion of physical paint following the target motion. An alternative approach is the use of neural style transfer in the context of fluid simulations [KAGS19, KAGS20]. However, those techniques consider only static exemplars to provide high frequency details, and the fluid's gross movement is determined by the original simulation.

Although physical models can be employed to achieve realistic simulation of appearance as well as dynamic effects of natural artistic media such as watercolor [CAS*97], oil paint [BWL04], or pastel [HLFR07], those techniques do not deal with the issue of preserving temporal consistency in animation.

3. Our Approach

The inputs to our algorithm are the following sequences (see Fig. 2):

- S^{rgb}—a sequence of l_S images serving as a style exemplar, captured by a camera perpendicular to a canvas,
- S^{mask}—a sequence of l_S binary masks denoting the presence of artistic media in the exemplar image,
- T^{mask}—a sequence of l_T binary masks that define the placement of artistic media in the target animation.

Optionally, the user may also specify two additional sequences of source and target flow fields: S^{flow} and T^{flow}. These are also required as an input to our algorithm, but when not available we provide a solution for how to approximate them automatically from S^{mask} and T^{mask} (see Appendix A).

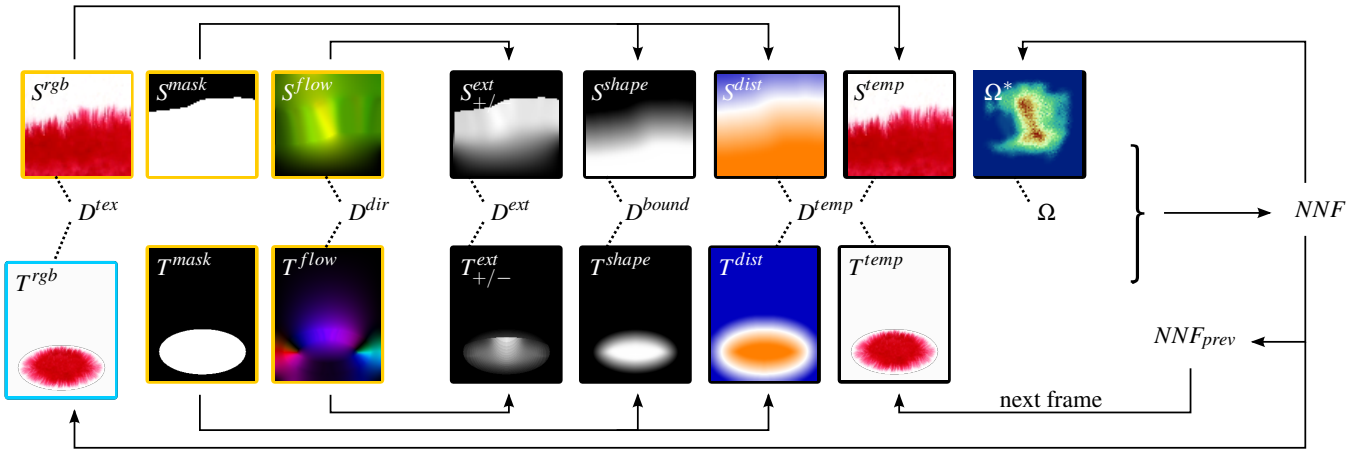


Figure 2: *An overview of all inputs (yellow border), guiding channels (black border), and the output frame (blue border) including an illustration of how those are derived and plugged into the patch similarity measure D. During the optimization, the nearest-neighbour field NNF is iteratively refined to minimize D over all source patches taken from S^{rgb} and used in the output frame T^{rgb}. The 3D occurence map Ω^* of the uniformity term Ω reflects the current source patch utilization to prevent excessive use of certain patches. To encourage temporal coherence and speed up the convergence, the nearest-neighbour field from the previous frame NNF_{prev} is used during the optimization.*

The output of our method is a sequence of images T^{rgb} composed of a selection of 2D patches taken from S^{rgb} that can be rotated by an angle θ prescribed in an orientation field T^{rot}. For every pixel $q = (j, \hat{x}, \hat{y})$ in the target sequence T^{rgb} where j is q's frame number and $[\hat{x}, \hat{y}]$ are its 2D coordinates we seek a suitable patch s centered at a pixel $p = (i, x, y)$ in the source sequence S^{rgb} and rotated by θ stored in $T^{rot}(q)$.

In contrast to the method of Jamriška et al. [JFA*15] our new formulation provides two important advantages: (1) source patches can be retrieved from the entire exemplar sequence and rotated. This helps to increase the variety of exemplars to better convey the extent of target motion and keep the source patches aligned with its direction; and (2) explicit advection of previously synthesized frames is no longer required. Instead, we use an incrementation of i as described in Section 3.4.

To obtain T^{rgb} and T^{rot} we proceed frame by frame and for each frame j we minimize the following energy:

$$\mathrm{E}(S, T_j) = \sum_{q \in T_j} \min_{p \in S} \mathrm{D}(p, q),$$

where the patch similarity function D is defined as follows:

$$\begin{aligned}\mathrm{D}(p,q) = \sum_{p' \in s, q' \in t} & \mathrm{D}^{tex}(p',q') + w^{bound}\mathrm{D}^{bound}(p',q') + \\ & w^{dir}\mathrm{D}^{dir}(p',q') + w^{ext}\mathrm{D}^{ext}(p',q') + \\ & w^{temp}\mathrm{D}^{temp}(p',q') + w^{uni}\Omega(p').\end{aligned}$$

Here s and t are patches centered at a source pixel location p and a target pixel location q, respectively while p' and q' are coordinates of pixels covered by 2D patches s and t of size $n \times n$. D^{tex} measures the consistency of texture and D^{bound} maintains alignment of mask boundaries, i.e., during the optimization those two terms mostly affect the p's spatial coordinates $[x, y]$. The term D^{dir} upholds the consistency of motion direction controlled by the parameter θ, D^{ext} encourages the selection of patches that have an appropriate extent of motion, i.e., influences the temporal coordinate i of the pixel p, and D^{temp} keeps the output coherent in time. Finally, the uniformity term Ω helps to avoid the overuse of particular source patches that may cause so called *wash-out* artifacts [JFA*15], i.e., a lack of visual variety seen in the original source exemplar. The influence of each individual term D^* is balanced relatively to D^{tex} using a weighting factor w^*. All terms D^{tex}, D^{bound}, D^{dir}, D^{ext}, D^{temp}, and Ω are described in more detail in the following sections.

