DynTypo: Example-based Dynamic Text Effects Transfer

Yifang Men, Zhouhui Lian*, Yingmin Tang, Jianguo Xiao
Institute of Computer Science and Technology, Peking University, China

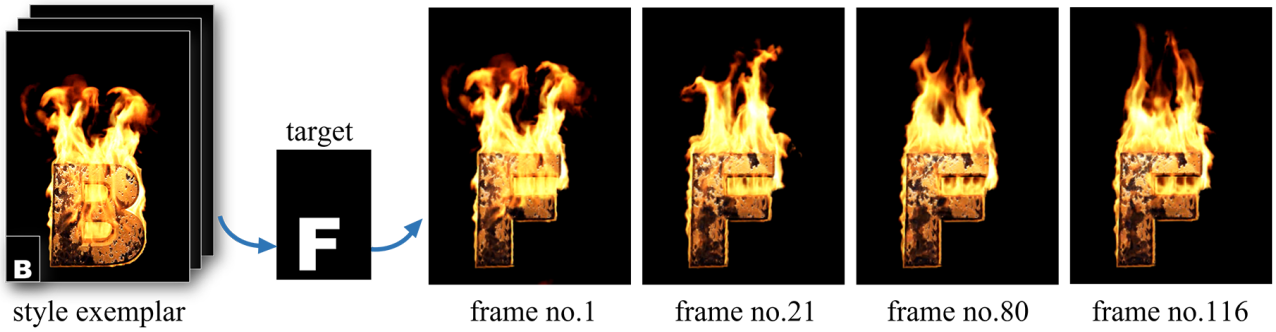


Figure 1: The dynamic text effects from an exemplar video (far left) can be transferred to a target text image (bottom) using our method. Exemplar video: © Thirdmenson via YouTube.

Abstract

In this paper, we present a novel approach for dynamic text effects transfer by using example-based texture synthesis. In contrast to previous works that require an input video of the target to provide motion guidance, we aim to animate a still image of the target text by transferring the desired dynamic effects from an observed exemplar. Due to the simplicity of target guidance and complexity of realistic effects, it is prone to producing temporal artifacts such as flickers and pulsations. To address the problem, our core idea is to find a common Nearest-neighbor Field (NNF) that would optimize the textural coherence across all keyframes simultaneously. With the static NNF for video sequences, we implicitly transfer motion properties from source to target. We also introduce a guided NNF search by employing the distance-based weight map and Simulated Annealing (SA) for deep direction-guided propagation to allow intense dynamic effects to be completely transferred with no semantic guidance provided. Experimental results demonstrate the effectiveness and superiority of our method in dynamic text effects transfer through extensive comparisons with state-of-the-art algorithms. We also show the potentiality of our method via multiple experiments for various application domains.

1. Introduction

Dynamic typography is an essential visual element in a variety of media such as films, advertisements and video clips. Using typographies with fantastic dynamic effects can markedly enhance the atmosphere of the scene and make it be more attractive to viewers. However, creating a dynamic text requires a series of complicated operations with editing software like Adobe After Effects and typically takes several hours even for skilled designers. Moreover, restructuring different texts with an existing dynamic effect can be painstaking, as huge amounts of repetitive operations are involved. It is already a labor-intensive work to generate a dynamic typography with the same effect for 26 letters in the English alphabet, let alone large-scale font libraries (e.g., Chinese fonts) with huge amounts of characters.

To solve the aforementioned problem, we introduce "DynTypo", a brand-new system that automatically transfers various dynamic effects such as burning flame, flowing water and billowing smoke from the source exemplar to the target plain texts. The input to our system includes a source text image, a source stylized video with dynamic effects and a target text image. With DynTypo the target stylized video with the same dynamic text effects in source can be generated (see Figure 1). Our system allows users to transfer an observed instance of dynamic typography to a new text and even non-experts can quickly and easily produce sophisticated dynamic texts in their own projects. Compared to LiveType [30], the earliest model to generate dynamic typography via simple shape deformations, this paper has the completely different motivation mentioned below.

Creating a dynamic typography with the chosen style from an exemplar poses several challenges. First, we use only a single image of the target text as input to animate and

* Corresponding author. E-mail: lianzhouhui@pku.edu.cn

stylize it. In contrast to previous works of stylized animation (e.g., [6, 19, 14]) that feed a target video or a sequence of motion fields as input, it is more difficult to maintain the temporal coherence without flow guidance. Second, the composition of effect patterns is more complicated with realistic fluid animation than a single designed image [36, 3]. And the static and dynamic effects are often blended making it easily suffer pulsation artifacts. Third, little semantic information is contained in raw text images and no guidance for intense text effects (such as flame effects in Figure 1).

