Fluidymation: Stylizing Animations Using Natural Dynamics of Artistic Media

A. Platkević¹, C. Curtis², D. Sýkora¹

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

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 [HEO4] or a 2D projection of the movement of 3D objects [Me96]. 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 [Me96], 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,WSL07] to produce the final stylized image. Such an approach can also be extended into the video domain [BCK'13,FLJ'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 exemplar, 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):

- \( S_{\text{rgb}} \) — a sequence of \( I_S \) images serving as a style exemplar, captured by a camera perpendicular to a canvas,
- \( S_{\text{mask}} \) — a sequence of \( I_S \) binary masks denoting the presence of artistic media in the exemplar image,
- \( T_{\text{mask}} \) — a sequence of \( I_T \) 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: \( S_{\text{flow}} \) and \( T_{\text{flow}} \). 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 \( S_{\text{mask}} \) and \( T_{\text{mask}} \) (see Appendix A).
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 $\theta$ prescribed in an orientation field $T^{rot}$. For every pixel $q = (x, y)$ in the target sequence $T^{rgb}$, we seek a suitable patch $s$ centered at a pixel $p = (i, x, y)$ in the source sequence $S^{rgb}$ and rotated by $\theta$ stored in $T^{rot}$. The recently computed nearest-neighbour field (NNF) from the previous frame $\text{NNF}_{\text{prev}}$ is used in the output frame $T^{rgb}$.

In contrast to the method of Jamriška et al. [JFA15] our new formulation provides two important advantages: (1) the 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:

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

where the patch similarity function $D$ is defined as follows:

$$D(p, q) = \sum_{p' \in S; q' \in S'} T^{ex}(p', q') + w^{\text{bound}} T^{\text{bound}}(p', q') + w^{\text{dir}} T^{\text{dir}}(p', q') + w^{\text{temp}} T^{\text{temp}}(p', q') + w^{\text{shape}} T^{\text{shape}}(p', q').$$

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 patches covered by 2D patches $s$ and $t$ of size $n \times n$. $T^{ex}$ measures the consistency of texture and $T^{\text{bound}}$ maintains alignment of mask boundaries, i.e., during the optimization those two terms mostly affect the $p'$ spatial coordinates $[x, y]$. The term $T^{\text{dir}}$ upholds the consistency of motion direction controlled by the parameter $\theta$, $T^{\text{shape}}$ encourages the selection of patches that have an appropriate extent of motion, i.e., influences the temporal coordinate $i$ of the pixel $p$, and $T^{\text{temp}}$ keeps the output coherent in time. Finally, the uniformity term $\Omega$ helps to avoid the overuse of particular source patches that may cause so-called wash-out artifacts [JFA15], i.e., a lack of visual variety seen in the original source exemplar. The influence of each individual term $D^*$ is balanced relatively to $D^{\text{ex}}$ using a weighting factor $w^*$. All terms $D^{\text{ex}}, D^{\text{bound}}, D^{\text{dir}}, D^{\text{temp}}, D^{\text{shape}}$, and $\Omega$ are described in more detail in the following sections.

### 3.1. Texture consistency and boundary effects

The computation of $D^{\text{ex}}$ and $D^{\text{bound}}$ is similar to that of Jamriška et al. [JFA15] except for the rotation $T^{\text{rot}}(q)$ of the corresponding source patch $p$ which needs to be taken into account when computing those two terms. The term $D^{\text{ex}}$, responsible for local visual similarity of the generated texture to the source one, is computed as follows:

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

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

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

### 3.2. Motion orientation alignment

The term $D^{\text{dir}}$ 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 $D^{\text{dir}}$ as follows:

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

where $S^{\text{flow}}$ and $T^{\text{flow}}$ are the forward velocity fields at $p$ and $q$, respectively. The term $D^{\text{temp}}$ encourages the selection of patches that have an appropriate extent of motion, i.e., influences the temporal coordinate $i$ of the pixel $p$, and $D^{\text{temp}}$ keeps the output coherent in time. Finally, the uniformity term $\Omega$ helps to avoid the overuse of particular source patches that may cause so-called wash-out artifacts [JFA15], i.e., a lack of visual variety seen in the original source exemplar.
where $S^\text{flow}_0(p')$ and $T^\text{flow}_0(q')$ are the respective directions of the flow fields at pixels $p'$ and $q'$ in radians. An orientation $T^\text{ext}(q')$ 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 superimosed 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 $H^\text{flow}$ (see Fig. 4b) to produce two guidance channels: $I^\text{ext}$ and $I^\text{dir}$ (where $I$ denotes either $S$ or $T$).