3.1. Texture consistency and boundary effects

The computation of D^{tex} and D^{bound} is similar to that of Jamriška et al. [JFA*15] except for the rotation $T^{rot}(q)$ of the corresponding source patch p which needs to be taken into account when computing those two terms. The term D^{tex}, responsible for local visual similarity of the generated texture to the source one, is computed as follows:

$$\mathrm{D}^{tex}(p',q') = \|S^{rgb}(p') - T^{rgb}(q')\|^2.$$

The term D^{bound} facilitates the expression of directional effects apparent at boundaries of a painted area. Additional guidance channels S^{shape} and T^{shape} are generated by filtering the binary masks using Gaussian blur with radius b (see Fig. 2). The term itself is then computed as

$$\mathrm{D}^{bound}(p',q') = \|S^{shape}(p') - T^{shape}(q')\|^2.$$

3.2. Motion orientation alignment

The term D^{dir} helps to keep the flow direction of a source patch s centered at a pixel p aligned with the direction of a target patch t centered at a location q. To accomplish this goal we evaluate D^{dir} as follows:

$$\mathrm{D}^{dir}(p',q') = |\tan(S^{flow}_{\theta}(p') + T^{rot}(q') - T^{flow}_{\theta}(q'))|,$$

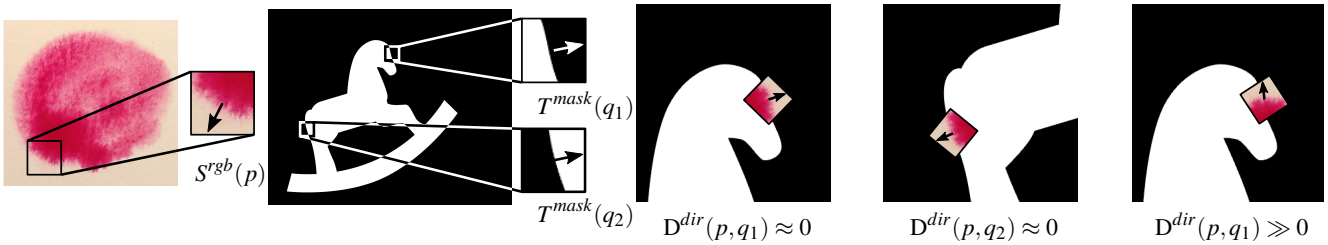


Figure 3: *The influence of the rotation alignment term when evaluating the distance of patches at pixels p and q. Arrows signify the general directions of flow fields $S_\theta^{flow}(p)$ and $T_\theta^{flow}(q)$. The alignment term D^{dir} is minimal for values of θ that lead to a perfect alignment of flow field orientations. The color and mask channels are shown only for clarity; they do not participate on the calculation of D^{dir}.*

where $S_\theta^{flow}(p')$ and $T_\theta^{flow}(q')$ are the respective orientations of the flow fields at pixels p' and q' in radians. An orientation $T^{rot}(q')$ of a source patch s centered at the target pixel q' is added to keep the directions consistent. Using this equation, similarly aligned flow orientations (possibly in the opposite direction) are preferred, and conversely, mappings resulting in an orientation close to being perpendicular to the direction of the target flow are strictly avoided (see Fig. 3).

3.3. Motion extent control

In the method of Jamriška et al. [JFA*15] the selection of source patches is limited only to those that are available in the current animation frame. This requirement imposes restrictions on the content of source and target animation, e.g., the extent of motion in the source sequence needs to be roughly in proportion to the motion in the target sequence, otherwise the texture evolution would not look motivated by the movement of the paint. This may lead to undesirable drifting artifacts where the motion of the source exemplar is superimposed on the target motion (see the comparison with the method of Jamriška et al. [JFA*15] in our supplementary video).

In our approach we increase the flexibility of synthesis by enabling retrieval of patches from the entire source sequence. However, to achieve plausible results, we need to guide the patch selection according to the past and future motion amount at each location and distinguish between parts where the material is subject to motion and those which are mostly stationary (see Fig. 4a). To accomplish this we use I^{flow} (see Fig. 4b) to produce two guidance channels: I_+^{ext} and I_-^{ext} (where I denotes either S or T).

The forward channel I_+^{ext} is constructed by accumulating the amount of motion at each pixel since the start of the sequence, i.e.,

$$I_+^{ext}(i,x,y) = \|acc_+(i,x,y)\|,$$

where i is a frame number, $[x,y]$ is a pixel location and

$$acc_+(i,x,y) = \begin{cases} acc_+(i-1,x,y) + I^{flow}(i-1,x,y), & \text{if } i > 1, \\ (0,0) & \text{otherwise.} \end{cases}$$

In addition to accumulation we zero acc_+ at pixels that are outside the mask I_i^{mask} to make sure the accumulation is restarted at coordinates where the material appears repeatedly. An example of I_+^{ext} is illustrated in Fig. 4c.

Since we would like to enable material diminishing which the exemplar sequence may not contain or is not physically plausible we introduce a complementary guiding channel (illustrated in Fig. 4d) that is calculated in the opposite direction of time, i.e.:

$$I_-^{ext}(i,x,y) = \|acc_-(i,x,y)\|$$

where

$$acc_-(i,x,y) = \begin{cases} acc_-(i+1,x,y) + I^{flow}(i,x,y), & \text{if } i < l_I, \\ (0,0) & \text{otherwise.} \end{cases}$$

Similarily to I_+^{ext} we zero the accumulator $acc_-(i,x,y)$ for pixels $[x,y]$ that are outside the mask, i.e., $I^{mask}(i,x,y) = 0$.

Using these two additional guiding channels D^{ext} is computed as follows:

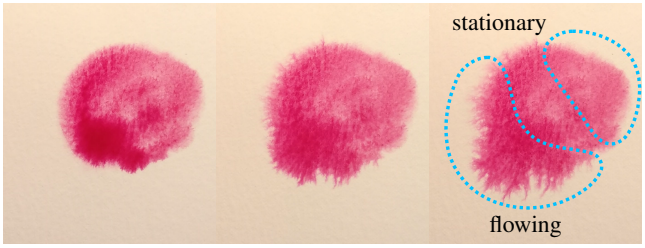
$$\mathrm{D}^{ext}(p',q') = \|S_+^{ext}(p') - T_+^{ext}(q')\|^2 + \|S_-^{ext}(p') - T_-^{ext}(q')\|^2.$$

The final step towards enabling the reversibility of time is that during the evaluation of distance between two patches that have roughly opposite flow directions, values in T_+^{ext} and T_-^{ext} are swapped before being subtracted from S_+^{ext} and S_-^{ext}.

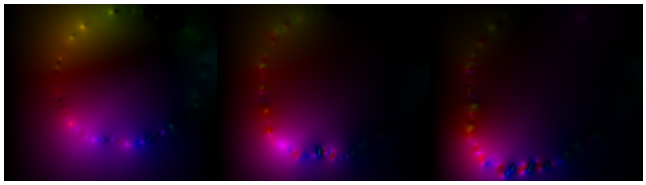
3.4. Temporal coherence

Previous approaches to maintain temporal coherence in guided patch-based synthesis [BCK*13, JFA*15, FJS*17, JvST*19] use a warped version of previous frames to encourage selection of patches that have similar content to those in previously synthesized frames.

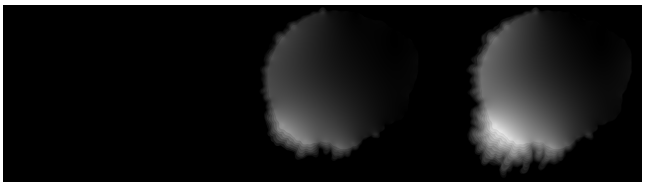
In our approach, we take into account the fact that exemplar patches are being retrieved from the entire sequence, i.e., patch coordinates include not only the spatial location but also an index of the source frame. If we shift this index by some amount Δi we can get an appearance similar to that if we perform warping of the previous patch using the motion field of the source sequence. Moreover, thanks to orientation alignment (see Section 3.2) after shifting in time the patch will also follow the motion direction of the target sequence. What remains to be determined is the actual Δi, i.e., the number of frames the index is shifted to meet the amount of motion in the target sequence.