To address these challenges, we propose a novel dynamic text effects transfer method to synthesize high-quality results with temporal smoothing and sophisticated dynamic effects. Unlike previous works [9, 13, 14] searching the Nearest-neighbor Field (NNF) for target synthesis in a frame-by-frame fashion, we simultaneously optimize the textural coherence across all keyframes to find a common NNF for all temporal frames, with which any frame of output sequence can be easily computed. In this way, we implicitly achieve a perfect temporal consistency with the static NNF and preserve the spatial texture continuity at the same time. Keyframes are extracted automatically by evaluating the intensity of particle movements and used to construct the semantic similarity term and spatial-temporal coherence term for dynamic guidance in the synthesis process. Moreover, we introduce a guided NNF search with the distance-based weight map and Simulated Annealing (SA) for deep direction-guided propagation to ensure that complicated effects can be completely synthesized. Our main contributions are as follows:

- We are the first to achieve the automatic generation of dynamic typography with sophisticated dynamic effects, and the effectiveness of our system is demonstrated via extensive experiments.
- We tackle the challenge of eliminating temporal artifacts such as flickers and pulsations by utilizing a single shared NNF and preserve spatial continuity at the same time via parallel optimization across all keyframes.
- We develop a guided NNF search by integrating the idea of Simulated Annealing and weight map, implementing deep direction guided propagation to make complicated intense effects without semantic guidance propagate more sufficiently.

2. Related Work

Up to now, many techniques have been proposed to synthesize stylized animations. Texture advection is a common approach to produce an animation by applying a velocity field to a static image. This technique was firstly proposed by [25] and later extended by [27]. A follow-up work by [8] utilized the bidirectional texture advection along lines of optical flow to create watercolor-like animations from videos. [37] proposed the Lagrangian texture advection to generate animated textures preserving local properties while following the velocity field. However, this method easily suffers from texture distortion for a long-time sequence and needs to be alleviated by new source blending. Thus, it fails to blend textures in different luminance and synthesize reasonable textures.

Another type of approach is to adopt an example-based texture synthesis technique. Earlier methods [12, 11] mostly relied on additional spatial constraints in the image quilting procedure to achieve texture transfer. A later work by [22] presented an optimization-based method to produce high-quality results with Expectation Maximization (EM)-like algorithm and image pyramid. They also animated a static image to a video via flow-guided synthesis [22, 7], which can be regarded as a combination of texture advection and synthesis. Later, [4, 5] introduced the PatchMatch algorithm to accelerate the search step of EM optimization.

The optimization-based method has been extended to a variety of scenes, such as video completion [34], stylized 3D rendering [13] and text effects transfer [36]. Yang et al. [36] assumed that the spatial distribution for text effects is closely related to their distances to the text skeleton and used the distance-based distribution term to guide synthesis. Here we seek to extend this application [36] to animation and enable it to transfer more delicate text effects, which do not satisfy the distribution regularity mentioned above. [26] proposed a general-purpose texture transfer method to guide the spatial distribution of internal structures. However, it needs additional stroke segmenting operations for better structure preservation in the text effects transfer task. Previous stylized animation synthesizing methods mostly leveraged optical flow to constrain the temporal coherence [6, 15, 19]. Unfortunately, these methods cannot perform well in our scenario where only a single target text image is used as input and the source exemplar contains complicated dynamic effects with the combination of dynamic and static patterns.

Neural-based image style transfer proposed by [17, 18] applied pre-trained deep convolutional networks like VGG-19 [31] to address the problem. It presented impressive style transfer results and attracted many follow-ups in model improvement [23] and algorithm acceleration [20, 33]. [29] and [2] extended it to video stylization with the optical flow constrained initialization and loss function. An online video style transfer system was proposed by [10] considering temporal coherence for feed-forward networks. More recently, deep convolutional networks are used to synthesize dynamic textures for fluid animations [35, 16, 32]. Despite the great success of neural style transfer methods, there still exists a key limitation that they are unable to reproduce tex-

Figure 2: Overview of the dynamic text effects transfer problem. With three inputs S_{text} (source text image), S_{sty} (source stylized video aligned to S_{text}) and T_{text} (target text image), the target stylized video T_{sty} with dynamic effects as S_{sty} can be generated.

tures with high-quality low-level details (as demonstrated in [14]).

3. Our Approach

In this section, we first formulate the task of dynamic text effects transfer and outline a strategy to solve the problem (Section 3.1). Then we leverage extracted keyframes (Section 3.2) to constrain the spatial-temporal coherence (Section 3.3) and introduce a guided NNF search algorithm with SA and weight map for deep direction-guided propagation (Section 3.4). Finally, we utilize a joint formulation (Section 3.5) to encompass the above-mentioned issues into one optimization problem and introduce two extended versions of our method (Section 3.6).

3.1. Problem Formulation and Analysis

Given a source text image S_{text} , its stylized animation S_{sty} and a target text image T_{text}, the goal is to synthesize the target stylized animation T_{sty} such that $S_{text} : S_{sty} :: T_{text} : T_{sty}$ (see Figure 2).

The native extension of applying text effects transfer to consecutive frames independently would result in strong flickers due to subtle appearance changes (e.g., motion, luminance, etc.) in the source animation S_{sty}. Moreover, unnoticeable changes in input would be amplified due to the unstable projection from source to target. To preserve temporal consistency, a common solution is to warp the previously synthesized frame by following the motion field and use it to guide the synthesis of the current frame (e.g., [14, 19]). However, no target motion field can be estimated from a single target image and the common solution still suffers from temporal pulsations when we use source-to-target projected motion field. Thus, a different methodology is required.

Through experiments, we observed that flickers always occur in pixels where the correspondences between source and target are unstable over the time. If we use the same NNF throughout the time domain, the target patch will imitate a series of dynamic changes containing texture appearances in the temporal sequence. In other words, the image patch will be extended to temporal patch by keeping the NNF unchanged. Such an intuitive strategy kills two birds with one stone: 1) it makes stylized results temporally smooth by removing the temporal domain for search space and only the two-dimensional plane needs to be considered; 2) it speeds up the synthesis procedure since all frames share a common NNF and the iterative optimization for NNF search, computationally expensive in runtime, is only performed once.