The forward channel $I^\text{dir}$ is constructed by accumulating the amount of motion at each pixel since the start of the sequence, i.e.,

$$I^\text{dir}_i(x, y) = \|\text{acc}_+ (i, x, y)\|,$$

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

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

In addition to accumulation we zero $\text{acc}_-$ at pixels that are outside the mask $I^\text{mask}$ to make sure the accumulation is restarted at coordinates where the material appears repeatedly. An example of $I^\text{dir}$ 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.,

$$I^\text{ext}_i(x, y) = \|\text{acc}_- (i, x, y)\|,$$

where

$$\text{acc}_- (i, x, y) = \begin{cases} \text{acc}_- (i - 1, x, y) + H^\text{flow}(i, x, y), & \text{if } i < t, \\ (0, 0), & \text{otherwise}. \end{cases}$$

Similarly to $I^\text{dir}$ we zero the accumulator $\text{acc}_-(i, x, y)$ for pixels $[x, y]$ that are outside the mask, i.e., $I^\text{mask}(i, x, y) = 0$.

Using these two additional guiding channels $D^\text{dir}$ is computed as follows:

$$D^\text{dir}(p', q') = \|S^\text{ext}_+(p') - T^\text{ext}_+(q')\|^2 + \|S^\text{ext}_-(p') - T^\text{ext}_-(q')\|^2.$$

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 $T^\text{ext}$ and $D^\text{ext}$ are swapped before being subtracted from $S^\text{ext}$ and $S^\text{ext}$.

### 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 $\Delta t$ 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 $\Delta t$, i.e., the number of frames the index is shifted to meet the amount of motion in the target sequence.
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 $\Delta i$ under this assumption we accumulate the amount of motion $S_{ext}$ 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 $q$:

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

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:

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

Then the optimal time shift $\Delta i^*$ can be retrieved as follows:

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

When the optimal $\Delta i^*$ is known the location of the corresponding patch $s^*$ centered at pixel $p^* = (i+\Delta i^*, x, y)$ (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 $S^\text{ext}_{i}^-$, we use future motion amount $S^\text{temp}_{i}$ (see Section 3.3).

As soon as all corresponding shifted source patches are known, we can produce a target temporal guide $T^\text{temp}$ 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 $S^\text{temp}_{i}$ and the final term $D^\text{temp}$ is computed accordingly to Fiser et al. [FJS*17]:

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

Since the temporal guide is meaningful only in regions where the previous mask overlaps the current one, the weight $w^\text{temp}$ is set to zero at locations where there is no overlap. In the first frame, $w^\text{temp}$ is set to zero as no previous frame is available.
When the movement in the target sequence is much stronger than in the exemplar, $s^{\text{dist}}(k, \hat{x}, \hat{y}) = | \hat{b} - \hat{a} |$ may not be sufficiently large to match that of $T^{\text{dist}}(k, \hat{x}, \hat{y})$, for example when $i + \Delta^*$ 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 $\omega^{\text{temp}}$ based on the difference of signed distance fields of the source and target masks as follows:

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

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

$$s(v, m_1, m_2) = \begin{cases} 0, & \text{if} \ v \leq m_1, \\ 1, & \text{if} \ v \geq m_2, \\ \left(3v^2 - 2v^3\right), & \text{otherwise}, \end{cases}$$

where $v' = (v - m_1)/(m_2 - m_1)$ and $m_1$ and $m_2$ 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 $\Omega^*$ 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 $\Omega^*$ into 3D and perform the occurrence accumulation with respect to the orientation of individual patches. The formula for our 3D occurrence map becomes:

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

where $(k, \hat{x}, \hat{y})$ are the coordinates of a nearest-neighbour patch $q$, $\Theta = T^{\text{int}}(k, \hat{x}, \hat{y})$, 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 $\Theta$ radians in the x,y-plane. The configurable parameter $r$ sets the temporal dimension of the penalised neighbourhood. This occurrence map is computed for each frame separately. Similarly to Jamriška et al. [JFA’15], we treat the boundary segments $S^{\text{bound}}$ and $q^{\text{bound}}$ and interior segments $S^{\text{int}}$ and $T^{\text{int}}$ separately, obtaining the following formula for $\omega^*$:

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

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

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

### 3.6. Optimization

To minimize $E$ we use a multi-resolution EM-like optimization scheme of Wexler et al. [WSI’07]. The number of resolution pyramid levels is set to $\lfloor \log_2 d_{\text{min}}/n \rfloor$, where $d_{\text{min}}$ 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 $\Theta$ of currently best matching source patches $s$ for each target patch $t$ centered at pixel $[\hat{x}, \hat{y}]$, i.e., $(i, x, y, \Theta) = \text{NNF}_l(j, \hat{x}, \hat{y})$. The advantage of NNF is that it can be upsamped when transferring the solution from a coarse level $l + 1$ to a finer level $l$ [TFF20]. However, since in our scenario source patches can be rotated, eachUpscaled coordinate $NNF^1_l(j, \hat{x}, \hat{y}) = (i, x^1, y^1, \Theta)$.
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):

\[(x', y') = 2 \cdot (x, y') + e + R_{\theta}(x \mod 2, y \mod 2) - e,\]

where \(e = (\frac{1}{2}, \frac{1}{2})\), \(R_{\theta}\) is the operator of rotation by \(\theta\) radians, and \((x', y', \theta)\) can be extracted from the coarse level \(\text{NNF}_{i+1}\) as follows:

\[(i, x', y', \theta) = \text{NNF}_{i+1}(j, [\hat{x}/2], [\hat{y}/2]).\]

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

\[\text{NNF}_{\text{prev}}(j, \hat{x}, \hat{y}) = (i + \Delta i, x, y, \theta)\]

where \((i, x, y, \theta) = \text{NNF}(j - 1, \hat{x}, \hat{y})\). On the level \(l\) the initial \(\text{NNF}_l\) is obtained by merging two \(\text{NNFs}\): (1) the upscaled \(\text{NNF}_l^u\) and (2) a downscaled \(\text{NNF}_l^d\). The \(\text{NNF}\) downsampling 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 \(\text{NNF}_{\text{prev}}\) 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 upsampling transformation (see Fig. 7b).
3. the mode of transformed patch coordinates \((i, x, y)\) is assigned as the new value of \(\text{NNF}_{\text{prev}}(j, \hat{x}, \hat{y})\) and the prescribed rotation \(\theta\) is stored to \(T_{\theta}^m(j, \hat{x}, \hat{y})\) (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 \(\text{NNF}\) 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 \(\text{NNFs}\) (through \(\text{NNF}_{\text{prev}}\) and \(\text{NNF}_l^d\)). 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 \(\theta\) and instead for each randomly sampled triplet \((i, x, y)\) we test only those rotations that minimize \(D_{\theta}^{\text{PPM}}\). 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].

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.

---

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

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>patch size</td>
<td>5</td>
</tr>
<tr>
<td>(b)</td>
<td>boundary region width</td>
<td>30 px</td>
</tr>
<tr>
<td>(r)</td>
<td>cuboid size</td>
<td>5 frames</td>
</tr>
<tr>
<td>(w_{\text{bound}})</td>
<td>boundary term weight</td>
<td>4</td>
</tr>
<tr>
<td>(w_{\text{motion}})</td>
<td>motion extent term weight</td>
<td>0.25</td>
</tr>
<tr>
<td>(w_{\text{dir}})</td>
<td>directional alignment weight</td>
<td>1</td>
</tr>
<tr>
<td>(w_{\text{temp}})</td>
<td>temporal coherence weight</td>
<td>4</td>
</tr>
<tr>
<td>(w_{\text{uni}})</td>
<td>uniformity weight</td>
<td>1</td>
</tr>
<tr>
<td>(m_l)</td>
<td>temporal modulation lower threshold</td>
<td>(d \cdot 0.01)</td>
</tr>
<tr>
<td>(m_u)</td>
<td>temporal modulation upper threshold</td>
<td>(d \cdot 0.1)</td>
</tr>
</tbody>
</table>

---

**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.
(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
Disabling the patch occurrence measure regions are stylized with the texture of stationary ones and vice versa. Therefore unnatural-looking results are produced, e.g., moving redo not provide a sufficient movement in the subsequent frames.

The reason why the output looks reasonable even when $D^{\text{temp}}$ is not active is the fact that texture coherence is also jointly enforced in quality becomes more noticeable. Without $D^{\text{temp}}$ the output is not coherent in time and by switching off $D^{\text{ext}}$ the texture details become slightly deteriorated. The reason why the output looks reasonable even when $D^{\text{ext}}$ is not active is the fact that texture coherence is also jointly enforced by $D^{\text{temp}}$. When both terms ($D^{\text{ext}}$ and $D^{\text{temp}}$) 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 [JvST19]—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 fluidimation. 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.

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 $D^{\text{bound}}$ leads to a loss of natural transition between the artistic media and the canvas. When the motion direction alignment $D^{\text{dir}}$ 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 $D^{\text{bound}}$ slightly reinforces correct direction. When both terms are set to zero, the resulting flow becomes completely random. Ignoring motion extent term $D^{\text{ext}}$ 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 $\Omega$ leads to visible wash-out artifacts since the variety of source patches becomes significantly reduced. Without $D^{\text{temp}}$ the output is not coherent in time and by switching off $D^{\text{ext}}$ the texture details become slightly deteriorated. The reason why the output looks reasonable even when $D^{\text{ext}}$ is not active is the fact that texture coherence is also jointly enforced by $D^{\text{temp}}$. When both terms ($D^{\text{ext}}$ and $D^{\text{temp}}$) 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

![Figure 9: A selection of target animation sequences used for evaluation.](image)

![Figure 10: Average timings for source-target combinations presented in Fig. 10.](image)

![Figure 11: User study results. On a likert scale 18 participants where asked to what extent our approach and two previous techniques (LazyFluids [JFA+15] and EbSynth [JvST19]) 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.](image)

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.