(a) *A splotch of watercolor in S^{rgb} manifesting different looks at flowing and stationary parts*



(b) *Corresponding flow field S^{flow}. Angle S^{flow}_{θ} is encoded as hue and magnitude S^{flow}_{m} as intensity.*



(c) *Derived guidance channel S^{ext}_{+}*

(d) *Derived guidance channel S^{ext}_{-}*

Figure 4: *Motion extent control illustrated on three frames picked from a longer sequence*

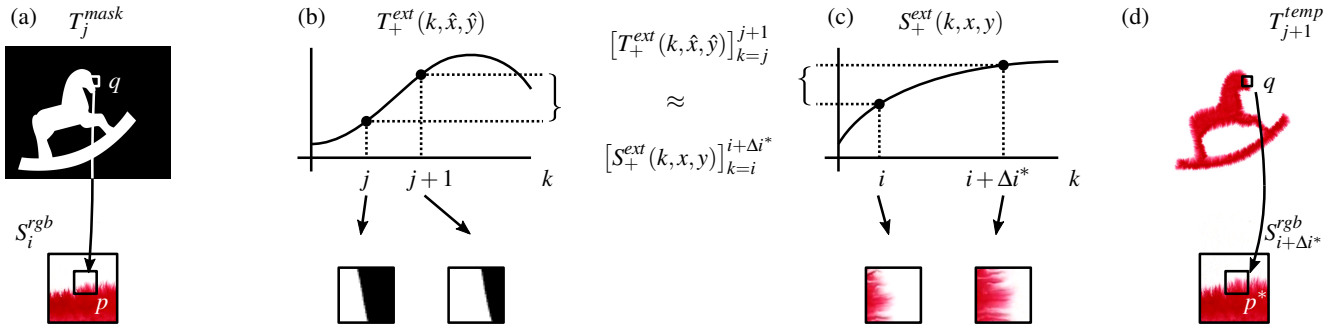


Figure 5: *Determining the time coordinate shift Δi at the source pixel $p=(i,x,y)$ that corresponds to target pixel $q=(j,\hat{x},\hat{y})$ (a). First, the amount of motion at q is determined by the difference stored in T^{ext}_{+} channel (b). Then a time shift Δi^* is found that most closely matches the target movement amount (c). Finally, the time coordinate i of the corresponding patch p is shifted by Δi^* to get $p^*=(i+\Delta i^*,x,y)$ (d).*

Let us assume, for now, that flow directions at the mapped source coordinates match the target ones (i.e., they point in the same direction besides being aligned). To compute Δi under this assumption we accumulate the amount of motion S^{ext}_{+} and T^{ext}_{+} at each pixel of the source and target sequence (see Section 3.3) from which we can compute a relative amount of motion between frames j and $j+1$ at the 2D location $[\hat{x},\hat{y}]$ of target pixel q:

$$[T^{ext}_{+}(k,\hat{x},\hat{y})]^{j+1}_{k=j} = T^{ext}_{+}(j+1,\hat{x},\hat{y}) - T^{ext}_{+}(j,\hat{x},\hat{y}).$$

Let $p=(i,x,y)$ be coordinates of a corresponding source patch to a target patch q in the previous frame j and the amount of motion at 2D location $[x,y]$ between frames i and $i+\Delta i$ is:

$$[S^{ext}_{+}(k,x,y)]^{i+\Delta i}_{k=i} = S^{ext}_{+}(i+\Delta i,x,y) - S^{ext}_{+}(i,x,y).$$

Then the optimal time shift Δi can be retrieved as follows:

$$\Delta i^* = \underset{\Delta i \geq 0}{\arg\min} \left| [T^{ext}_{+}(k,\hat{x},\hat{y})]^{j+1}_{k=j} - [S^{ext}_{+}(k,x,y)]^{i+\Delta i}_{k=i} \right|.$$

When the optimal Δi^* is known the location of the corresponding patch s^* centered at pixel p^* is set to $p^*=(i+\Delta i^*,x,y)$ (see Fig. 5).

When the target and rotated source flow at pixels p and q are in opposite directions, the frame index is shifted backward, i.e., $\Delta i \leq 0$, and instead of the past motion amount S^{ext}_{+}, we use future motion amount S^{ext}_{-} (see Section 3.3).

As soon as all corresponding shifted source patches are known, we can produce a target temporal guide T^{temp} using voting operation [WSI07], i.e., we compute a weighted average of collocated pixels in the overlapping patches. The source part of the temporal guide $S^{temp}=S^{rgb}$ and the final term D^{temp} is computed accordingly to Fišer et al. [FJS*17]:

$$\mathrm{D}^{temp}(p',q') = \|S^{temp}(p') - T^{temp}(q')\|^2.$$

Since the temporal guide is meaningful only in regions where the previous mask overlaps the current one, the weight w^{temp} is set to zero at locations where there is no overlap. In the first frame, w^{temp} is set to zero as no previous frame is available.

When the movement in the target sequence is much stronger than in the exemplar, $[S_+^{ext}(k,x,y)]_{k=i}^{i+\Delta i^*}$ may not be sufficiently large to match that of $[T_+^{ext}(k,\hat{x},\hat{y})]_{k=j}^{j+1}$, for example when $i+\Delta i^*$ reaches either end of the source sequence. In this case the synthesized animation may lag behind the target sequence. This occurs when the assigned source pixel p is too far or too near to the boundary of the source mask in contrast to the target patch location q (see a demonstration of this effect at the end of our supplementary video).

To mitigate this issue we employ a similar strategy as in Jamriška et al. [JFA*15]. We introduce a spatially varying modulation $m(q)$ of the temporal coherence weight w^{temp} based on the difference of signed distance fields of the source and target masks as follows:

$$m(j,\hat{x},\hat{y}) = \begin{cases} 0, & \text{if } T^{mask}(j-1,\hat{x},\hat{y}) = 0, \\ s(|T^{dist}(j,\hat{x},\hat{y}) - S^{dist}(i+\Delta i,x,y)|), & \text{otherwise,} \end{cases}$$

where T^{dist} and S^{dist} are signed distance fields defined in Appendix A and s is a smoothstep function defined as

$$s(v,m_l,m_u) = \begin{cases} 0, & \text{if } v \le m_l, \\ 1, & \text{if } v \ge m_u, \\ 3v'^2 - 2v'^3 & \text{otherwise,} \end{cases}$$

where $v' = (v-m_l)/(m_u-m_l)$ and m_u and m_l are configurable upper and lower thresholds.

3.5. Spatial uniformity

During the minimization of energy E, a smaller fraction of source patches may become more preferred due to their tendency to produce lower matching error (e.g., patches with mostly homogeneous color). Jamriška et al. [JFA*15] suppress this wash-out artifact by using an additional hard constraint that enforces uniform utilization of source patches. In our scenario, however, the entire sequence is used for synthesis and thus strictly uniform utilization is not reasonable. Instead, we adopt a soft constraint similar to that used by Kaspar et al. [KNL*15]. They use a 2D occurrence map Ω^* which stores the utilization of individual source patches. It allows us to adaptively penalize a patch at the location p whenever it was already used too often.