However, the strategy outlined above only preserves the temporal consistency and ignores the spatial continuity. Directly applying an NNF searched at one frame to others is prone to produce discontinuous textures. To further maintain the spatial continuity for all frames, we simultaneously optimize the textural coherence and reconstruct target stylized images across all keyframes to find the common NNF. Moreover, we utilize the direction guided deep propagation to encourage the text effects to spread around from the text contour, where patches contain more semantic features and could easily find correct correspondences from the source. In this way, it is sufficient for our method to transfer stylized appearance from source to target.

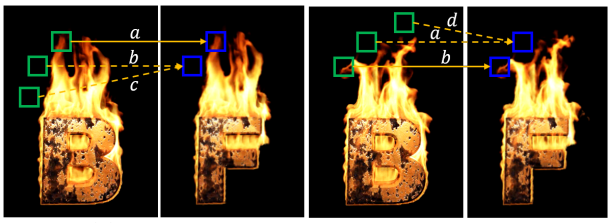
3.2. Keyframe Extraction

In our system, keyframes are extracted based on the motion intensity and used to construct spatial-temporal coherence described in the next section. As shown in Figure 3 (a), it is easy for the left pair to find the best correspondence a but it is confused to choose one between b and c, which are completely identical with background patches. However, the right pair make it clearly that correspondence b is required for the texture continuity at current frame. We can introduce more constraints for texture coherence with keyframes containing more representative textural features. Thus, the keyframes are expected to show violent movements compared with previous frames, especially more emitters-places where new fluid is spawned (pixels marked in pink in Figure 3 (c)). We first extract dynamic pixels using color-changed ones at consecutive frames and compute changed values by

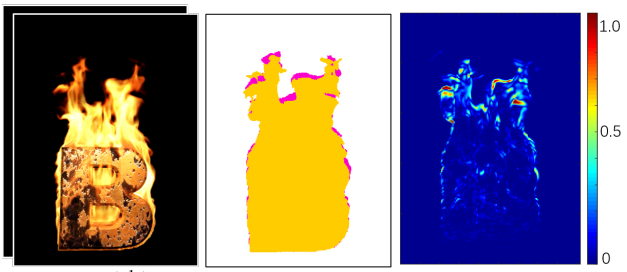
$$m_t = \mid g(S^t_{sty}) - g(S^{t-1}_{sty}) \mid, 2 \leq t \leq N, \qquad (1)$$

where m_t is the map of color-changed values at t^{th} frame and $g(S^t_{sty})$ is the gray image of source stylized image at t^{th} frame. The same goes for $g(S^{t-1}_{sty})$ at $(t-1)^{th}$ frame. N is the total number of video frames.

We normalize the color-changed map m_t with the Min-Max method and the normalized map $\widetilde{m}_t$ is shown in Figure 3 (d). Compared with emitters in Figure 3 (c)), emitters in map $\widetilde{m}_t$ have larger color-changed values due to new fluids spawned. So we use values in $\widetilde{m}_t$ as weights to evaluate the importance of each pixel when we compute the number



(a) Source-target pairs at different frames



(b) $S_{sty}^{t-1,t}$ (c) $M_{sty}^{t-1,t}$ (d) $\widetilde{m}_t$

Figure 3: Keyframe extraction. (a) Two source-target pairs at different frames. (b) Two source stylized images at consecutive frames. (c) The overlying masks for textures with special effects at consecutive frames. (d) The normalized color-changed map $\widetilde{m}_t$.

of color-changed pixels. The motion intensity can be computed by weighted numbers of color-changed pixels with the following equation

$$v_t = \sum_{p=1\ldots w\times h} \widetilde{m}_t(p), \tag{2}$$

where v_t is the value of motion intensity at the t^{th} frame and $\widetilde{m}_t(p)$ represents the value of each pixel in map $\widetilde{m}_t$. Then, the frames with top β motion intensity are picked up as keyframes.

3.3. Spatial-temporal Coherence

After extracting keyframes, we construct the spatial-temporal coherence term to preserve both temporal consistency and spatial texture continuity. For temporal consistency, we find a single, shared NNF that would simultaneously optimize the textural coherence and reconstruct target images across all keyframes. Since the NNF is static, it implicitly achieves a perfect temporal coherence. For example, once the correspondence between patch N_q in target and patch N_p in source is confirmed, the image patch N_q will imitate the appearance variances of N_p over video sequences and pixels in target will acquire motion properties of their correspondences implicitly. For spatial continuity, we search for target patches that are independently similar to the source at each keyframe and employ the sum of this source-target similarity across all keyframes as spatial-temporal coherence term, a soft constraint for texture transfer guidance. We show the intuitive idea in Fig. 4.