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 Šykorová 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


Platkević et al. / Fluidymation


Appendix A: Flow field construction

When either \( S_{\text{flow}} \) or \( T_{\text{flow}} \) is not provided as an input to our algorithm, we can approximate them using the motion at the boundaries of masks \( S_{\text{mask}} \) and \( T_{\text{mask}} \). 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 \( I_{\text{flow}} \) at the boundary of \( S_{\text{mask}} \) is perpendicular to its tangent, and therefore parallel to the gradient of a distance field \( d_{\text{dist}} \) computed from the mask’s boundary \( [FH12] \) (see Fig. 13c). The magnitude of \( I_{\text{flow}} \) 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
I with (bottom) the distance field temporal smoothing applied. 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 $I^\text{mask}_i$ (see Fig. 13e):

$$I^{\text{flow}}_i = \left( I^\text{dist}_i - I^\text{dist}_{i-1} \right) \| \nabla I^\text{dist}_i \| \nabla I^\text{dist}_{i-1} \| .$$

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

To improve smoothness in the temporal domain and enable sub-pixel movements which cannot be captured by binary masks, we convolve the estimated sequence $I^{\text{flow}}$ 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 $\text{SHIFT}NFF$ is described in Section 3.4 and operations $\text{UPSCALE}NFF$, $\text{DOWNSCALE}NFF$, and $\text{MERGE}NFF$ are specified in Section 3.6. Inputs $S^\text{guides}$ and $T^\text{guides}$ denote all additional guiding channels and their weights (i.e., bound, int, flow, rot, dist, temp, ext+, ext−, and unit).

Algorithm 1 Multi-scale optimization

\[
\begin{align*}
& NNF_{\text{prev}} \leftarrow \text{empty mapping} \\
& T_i \leftarrow \text{empty image} \\
& \text{for each level } i \text{ do} \\
& \quad \text{if } i \text{ is the coarsest level then} \\
& \quad \quad \text{if first frame then} \\
& \quad \quad \quad NNF_{\text{init}} \leftarrow \text{random mapping} \\
& \quad \quad \text{else} \\
& \quad \quad \quad NNF_{\text{init}} \leftarrow \text{DOWNSCALE}NFF(NNF_{\text{prev}}) \\
& \quad \text{end if} \\
& \quad \text{else} \\
& \quad \quad \text{if first frame then} \\
& \quad \quad \quad NNF_{\text{init}} \leftarrow \text{UPSCALE}NFF(NNF_{i+1}) \\
& \quad \quad \text{else} \\
& \quad \quad \quad NNF^\uparrow_i \leftarrow \text{UPSCALE}NFF(NNF_{i+1}) \\
& \quad \quad \quad NNF_{\text{prev}} \leftarrow \text{DOWNSCALE}NFF(NNF_{\text{prev}}) \\
& \quad \quad \quad NNF_{\text{init}} \leftarrow \text{MERGE}NFF(NNF^\uparrow_i, NNF_{\text{prev}}) \\
& \quad \text{end if} \\
& \quad \text{end if} \\
& \quad \text{end for} \\
& \text{output } T^\text{rgb}_i \\
& NNF_{\text{prev}} \leftarrow \text{SHIFT}NFF(NNF)
\end{align*}
\]

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

Algorithm 2 Texture Optimization

\[
\begin{align*}
& \text{function } \text{OPTIMIZE}(S^{\text{rgb}}, S^\text{mask}, S^\text{guides}, T^\text{mask}, T^\text{guides}, NNF_{\text{init}}) \\
& \quad NNF \leftarrow NNF_{\text{init}} \\
& \quad T^\text{rgb}_i \leftarrow \text{VOTE}(S^{\text{rgb}}, NNF) \\
& \quad \text{for } j \leftarrow 1 \ldots \text{opt\_iters} \text{ do} \\
& \quad \quad NNF \leftarrow \text{FINDDNN}(S^{\text{rgb}}, S^\text{mask}, S^\text{guides}, T^\text{rgb}_i, T^\text{mask}, T^\text{guides}) \\
& \quad \quad T^\text{rgb}_i \leftarrow \text{VOTE}(S^{\text{rgb}}, NNF) \\
& \quad \text{end for} \\
& \quad \text{return } NNF, T^\text{rgb}_i \\
& \text{end function}
\end{align*}
\]