In our scenario, we consider not only the 2D location of patches but also their orientations and positions in time. Due to this fact we need to extend Ω^* into 3D and perform the occurrence accumulation with respect to the orientation of individual patches. The formula for our 3D occurrence map becomes:

$$\Omega^*(i,x,y) = \left| \{ q \in T^{mask} \,|\, (i,x,y) \in \mathcal{N}^{\theta}(k,\hat{x},\hat{y}) \} \right|,$$

where $(k,\hat{x},\hat{y})$ are the coordinates of a nearest-neighbour patch q, $\theta = T^{rot}(k,\hat{x},\hat{y})$, and $\mathcal{N}^{\theta}(k,\hat{x},\hat{y})$ represents a cuboid with dimensions $n \times n \times (2r+1)$, centered at pixel $(k,\hat{x},\hat{y})$, and rotated by θ radians in the x,y-plane. The configurable parameter r sets the temporal dimension of the penalized neighbourhood. This occurrence map is computed for each frame separately. Similarly to Jamriška et al. [JFA*15], we treat the boundary segments S^{bound} and T^{bound} and interior segments S^{int} and T^{int} separately, obtaining the following formula for ω^*:

$$\omega^* = \begin{cases} n^2(2r+1)|T^{bound}|/|S^{bound}| & \text{in the border segment, and} \\ n^2(2r+1)|T^{int}|/|S^{int}| & \text{in the interior segment.} \end{cases}$$

Following Kaspar et al. [KNL*15], we set the uniformity term $\Omega(p')$ at the pixel $p' = (i,x,y)$ to:

$$\Omega(p') = \frac{\Omega^*(p')}{n^2 \cdot \omega^*}.$$

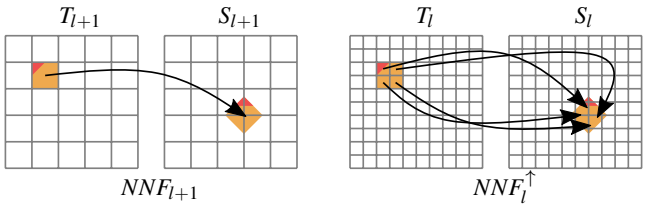
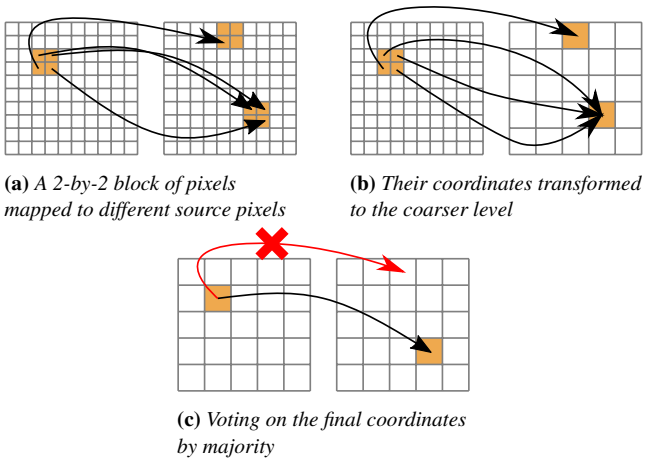


Figure 6: *Upscaling a nearest-neighbour field with a target window mapped to a rotated source window*



(a) *A 2-by-2 block of pixels mapped to different source pixels*

(b) *Their coordinates transformed to the coarser level*

(c) *Voting on the final coordinates by majority*

Figure 7: *Downscaling an NNF mapping*

3.6. Optimization

To minimize E we use a multi-resolution EM-like optimization scheme of Wexler et al. [WSI07]. The number of resolution pyramid levels is set to $\lfloor \log_2 d_{min}/n \rfloor$, where d_{min} is the minimum dimension of the full-resolution source and target images and n is the patch size. This ensures that the patch size gets close to, but does not exceed, the size of the images in the coarsest level.

During the synthesis at each pyramid level l a nearest-neighbour field NNF_l is constructed. It stores frame number i, centroid $[x,y]$, and rotation θ of currently best matching source patches s for each target patch t centered at pixel $[\hat{x},\hat{y}]$, i.e., $(i,x,y,\theta) = NNF_l(j,\hat{x},\hat{y})$. The advantage of NNF is that it can be upsampled when transferring the solution from a coarse level $l+1$ to a finer level l [TFF*20]. However, since in our scenario source patches can be rotated, each upscaled coordinate

$$NNF_l^{\uparrow}(j,\hat{x},\hat{y}) = (i,x^{\uparrow},y^{\uparrow},\theta)$$

has to have an additional offset given by the Jacobian of the underlying transformation, i.e., backward rotation of the corresponding patch (see Fig. 6):

$$(x^{\uparrow}, y^{\uparrow}) = 2 \cdot (x', y') + \mathbf{c} + R_{-\theta}((\hat{x} \bmod 2, \hat{y} \bmod 2) - \mathbf{c}),$$

where $\mathbf{c} = (\frac{1}{2}, \frac{1}{2})$, R_θ is the operator of rotation by θ radians, and (x', y', θ) can be extracted from the coarse level NNF_{l+1} as follows:

$$(i, x', y', \theta) = NNF_{l+1}(j, \lfloor \hat{x}/2 \rfloor, \lfloor \hat{y}/2 \rfloor).$$

Since the optimization of E is performed sequentially we can further speed up the convergence by initializing the NNF of the current frame using the values from the previous frame with shifted frame indices

$$NNF_{prev}(j, \hat{x}, \hat{y}) = (i + \Delta i, x, y, \theta)$$

where $(i, x, y, \theta) = NNF(j-1, \hat{x}, \hat{y})$. On the level l the initial NNF_l is obtained by merging two NNFs: (1) the upscaled $NNF_l^{\uparrow}$ and (2) a downscaled $NNF_{prev}^{\downarrow}$. The NNF downscaling process consists of the following steps done for each target pixel $[j, \hat{x}, \hat{y}]$:

1. nearest-neighbour coordinates are gathered from a square window from NNF_{prev} of width 2^l with the top-left corner positioned at $2^l(\hat{x}, \hat{y})$ (see Fig. 7a),
2. each corresponding patch coordinate (i, x, y) is transformed by the inverse of the upscaling transformation (see Fig. 7b),
3. the mode of transformed patch coordinates (i, x, y) is assigned as the new value of $NNF_{prev}^{\downarrow}(j, \hat{x}, \hat{y})$ and the prescribed rotation θ is stored to $T^{rot}(j, \hat{x}, \hat{y})$ (see Fig. 7c).

These two mappings are then merged on a per-pixel basis based on which of the two mappings has a smaller error.

Thanks to these extensions, the initialization of the NNF needs to be done only in the coarsest level of the first frame. In the following frames and pyramid levels, we start the optimization using good estimates from previous NNFs (through $NNF_{prev}^{\downarrow}$ and $NNF_l^{\uparrow}$). This enables us to bring a significant performance gain over previous sequential solvers.