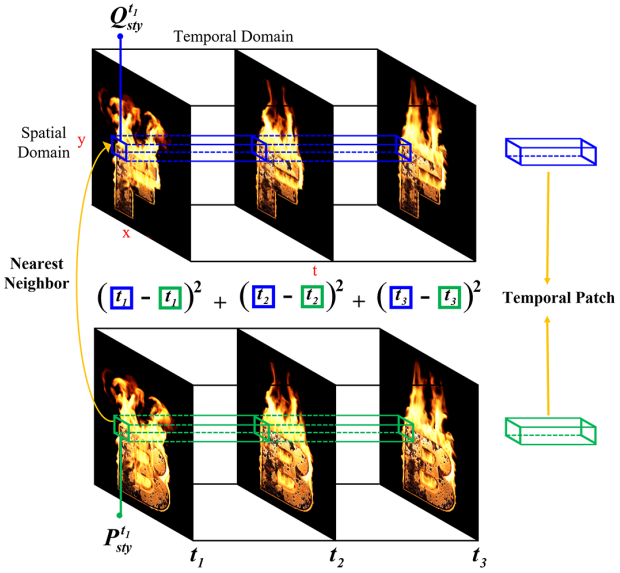


Figure 4: Visualization of the spatial-temporal coherence term. This term is constructed by computing the independent similarity between source and target at keyframes. We find a single shared NNF that would simultaneously optimize the textural coherence across all keyframes. Since the NNF is static over video sequences, it implicitly achieves a perfect temporal coherence.

This simple strategy improves the final results significantly as shown in Fig. 5. Without the spatial-temporal term for a common NNF, the results in Fig. 5 (b) suffer from severe temporal flicks. With a common NNF introduced but only the first frame used for NNF search, the results in Fig. 5 (c) become temporally smooth but spatially discontinuous at subsequent frames. When the spatial-temporal term is introduced with keyframes selected by uniform sampling, texture discontinuity in Fig. 5 (d) can be obviously improved but still exists. Fig. 5 (e) shows our results with both temporal consistency and spatial continuity.

3.4. Guided Nearest-neighbor Field Search

Our NNF search is designed based on PatchMatch [4] using propagation and random search. In addition to the regular propagation, we extend the propagation by incorporating our distance-based weight map for direction guidance, which encourages the text effects to spread around from the text contour. To make complicated textures propagate more sufficiently, we introduce the idea of Simulated Annealing [21] (an optimization method imitating the physical process of heating a material) to our method to get a self-adaptive propagation probability, which declines as the temperature decreases.

Direction Guided Propagation. In our dynamic text effects transfer, raw text images (S_{text} and T_{text}) are provided as semantic inputs. With more features in the boundary

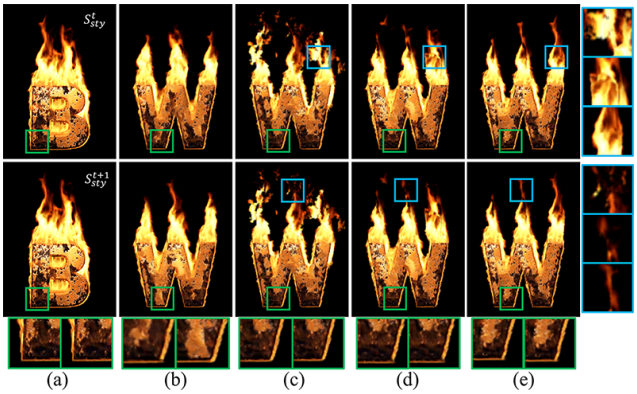


Figure 5: Effects of the spatial-temporal term and keyframe selection. (a) Source stylized images at t^{th} and $(t+1)^{th}$ frames. (b) Stylized results at consecutive frames synthesized without the spatial-temporal term. (c) Results synthesized with a common NNF but no keyframe. (d) Results synthesized with a common NNF and uniformly-spaced keyframes. (e) Results synthesized with a common NNF and our motion-based keyframes. Close-up views for temporally smooth (in green) and spatially continuity (in blue).

patches of T_{text}, we utilize the direction guided propagation to lead the flow of information outward from the boundary. As shown in Figure 6, we first compute the distance between each pixel q of T_{text} and the contour of text Ω (outline in red). For each patch N_q with the center coordinate q, we define its weight map based on the distance as

$$\alpha_{q'} = \gamma^{-(d_\perp(q',\Omega)-d_\perp(q,\Omega))}, \tag{3}$$

where $\alpha_{q'}$ is the weight of pixel q' in patch Nq, the base γ is equal to 2 and $d_\perp(q',\Omega)$ is the distance between q' and its nearest pixel on the text contour Ω. We normalize the weights of all pixels in each patch so that the normalized weight of the center pixel α_q equals to 1. The smaller the distance between a pixel to the text contour is, the larger the pixel's weight will be. The weight map is then used to compute the patch similarity in NNF search and reconstruct the target image in vote step for optimization. In this manner, we can direct the propagation outward from the text contour for complicated textures without semantic guidance.

Deep Propagation. As described in Algorithm 1, we set the initial and terminal temperature as T_0 and T_f, respectively. The parameters φ_{cur} and φ_{total} denote the current iteration time and the total time of EM iterations (described in Section 3.5), respectively. The temperature decreases with iterations. A weaker candidate solution generated in neighbors is accepted with a probability, which reduces when the temperature is decreasing. Thus, the effect textures can be propagated more sufficiently in the initial stage and later only better correspondences will be accepted. In this way, the stylized textures can be thoroughly spread with few semantic information provided. The effects of the weight map and SA for the deep direction-guided propagation are shown in Figure 7.