We provide pseudocode of the entire algorithm in Appendix B.

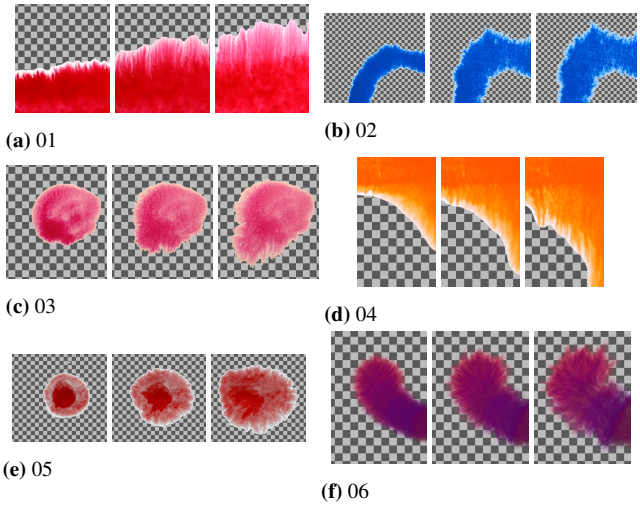
4. Results

We implemented our approach using C++ and set all tunable parameters to values presented in Table 1.

During the optimization of E we accelerate the retrieval of nearest-neighbour patch using generalized PatchMatch algorithm [BSGF10]. To further decrease computational overhead, we omit search for rotations θ and instead for each randomly sampled triplet (i, x, y) we test only those rotations that minimize D^{dir}. This allows us to reduce the search space to only three dimensions and thus substantially lower the number of random samples required to get satisfactory results. Besides that we also take advantage of the nearest-neighbour field upsampling and reuse from previous frames (see Section 3.6) which further lower the computation overhead. With all those optimizations on a quad core 3 GHz CPU our method is more than an order of magnitude faster when compared to the computational overhead of the LazyFluids algorithm [JFA*15].

Table 1: *Settings of all tunable parameters used to generate results presented in Fig. 10 (d is the maximal dimension of the target animation).*

parameter	description	value
n	patch size	5
b	boundary region width	30 px
r	cuboid size	5 frames
w^{bound}	boundary term weight	4
w^{ext}	motion extent term weight	0.25
w^{dir}	directional alignment weight	1
w^{temp}	temporal coherence weight	4
w^{uni}	uniformity weight	1
m_l	temporal modulation lower threshold	$d \cdot 0.01$
m_u	temporal modulation upper threshold	$d \cdot 0.1$



(a) 01 **(b)** 02 **(c)** 03 **(d)** 04 **(e)** 05 **(f)** 06

Figure 8: *A selection of source exemplars used for evaluation. The checkerboard pattern indicates areas outside the mask. Exemplars* 01, 02, 03, 04, 05 *are natural and* 06 *is synthetically generated using simulation.*

To validate our method we recorded five natural style exemplars, and one synthetic one using fluid simulation [CAS*97] (see Fig. 8). For target animations we prepared six different sequences manifesting various kinds of movement (see Fig. 9). Results for various combinations of styles and target animations are available in our supplementary video and are depicted in Fig. 10. The average computational overhead for individual target sequences is presented in Table 2.

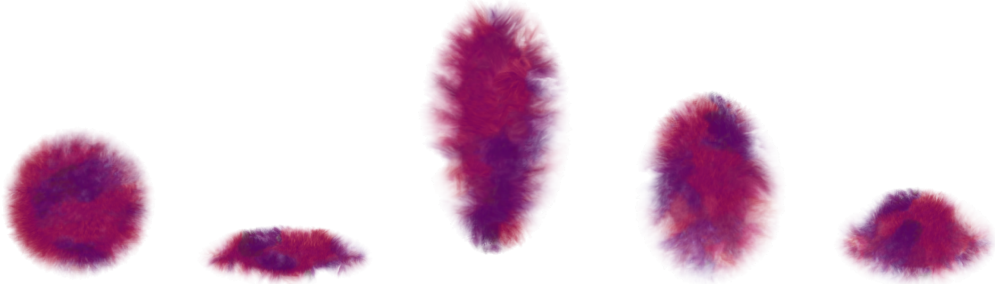
The results demonstrate that our method handles complex shapes and can transfer fine detailed texture while maintaining the appearance and dynamics of the original style exemplar. Despite the interpolation of source pixels' colors due to arbitrary rotation and blending of the rotated patches, the output does not significantly suffer from a detail loss or wash-out.

(a) *The* genie *animation stylized using natural exemplar* 01.

(b) *The* horse *animation stylized using natural exemplar* 02.

(c) *The* sunflower *animation stylized using natural exemplar* 03.

(d) *The* waving *animation stylized using the exemplar* 05. *Segments were generated separately, colorized, and composited together with a solid color background.*



(e) *The* jump *animation stylized using synthetic exemplar* 06.

Figure 10: *Previews of the results*

Table 2: *Average timings for source-target combinations presented in Fig. 10.*

resulting sequence source + target	source sequence width × height × #frames	target sequence width × height × #frames	target sequence #pixels (inside mask)	avg. time per frame seconds
01 + *genie* (Fig. 10a)	$400 \times 120 \times 30$	$900 \times 900 \times 400$	123×10^3	3.48
02 + *horse* (Fig. 10b)	$400 \times 222 \times 20$	$1000 \times 800 \times 400$	111×10^3	2.89
03 + *sunflower* (Fig. 10c)	$200 \times 200 \times 20$	$500 \times 600 \times 200$	60×10^3	1.33
05 + *waving* (Fig. 10d)	$160 \times 149 \times 20$	$640 \times 740 \times 250$	80×10^3	2.62
06 + *jump* (Fig. 10e)	$225 \times 94 \times 20$	$700 \times 300 \times 225$	12×10^3	0.55

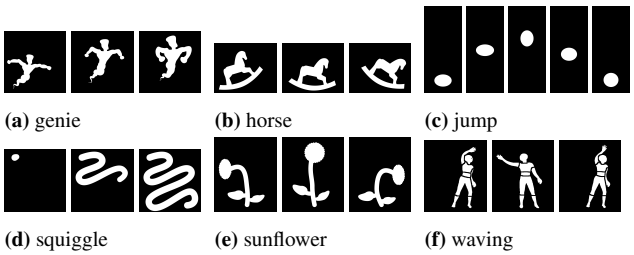


Figure 9: *A selection of target animation sequences used for evaluation.*

To demonstrate the effect of individual terms in our energy function E we performed an ablation study (see our supplementary video) where we selectively set the weight of each individual term to zero. It is visible that omitting the boundary term D^{bound} leads to a loss of natural transition between the artistic media and the canvas. When the motion direction alignment D^{dir} is missing, the material is perceived as moving in random directions which are not in line with the prescribed target flow except at boundaries where the D^{bound} slightly reinforces correct direction. When both terms are set to zero, the resulting flow becomes completely random. Ignoring motion extent term D^{ext} leads to a selection of patches which do not provide a sufficient movement in the subsequent frames therefore unnatural-looking results are produced, e.g., moving regions are stylized with the texture of stationary ones and vice versa. Disabling the patch occurrence measure Ω leads to visible wash-out artifacts since the variety of source patches becomes significantly reduced. Without D^{temp} the output is not coherent in time and by switching off D^{tex} the texture details become slightly deteriorated. The reason why the output looks reasonable even when D^{tex} is not active is the fact that texture coherence is also jointly enforced by D^{temp}. When both terms (D^{tex} and D^{temp}) are disabled, the drop in quality becomes more noticeable.

We compared our technique to the LazyFluids algorithm [JFA*15] which was originally developed for a different application scenario (appearance transfer to fluid simulations), however, it still represents a closest previous state-of-the-art method that can be applied in our setting. We use our source binary masks as alpha channels in their RGBA input and the target flow field is constructed using the method described in Appendix A. See our supplementary video for a comparison, where it is visible that both methods successfully retain the appearance of the style exemplar, but when seen in motion it is apparent that the texture in the static area is gradually warped by the LazyFluids algorithm and thus the motion characteristics of the original exemplar sequence are not preserved well. Also, LazyFluids tends to superimpose the exemplar motion on top of the target motion, resulting in a composition typically not in line with the prescribed direction, which may lead to drifting artifacts. In contrast our method better preserves the stationary components and also more faithfully resembles the dynamic properties of the exemplar artistic media.

In our supplementary video, we also provide a comparison to EbSynth [JvST*19]—an example-based method that represents the traditional approach to video stylization. In this technique style exemplar follows the motion in the target video precisely while the temporal coherence is enforced explicitly. Although the purpose of EbSynth differs from our scenario, seeing its results side-by-side with our technique helps to understand the core idea behind fluidymation. The aim is not to keep the texture attached to the target moving object but instead convey the motion as if the paint diffuses over the canvas.

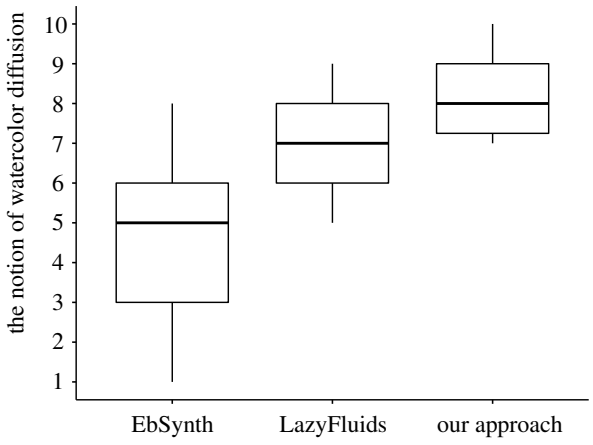


Figure 11: *User study results. On a likert scale 18 participants where asked to what extent our approach and two previous techniques (LazyFludis [JFA*15] and EbSynth [JvST*19]) convey the notion of watercolor diffusion (1 is "not at all" and 10 is "absolutely"). Our approach was almost consistently evaluated to better preserve the desired motion dynamics.*

To provide a quantitative evaluation we conducted a user study with 18 participants (10 men and 8 women) out of which 7 were professional artists and 11 casual observers. We presented them with sequences produced by our approach and also two previous

methods (LazyFluids [JFA*15] and EbSynth [JvST*19]) and asked them on a likert scale to what extent they think each particular technique conveys the notion of watercolor diffusion (1 is "not at all" and 10 is "absolutely"). Results of the user study are presented in Fig. 11, which shows that users perceive our approach as delivering results that are closest to the desired motion dynamics.

5. Limitations and Future Work

Although the proposed method provides a viable solution to the Fluidymation scenario, there are still some limitations that could inspire future improvements.

Since the synthesis algorithm searches for patches over the entire source sequence, we need to upload it into memory. This limits the dimensions and length of the source sequence that can be used to stylize the output in a single run. In practice such a limitation can be bypassed by uploading only a fraction of the source frames in a sliding window that can be shifted in time.

A scenario in which our method can encounter difficulties is when new material is added on the canvas during the animation (see, e.g., results with the *squiggle* animation in Fig. 9d). In this case the area under the imaginary brush appears like it has already received the paint (see Fig. 12a) which may not be perceived as a realistic behavior. This problem can be alleviated by generating the output in a reverse order (see Fig. 12b). This, however, requires an additional supervision which we plan to automatize in future work.

When the style exemplar contains only a small area with substantial motion (see, e.g., Fig. 8c), the resulting sequence may contain slight flicker due to lack of sufficiently dynamic content at the area of moving edges. Also, the areas where the target motion is directed inward may contain more artifacts than those with outward motion. This is caused by the fact that the temporal coherence is maintained only in the forward direction. In future work it would be beneficial to consider also bidirectional optimization in the spirit of [BCK*13].

A challenge for our technique could be generalization to subtle geometric details (see, e.g., the lower tip of the genie in Fig. 9a) for which there are no similar counterparts available in the style exemplar. In those cases spurious shape fragmentation may occur in the resulting stylized sequence (see Fig. 10a).

Our simple flow field construction algorithm (see Appendix A) was designed to capture diffusive motions where the dominant flow component is usually perpendicular to the mask boundary. Due to this design some type of movements such as rotations may not be captured correctly. This drawback could manifest in the results as if the material is moving diagonally to the mask boundary. In future we envision to employ more robust flow field construction algorithms (e.g., [OF03, NBM05]) which would capture also these additional details.

6. Conclusion

We have presented an approach to the example-based stylization of animations that retains the appearance and the dynamic properties of the original hand-painted style exemplar. Thanks to this property, we can avoid the temporal incoherence issue typical for hand-colored animations while at the same time overcoming the unnatural stiffness and dissonance of previous stylization techniques that enforce temporal coherence explicitly. We believe our approach can inspire artists to bring new life to their animations, giving them a more natural hand-painted look.

(a) *Normal result* **(b)** *Target animation reversed*

Figure 12: *The* squiggle *animation displays an unrealistic appearance of a material being added onto a canvas while a more realistic result can be obtained when the target animation is reversed.*

Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback. We are also grateful to Ondřej Jamriška for his help with initial development and paper writing as well as Pavla Sýkorová for capturing style exemplars. This research was supported by Google, the Fulbright Commission in the Czech Republic, the Grant Agency of the Czech Technical University in Prague, grant No. SGS19/179/OHK3/3T/13 (Research of Modern Computer Graphics Methods), and by the Research Center for Informatics, grant No. CZ.02.1.01/0.0/0.0/16_019/0000765.

References

[BBRF14] Browning M., Barnes C., Ritter S., Finkelstein A.: Stylized keyframe animation of fluid simulations. In *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering* (2014), pp. 63–70.

[BCK*13] Bénard P., Cole F., Kass M., Mordatch I., Hegarty J., Senn M. S., Fleischer K., Pesare D., Breeden K.: Stylizing animation by example. *ACM Transactions on Graphics 32*, 4 (2013), 119.

[BKTS06] Bousseau A., Kaplan M., Thollot J., Sillion F. X.: Interactive watercolor rendering with temporal coherence and abstraction. In *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering* (2006), pp. 141–149.

[BLV*10] Bénard P., Lagae A., Vangorp P., Lefebvre S., Dretakis G., Thollot J.: A dynamic noise primitive for coherent stylization. *Computer Graphics Forum 29*, 4 (2010), 1497–1506.