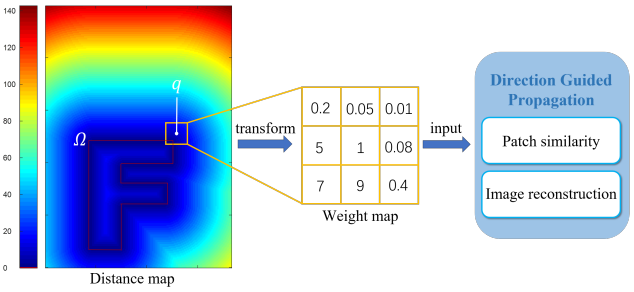


Figure 6: Utilizing the weight map for direction guided propagation. The weight map of each patch with the center coordinate q is computed using the distance to the text contour Ω. By leveraging the weight map to compute the patch similarity in search step and reconstruct the target image in vote step, we achieve direction guided propagation which leads the flow of information outward from the boundary.

ALGORITHM 1: Deep Propagation with Simulated Annealing

Input: $S_{text}, T_{text}, S_{sty}, T_{sty}, NN, \varphi_{cur}, \varphi_{total}, T_0, T_f$
Output: Nearest neighbor fields NN
repeat
  Generate candidate solution NN' and compute its energy value E' using the weight map;
  Compute $\Delta E = E' - E$;
  if $\Delta E < 0$ **then**
    Update nearest neighbor $NN = NN'$;
  else
    Set $T_k = T_0 - \frac{\varphi_{cur}}{\varphi_{total}}(T_0 - T_f)$;
    Compute acceptance probability $prob = min(1, exp\{-\frac{\Delta E}{T_k}\})$;
    if $prob > \xi(random(0,1))$ **then**
      Update nearest neighbor $NN = NN'$;
    end
  end
until $nUpdate \leq 0$;

3.5. Joint Formulation

In this section, the implementations proposed in Sections 3.2, 3.3 and 3.4 are combined into one joint patch-based optimization problem. The energy function concatenating the semantic guidance and spatial-temporal coherence is defined as follows

$$E = \sum_{Q \in T} \min_{P \in S} (\lambda D(P_{text}, Q_{text}) + \sum_{t \in kf} D(P^t_{sty}, Q^t_{sty})), \tag{4}$$

where P and Q denote the patches in S_{text}/S^t_{sty} and T_{text}/T^t_{sty}, respectively. P_{text} and P^t_{sty} represent the patches in the source text image S_{text} and target stylized image

(a) τ=0, T_0=0 (b) τ=1, T_0=0 (c) τ=1, T_0=2 (d) τ=1, T_0=5

Figure 7: Effects of the weight map and SA. Results are synthesized with various configurations (binary value τ for weight map (1-with, 0-without) and initial temperature T_0 for SA).

S^t_{sty} at frame t. The same goes for patches Q_{text} and Q^t_{sty} in T_{text} and T^t_{sty}. We compute the distance between P_{text} and Q_{text} to ensure that the distribution of stylized textures well follows the target text image. The distance between P^t_{sty} and Q^t_{sty} at keyframes kf is used to constrain the spatial-temporal coherence. The distance D is computed by the weighted L2-norm in RGB space using the weight map described in Section 3.4. λ is used to control the balance between semantic similarity and spatial-temporal coherence. We define it as a linear variable decreasing with iteration times for dynamic guidance as mentioned in [26].

To minimize Equation (4), we use the multi-scale EM-like iterations with two steps (i.e., guided NNF search and vote) performed alternatively. The guided NNF search described in Section 3.4.2 is used for deep direction-guided propagation. In the vote step, the target stylized images at keyframes are reconstructed simultaneously with the given NNF. We compute each pixel in T^t_{sty} with the weighted average color of co-located pixels in neighbor patches and we use weights presented in Section 3.4.1. The image patches over the entire video sequences are regarded as a whole and the optimization is performed once to find the common NNF. Finally, we apply the shared NNF to all frames to generate the dynamic typography with text effects as exemplar.

3.6. Further Extensions

Background Embedding. To make DynTypo widely applicable in real situations, we extend the method with arbitrary background embedding. The texts with texture effects can be easily extracted from the target stylized image using image segmentation techniques such as [1]. Then, we seamlessly insert the texts into a background image by leveraging Poisson Image Editing [28], a powerful object insertion method in a seamless and effortless manner with no need for precise object delineation. In this way, even source videos with complex background (e.g., gradient color, illumination) can be perfectly processed. Figure 8 shows a comparison of synthesis results obtained with and without the proposed background refinement.

Potential Applications. Actually, no step specially tailored for texts is used in our method, and thus the method can be easily extended to dynamic effects transfer between other objects with semantically related or geometrically similar parts. We show an example for animating portraits in Figure 9 and more results can be found in the supplemental video (online version at https://youtu.be/FkFQ6bV1s-o).

Inputs Output (without) Output (with)

Figure 8: Comparison of synthesized results with and without background refinement.

Figure 9: Animating portraits. With a source exemplar as driving video (first row), a target doodle can be turned into the stylized portrait animation (second row).

4. Results

We implement our method in Matlab. It takes around 20 minutes with a 4 GHz quad-core CPU to synthesize a target stylized animation with 160 frames in the resolution of 500×360. For the NNF retrieval over all video sequences, we use an image pyramid of 10 levels with a fixed coarsest size (32×32). In each level, m optimizing iterations are performed with two steps (NNF search and vote), where m linearly decreases when synthesizing from coarse to fine. The patch size is fixed to 5×5 and the keyframe number $\beta = 15$.