[BNTS07] Bousseau A., Neyret F., Thollot J., Salesin D.: Video watercolorization using bidirectional texture advection. *ACM Transactions on Graphics 26*, 3 (2007), 104.

[BSGF10] Barnes C., Shechtman E., Goldman D. B., Finkelstein A.: The generalized PatchMatch correspondence algorithm. In *Proceedings of European Conference on Computer Vision* (2010), pp. 29–43.

[BSM*07] Breslav S., Szerszen K., Markosian L., Barla P., Thollot J.: Dynamic 2D patterns for shading 3D scenes. *ACM Transactions on Graphics 26*, 3 (2007), 20.

[BWL04] BAXTER W., WENDT J., LIN M. C.: IMPaSTo: A realistic, interactive model for paint. In *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering* (2004), pp. 45–56.

[CAS*97] CURTIS C. J., ANDERSON S. E., SEIMS J. E., FLEISCHER K. W., SALESIN D. H.: Computer-generated watercolor. In *SIGGRAPH Conference Proceedings* (1997), pp. 421–430.

[FCC*19] FUTSCHIK D., CHAI M., CAO C., MA C., STOLIAR A., KOROLEV S., TULYAKOV S., KUČERA M., SÝKORA D.: Real-time patch-based stylization of portraits using generative adversarial network. In *Proceedings of the ACM/EG Expressive Symposium* (2019), pp. 33–42.

[FH12] FELZENSZWALB P. F., HUTTENLOCHER D. P.: Distance transforms of sampled functions. *Theory of Computing 8*, 1 (2012), 415–428.

[FJL*16] FIŠER J., JAMRIŠKA O., LUKÁČ M., SHECHTMAN E., ASENTE P., LU J., SÝKORA D.: StyLit: Illumination-guided example-based stylization of 3D renderings. *ACM Transactions on Graphics 35*, 4 (2016), 92.

[FJS*17] FIŠER J., JAMRIŠKA O., SIMONS D., SHECHTMAN E., LU J., ASENTE P., LUKÁČ M., SÝKORA D.: Example-based synthesis of stylized facial animations. *ACM Transactions on Graphics 36*, 4 (2017), 155.

[FLJ*14] FIŠER J., LUKÁČ M., JAMRIŠKA O., ČADÍK M., GINGOLD Y., ASENTE P., SÝKORA D.: Color Me Noisy: Example-based rendering of hand-colored animations with temporal noise control. *Computer Graphics Forum 33*, 4 (2014), 1–10.

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2414–2423.

[Hae90] HAEBERLI P.: Paint by numbers: Abstract image representations. *SIGGRAPH Computer Graphics 24*, 4 (1990), 207–214.

[HE04] HAYS J., ESSA I. A.: Image and video based painterly animation. In *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering* (2004), pp. 113–120.

[Her98] HERTZMANN A.: Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH Conference Proceedings* (1998), pp. 453–460.

[Her01] HERTZMANN A.: Paint by relaxation. In *Proceedings of Computer Graphics International* (2001), pp. 47–54.

[HJO*01] HERTZMANN A., JACOBS C. E., OLIVER N., CURLESS B., SALESIN D. H.: Image analogies. In *SIGGRAPH Conference Proceedings* (2001), pp. 327–340.

[HLFR07] HAEVRE W. V., LAERHOVEN T. V., FIORE F. D., REETH F. V.: From Dust Till Drawn: A real-time bidirectional pastel simulation. *The Visual Computer 23*, 9–11 (2007), 925–934.

[IZZE17] ISOLA P., ZHU J.-Y., ZHOU T., EFROS A. A.: Image-to-image translation with conditional adversarial networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 5967–5976.

[JFA*15] JAMRIŠKA O., FIŠER J., ASENTE P., LU J., SHECHTMAN E., SÝKORA D.: LazyFluids: Appearance transfer for fluid animations. *ACM Transactions on Graphics 34*, 4 (2015), 92.

[Joh02] JOHNSTON S. F.: Lumo: Illumination for cel animation. In *Proceedings of International Symposium on Non-Photorealistic Animation and Rendering* (2002), pp. 45–52.

[JvST*19] JAMRIŠKA O., ŠÁRKA SOCHOROVÁ, TEXLER O., LUKÁČ M., FIŠER J., LU J., SHECHTMAN E., SÝKORA D.: Stylizing video by example. *ACM Transactions on Graphics 38*, 4 (2019), 107.

[KAGS19] KIM B., AZEVEDO V. C., GROSS M., SOLENTHALER B.: Transport-based neural style transfer for smoke simulations. *ACM Transactions on Graphics 38*, 6 (2019), 188.

[KAGS20] KIM B., AZEVEDO V. C., GROSS M., SOLENTHALER B.: Lagrangian neural style transfer for fluids. *ACM Transactions on Graphics 39*, 4 (2020), 52.

[KEBK05] KWATRA V., ESSA I. A., BOBICK A. F., KWATRA N.: Texture optimization for example-based synthesis. *ACM Transactions on Graphics 24*, 3 (2005), 795–802.

[KNL*15] KASPAR A., NEUBERT B., LISCHINSKI D., PAULY M., KOPF J.: Self tuning texture optimization. *Computer Graphics Forum 34*, 2 (2015), 349–360.

[KSHTB*03] KROLAK-SALMON P., HÉNAFF M.-A., TALLON-BAUDRY C., YVERT B., GUÉNOT M., VIGHETTO A., MAUGUIERE F., BERTRAND O.: Human lateral geniculate nucleus and visual cortex respond to screen flicker. *Annals of Neurology 53*, 1 (2003), 73–80.

[KSM*19] KOTOVENKO D., SANAKOYEU A., MA P., LANG S., OMMER B.: A content transformation block for image style transfer. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 10032–10041.

[LBDF13] LU J., BARNES C., DIVERDI S., FINKELSTEIN A.: RealBrush: painting with examples of physical media. *ACM Transactions on Graphics 32*, 4 (2013), 117.

[Mei96] MEIER B. J.: Painterly rendering for animation. In *SIGGRAPH Conference Proceedings* (1996), pp. 477–484.

[MSS*18] MONTESDEOCA S. E., SEAH H. S., SEMMO A., BÉNARD P., VERGNE R., THOLLOT J., BENVENUTI D.: Mnpr: A framework for real-time expressive non-photorealistic rendering of 3d computer graphics. In *Proceedings of The Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering* (2018), p. 11.

[NBM05] NILSSON O., BREEN D. E., MUSETH K.: Surface reconstruction via contour metamorphosis: An eulerian approach with lagrangian particle tracking. In *IEEE Visualization* (2005), pp. 407–414.

[OF03] OSHER S., FEDKIW R. P.: *Level set methods and dynamic implicit surfaces*, vol. 153. 2003.

[RDB18] RUDER M., DOSOVITSKIY A., BROX T.: Artistic style transfer for videos and spherical images. *International Journal of Computer Vision 126*, 11 (2018), 1199–1219.

[SED16] SELIM A., ELGHARIB M., DOYLE L.: Painting style transfer for head portraits using convolutional neural networks. *ACM Transactions on Graphics 35*, 4 (2016), 129.

[SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014).

[TFF*20] TEXLER O., FUTSCHIK D., FIŠER J., LUKÁČ M., LU J., SHECHTMAN E., SÝKORA D.: Arbitrary style transfer using neurally-guided patch-based synthesis. *Computers & Graphics 87* (2020), 62–71.

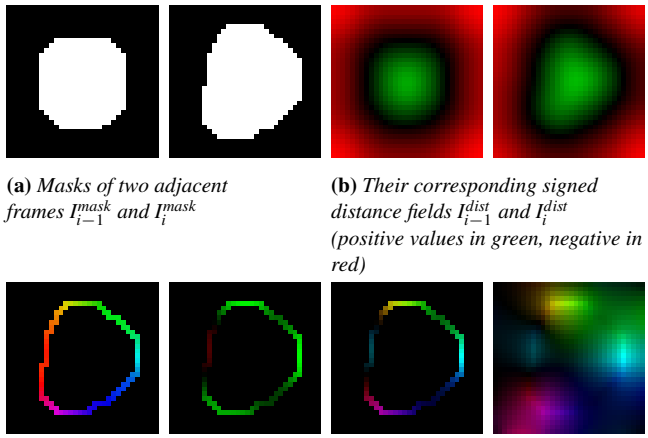
[Wel19] WELCHMAN H.: *Loving Vincent: The Journey*. 2019.

[WSI07] WEXLER Y., SHECHTMAN E., IRANI M.: Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 3 (2007), 463–476.

Appendix A: Flow field construction

When either S^{flow} or T^{flow} is not provided as an input to our algorithm, we can approximate them using the motion at the boundaries of masks S^{mask} and T^{mask}. As the procedure is the same for the source S as well as the target T sequence, we will denote both as I.

We assume a direction of a flow field I_i^{flow} at the boundary of I_i^{mask} is perpendicular to its tangent, and therefore parallel to the gradient of a distance field I_i^{dist} computed from the mask's boundary [FH12] (see Fig. 13c). The magnitude of I_i^{flow} at the mask boundary (see Fig. 13d) can then be estimated as a difference of distance fields of two consecutive frames (see Fig. 13d) giving us



(a) *Masks of two adjacent frames I_{i-1}^{mask} and I_i^{mask}*

(b) *Their corresponding signed distance fields I_{i-1}^{dist} and I_i^{dist} (positive values in green, negative in red)*

(c) *The gradient of the second distance field ∇I_i^{dist}, i.e., the flow direction (hue is direction, intensity is magnitude)*

(d) *The difference of the distance fields $I_i^{dist} - I_{i-1}^{dist}$ at the second mask's border, i.e., the amount of flow*

(e) *The difference and gradient multiplied, i.e., the final flow at the border*

(f) *The final flow field I_i^{flow} obtained by diffusing the border pixels' values*

Figure 13: *Flow field construction in 2D*

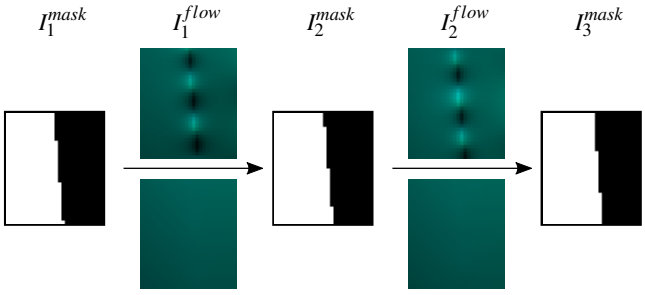


Figure 14: *Enlarged part of three successive frames' masks with corresponding flow fields between them, without (top) and with (bottom) the distance field temporal smoothing applied.*

a complete estimate of a flow field at the boundary of mask I_i^{mask} (see Fig. 13e):

$$I_i^{flow} = \left(I_i^{dist} - I_{i-1}^{dist}\right) \frac{\nabla I_i^{dist}}{\left|\nabla I_i^{dist}\right|}.$$

To compute flow field vectors for the pixels in the interior of I_i^{mask} we use diffusion of the border values as in Johnston et al. [Joh02] (see Fig. 13f).

To improve smoothness in the temporal domain and enable sub-pixel movements which cannot be captured by binary masks, we convolve the estimated sequence I^{flow} with a temporal Gaussian filter with radius of 3 frames (the difference between flow fields with and without the temporal smoothing can be seen in Fig. 14).

Appendix B: Algorithm overview

An overview of multi-scale optimization scheme for minimizing E which includes also our special handling of nearest-neighbour fields is presented in Algorithm 1. The operation SHIFTNNF is described in Section 3.4 and operations UPSCALENNF, DOWNSCALENNF, and MERGENNF are specified in Section 3.6. Inputs S^{guides} and T^{guides} denote all additional guiding channels and their weights (i.e., *bound*, *int*, *flow*, *rot*, *dist*, *temp*, *ext+*, *ext−*, and *uni*).

Algorithm 1 Multi-scale optimization

$NNF_{prev} \leftarrow$ empty mapping
$T_i \leftarrow$ empty image
for each level l **do**
  if l is the coarsest level **then**
    if first frame **then**
      $NNF_{init} \leftarrow$ random mapping
    else
      $NNF_{init} \leftarrow$ DOWNSCALENNF(NNF_{prev})
    end if
  else
    if first frame **then**
      $NNF_{init} \leftarrow$ UPSCALENNF(NNF_{l+1})
    else
      $NNF_l^{\uparrow} \leftarrow$ UPSCALENNF(NNF_{l+1})
      $NNF_{prev}^{\downarrow} \leftarrow$ DOWNSCALENNF(NNF_{prev})
      $NNF_{init} \leftarrow$ MERGENNF($NNF_l^{\uparrow}, NNF_{prev}^{\downarrow}$)
    end if
  end if
  $NNF, T_i^{rgb} \leftarrow$ OPTIMIZE($S^{rgb}, S^{mask}, S^{guides}, T_i^{mask}, T_i^{guides}, NNF_{init}$)
end for
output T_i^{rgb}
$NNF_{prev} \leftarrow$ SHIFTNNF(NNF)

The central EM-like texture optimization loop OPTIMIZE inspired by [KEBK05, WSI07] is presented in Algorithm 2. Here the operation FINDNNF uses patch similarity D to retrieve nearest neighbour patches whereas VOTENNF performs voting that averages color of all co-located pixels within intersecting patches (as in [KEBK05]).

Algorithm 2 Texture Optimization

function OPTIMIZE($S^{rgb}, S^{mask}, S^{guides}, T_i^{mask}, T_i^{guides}, NNF_{init}$)
  $NNF \leftarrow NNF_{init}$
  $T_i^{rgb} \leftarrow$ VOTE(S^{rgb}, NNF)
  for $j \leftarrow 1 \ldots opt\_iters$ **do**
    $NNF \leftarrow$ FINDNNF($S^{rgb}, S^{mask}, S^{guides}, T_i^{rgb}, T_i^{mask}, T_i^{guides}$)
    $T_i^{rgb} \leftarrow$ VOTE(S^{rgb}, NNF)
  end for
  return NNF, T_i^{rgb}
end function