To demonstrate the effectiveness of the proposed DynTypo, we transfer dynamic text effects with various styles to some representative types of glyphs (English letters, Chinese characters, handwritings) and illustrate that it performs better than other state-of-the-art methods.

4.1. Dynamic text effects transfer

We apply our dynamic texture transfer method to the set of English letters from A to Z and other representative glyphs such as Chinese characters and handwritings with some stylized examples we collected through the Internet. Various sophisticated dynamic effects such as burning flame, flowing water and other designed ones can be suc-

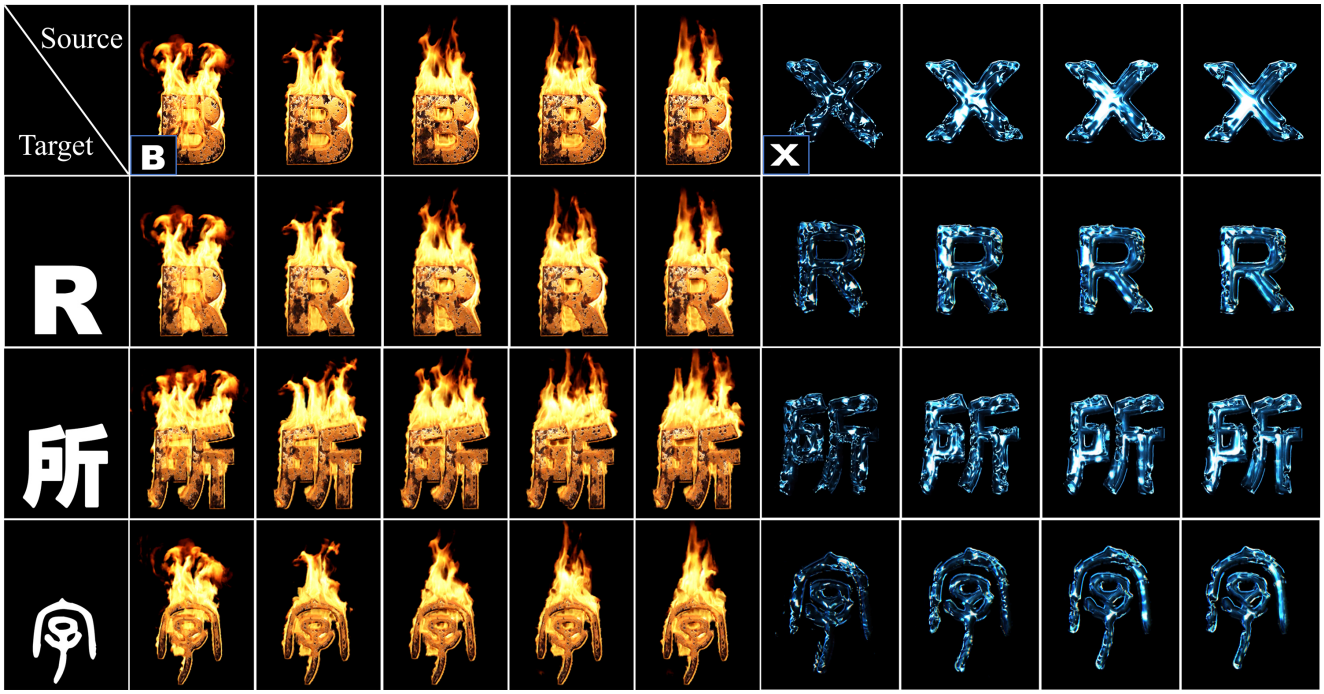


Figure 10: Results of applying different dynamic text effects to several representative glyphs (English letters, Chinese characters, handwritings).

cessfully transferred via our method. Results are shown in Figure 10 and more are available in the supplemental video (online version at https://youtu.be/FkFQ6bV1s-o).

4.2. Comparisons

In this subsection, we compare our method with several state-of-the-art style transfer approaches [22, 14, 36] for dynamic typography generation. Results of some representative frames are depicted in Figure 11 and the corresponding complete videos can be found in supplemental materials.

Flow-guided synthesis [22] is a pioneering approach for combining texture synthesis with the advection technique. Since it does not intrinsically support dynamic effects transfer with a single target text image, we add our semantic guidance term to its objective function and use this to synthesize the first frame of the target stylized animation. The flow field F_s is computed by using SiftFlow [24] to estimate the motion in source stylized sequences and F_s is transformed based on the correspondences between source and target to acquire the target flow field F_t. With the target flow field, the first frame can be animated by applying the flow-guided synthesis. As shown in the second row of Figure 11, limited by the accuracy of the estimated optical flow and transformation from source to target, synthesis results suffer from notable texture distortions after propagating over a long period. Moreover, small flow errors may accumulate, causing ghosting artifacts.

Still text effects transfer [36] exploits a distribution term to guide the synthesis process based on the high correlation between patch patterns and their distances to the text skeleton. One possible way to achieve dynamic effects transfer with the above-mentioned method [36] is to directly perform still text effects transfer frame-by-frame. However, without temporal constraint it suffers severe flickers. To alleviate the problem, we introduce a temporal coherence as described in [19] to this dynamic text effects transfer process. It improves temporal smoothing but still generates subtle trembling for short-term consistency, and fails to produce stable results over longer periods of time for long-term consistency (as shown in the third row of Fig. 11). Moreover, it is difficult to transfer the intense flame effect whose effect patterns do not distribute according to the distances.

Fišer et al. [14] proposed an animation stylization method tailored to portrait videos. We implement the method in our scenario by using semantic text images (S_{text} and T_{text}) as the segmentation guide and discarding the special treatments for facial features. As shown in the fourth row of Figure 11, the synthesized results are optimized over frames as we propagate NNF from the previous moment and use it as the initialization of the current frame. However, it generates unsatisfactory results in the early stage and suffers from temporal artifacts similar to [36].

From the last row of Figure 11, we can see that our method convincingly preserves realistic fluid effects, while avoiding disturbing temporal artifacts. The static effects like rust are stable and the dynamic effects like burning

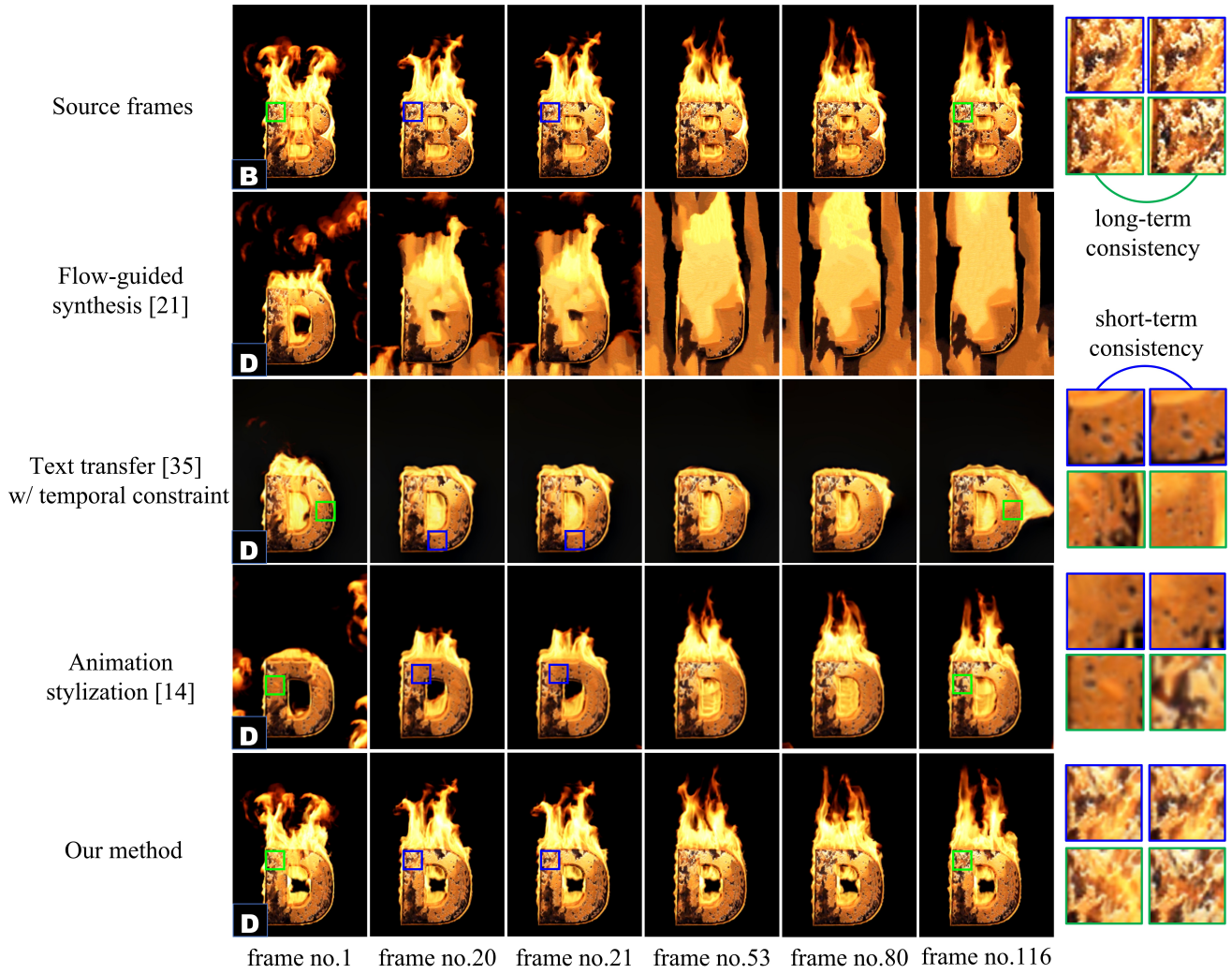


Figure 11: Comparison with state-of-the-art methods on dynamic text effects transfer. The short-term consistency is marked in blue with consecutive frames and the long-term consistency is marked in green for two frames with large time interval. The last column shows close-up views for the short-term and long-term consistency.

flame follow the motion of the source exemplar. Furthermore, under the same experimental settings, our method runs much faster (≈20 minutes for a stylized video) than other existing approaches such as [36] (≈2 minutes per frame and a total of 4∼6 hours) and [14] (≈3 minutes per frame and a total of 8 hours).

5. Conclusion

This paper presented a novel approach for example-based dynamic text effects transfer. To the best of our knowledge, our system is the first to automatically generate dynamic typography with realistic fluid effects, while avoiding disturbing temporal artifacts. Experimental results demonstrated that our method is effective for various glyphs with different effects and can be easily extended to dynamic effects transfer for general objects. We also believed that our spatial-temporal coherence guidance and the guided NNF search, which uses the weight map and simulated annealing for deep direction-guided propagation, could inspire future researches that will further improve the performance of semantic guided texture transfer and related animation stylization techniques.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No.: 61672043 and 61672056), National Key Research and Development Program of China (2017YFB1002601) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

References

[1] Texture segmentation using texture filters. https://ww2.mathworks.cn/help/images/examples/texture-segmentation-using-texture-filters.html.

[2] Alexander G Anderson, Cory P Berg, Daniel P Mossing, and Bruno A Olshausen. Deepmovie: Using optical flow and deep neural networks to stylize movies. *arXiv preprint arXiv:1605.08153*, 2016.

[3] Samaneh Azadi, Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. Multi-content gan for few-shot font style transfer. *arXiv preprint arXiv:1712.00516*, 2017.

[4] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009.

[5] Connelly Barnes, Eli Shechtman, Dan B Goldman, and Adam Finkelstein. The generalized patchmatch correspondence algorithm. In *European Conference on Computer Vision*, pages 29–43. Springer, 2010.

[6] Pierre Bénard, Forrester Cole, Michael Kass, Igor Mordatch, James Hegarty, Martin Sebastian Senn, Kurt Fleischer, Davide Pesare, and Katherine Breeden. Stylizing animation by example. *ACM Transactions on Graphics (TOG)*, 32(4):119, 2013.

[7] Kiran S Bhat, Steven M Seitz, Jessica K Hodgins, and Pradeep K Khosla. Flow-based video synthesis and editing. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 360–363. ACM, 2004.

[8] Adrien Bousseau, Fabrice Neyret, Joëlle Thollot, and David Salesin. Video watercolorization using bidirectional texture advection. *ACM Transactions on Graphics (ToG)*, 26(3):104, 2007.

[9] Mark Browning, Connelly Barnes, Samantha Ritter, and Adam Finkelstein. Stylized keyframe animation of fluid simulations. In *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering*, pages 63–70. ACM, 2014.

[10] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. Coherent online video style transfer. In *Proc. Intl. Conf. Computer Vision (ICCV)*, 2017.

[11] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[12] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[13] Jakub Fišer, Ondřej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sỳkora. Stylit: illumination-guided example-based stylization of 3d renderings. *ACM Transactions on Graphics (TOG)*, 35(4):92, 2016.

[14] Jakub Fišer, Ondřej Jamriška, David Simons, Eli Shechtman, Jingwan Lu, Paul Asente, Michal Lukáč, and Daniel Sỳkora. Example-based synthesis of stylized facial animations. *ACM Transactions on Graphics (TOG)*, 36(4):155, 2017.

[15] Jakub Fišer, Michal Lukáč, Ondřej Jamriška, Martin Čadík, Yotam Gingold, Paul Asente, and Daniel Sỳkora. Color me noisy: Example-based rendering of hand-colored animations with temporal noise control. In *Computer Graphics Forum*, volume 33, pages 1–10. Wiley Online Library, 2014.

[16] Christina M Funke, Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Synthesising dynamic textures using convolutional neural networks. *arXiv preprint arXiv:1702.07006*, 2017.

[17] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.

[18] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2414–2423. IEEE, 2016.

[19] Ondřej Jamriška, Jakub Fišer, Paul Asente, Jingwan Lu, Eli Shechtman, and Daniel Sỳkora. Lazyfluids: appearance transfer for fluid animations. *ACM Transactions on Graphics (TOG)*, 34(4):92, 2015.

[20] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[21] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[22] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. In *ACM Transactions on Graphics (ToG)*, volume 24, pages 795–802. ACM, 2005.

[23] Tsung-Yu Lin and Subhransu Maji. Visualizing and understanding deep texture representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2791–2799, 2016.

[24] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. In *Dense Image Correspondences for Computer Vision*, pages 15–49. Springer, 2016.

[25] Nelson Max, Roger Crawfis, and Dean Williams. Visualizing wind velocities by advecting cloud textures. In *Proceedings of the 3rd conference on Visualization'92*, pages 179–184. IEEE Computer Society Press, 1992.

[26] Yifang Men, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. A common framework for interactive texture transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6353–6362, 2018.

[27] Fabrice Neyret. Advected textures. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 147–153. Eurographics Association, 2003.

[28] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on graphics (TOG)*, 22(3):313–318, 2003.

[29] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *German Conference on Pattern Recognition*, pages 26–36. Springer, 2016.

[30] Ariel Shamir and Ari Rappoport. *LiveType: a parametric font model based on features and constraints*. ACM, 1997.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] Matthew Tesfaldet, Marcus A Brubaker, and Konstantinos G Derpanis. Two-stream convolutional networks for dynamic texture synthesis. *arXiv preprint arXiv:1706.06982*, 2017.

[33] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, pages 1349–1357, 2016.

[34] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *IEEE Transactions on pattern analysis and machine intelligence*, 29(3), 2007.

[35] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative convnet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7093–7101, 2017.

[36] Shuai Yang, Jiaying Liu, Zhouhui Lian, and Zongming Guo. Awesome typography: Statistics-based text effects transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7464–7473, 2017.

[37] Qizhi Yu, Fabrice Neyret, Eric Bruneton, and Nicolas Holzschuch. Lagrangian texture advection: Preserving both spectrum and velocity field. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1612–1623, 2011.