# Supplementary Material: Learning Joint Spatial-Temporal Transformations for Video Inpainting

Yanhong Zeng[1,2*], Jianlong Fu[3†], and Hongyang Chao[1,2†]

[1] School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
[2] Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China
[3] Microsoft Research Asia
zengyh7@mail2.sysu.edu.cn, isschhy@mail.sysu.edu.cn, jianf@microsoft.com

This supplementary material presents the details of complete video inpainting results in Section A and our stationary mask generation algorithm in Section B. We provide the details of our network architectures in Section C and the implementation details in Section D. Finally, extensive ablation studies and analysis for the proposed Spatial-Temporal Transformer Networks for video inpainting can be found in Section E.

## A Video Inpainting Results

To compare visual results from different inpainting models in our main paper, we follow the setting used in most video inpainting works [5,7,12]. Specifically, we sample several frames from video results and show them in Figure 4 and Figure 5 in the main paper. However, sampled frames cannot truly reflect video results. Sometimes sampled static frames look less blurry but artifacts can be stronger in a dynamic video. Therefore, we provide 20 video cases for a more comprehensive comparison. [§].

In practice, we test all the videos in the test sets of DAVIS dataset [1] (90 cases) and Youtube-VOS dataset [11] (508 cases), and we randomly show 20 cases for visual comparisons. Specifically, five cases from DAVIS and five cases from Youtube-VOS are used to test filling stationary masks. Since Youtube-VOS has no dense object annotations, we sample 10 videos with dense object annotations from DAVIS to test filling moving masks following the setting used in previous works [7,8,12]. To conduct side-by-side comparisons and analysis, we select the two most competitive video inpainting models, LGTSM [3] and CAP [8] in the videos. LGTSM and CAP are fine-tuned multiple times to achieve optimal video results by the codes and models publicly provided by their official Github

*This work was done when Y. Zeng was an intern at Microsoft Research Asia.
†J. Fu and H. Chao are the corresponding authors.
§Video Demo: https://youtu.be/tgiWGdr1SnE

homepage ¶. We can find from the video results that our model outperforms the state-of-the-art models in most cases.

## B Stationary Mask Generation Algorithm

Inspired by Xu et al. [12], we use stationary masks and moving masks as testing masks to simulate real-world applications (e.g., watermark removal and object removal) in the main paper. As introduced in Section 4.1 in the main paper, on one hand, we use frame-wise foreground object annotations from DAVIS datasets [1] as moving masks to simulate applications like object removal. On the other hand, we generate random shapes as stationary masks to simulate applications like watermark removal. Specifically, for the task of removing watermarks, a user often draw a mask along the outline of a watermark. Inspired by previous mask generation algorithms [2,13], we propose a stationary mask generation algorithm to simulate such a behavior for drawing masks for watermarks. Specifically, the proposed algorithm randomly generates a set of control points around a unit circle, and then it smoothly connects these points into a closed cyclic contour by cubic Bezier curves. The details of the stationary mask generation algorithm are shown in Algorithm 1 as follows.

**Algorithm 1** Algorithm for stationary mask generation. *maxPointNum*, *maxLength* are hyper-parameters to control the statinary mask generation.

```
mask = zeros(imgHeight, imgWidth)
pointNum = random.uniform(maxPointNum)
startX = origX = random.uniform(imgWidth)
startY = origY = random.uniform(imgHeight)
angles = linspace(0, 2*pi, pointNum)
for i=0 to pointNum do
  length = random.uniform(maxLength)
  x = sin(angles[i]) * length
  y = cos(angles[i]) * length
  // comment: ensuring smoothness of contours
  Connect (startX, startY) to (x, y) by cubic Bezier curves.
  startX = x
  startY = y
end for
// comment: ensuring a closed cyclic contour
Connect (startX, startY) to (origX, origY) by cubic Bezier curves.
```

¶LGTSM: `https://github.com/amjltc295/Free-Form-Video-Inpainting`
CAP: `https://github.com/shleecs/Copy-and-Paste-Networks-for-Deep-Video-Inpainting`

## C Details of Network Architecture

The Spatial-Temporal Transformer Network (STTN) is built upon a generative adversarial framework. Specifically, the proposed STTN plays a role as a generator in the framework, and we adopt a Temporal PatchGAN (T-PatchGAN) [2] as our discriminator. The T-PatchGAN is composed of six layers of 3D convolution layers. Specifically, the T-PatchGAN learns to classify each spatial-temporal feature as real or fake, while STTN learns to fool the T-PatchGAN. Such an adversarial training allows STTN to model the local-global perceptual rationality and the spatial-temporal coherence of real videos [2]. In addition to the introduction in Section 3 in the main paper, we provide the details of the architectures of STTN and the T-PatchGAN in Table 1 and Table 2, respectively. Specifically, features inside holes are computed by dilated 2D convolutions. We argue that STTN is able to leverages multi-scale contexts and updates holes' features multiple times to improve attention results.

| Module Name | Filter Size | # Channels | Stride/Up Factor | Nonlinearity |
|---|---|---|---|---|
| 2dConv | $3 \times 3$ | 64 | 2 | LeakyReLU(0.2) |
| 2dConv | $3 \times 3$ | 64 | 1 | LeakyReLU(0.2) |
| 2dConv | $3 \times 3$ | 128 | 2 | LeakyReLU(0.2) |
| 2dConv | $3 \times 3$ | 256 | 1 | LeakyReLU(0.2) |
| Transformer $\times$ 8 | $1 \times 1$ | 256 | 1 | - |
| | $3 \times 3$ | | 1 | LeakyReLU(0.2) |
| BilinearUpSample | - | 256 | 2 | - |
| 2dConv | $3 \times 3$ | 128 | 1 | LeakyReLU(0.2) |
| 2dConv | $3 \times 3$ | 64 | 1 | LeakyReLU(0.2) |
| BilinearUpSample | - | 64 | 2 | - |
| 2dConv | $3 \times 3$ | 64 | 1 | LeakyReLU(0.2) |
| 2dConv | $3 \times 3$ | 3 | 1 | Tanh |

**Table 1.** Details of the proposed Spatial-Temporal Transformer Networks (STTN). "2dConv" means 2D convolution layers. "Transformer $\times$ 8" denotes stacking the proposed spatial-temporal transformers by eight layers. A transformer layer involves $1 \times 1$ and $3 \times 3$ convolutions (The overview of STTN is shown in Fig. 2 in the main paper). We use bilinear interpolations for all upsample operations on feature maps [9,10]. We show whether and what nonlinearity layer is used in the nonlinearity column.

## D Implementation details

**Hyper-parameters:** To maintain the aspect ratio of videos and take into account the memory limitations of modern GPUs, we resize all video frames into $432 \times 240$ for both training and testing [5,7,8,12]. During training, we set the batch size as 8, and the learning rate starts with 1e-4 and decays with factor

| Module Name | Filter Size | # Channels | Stride | Nonlinearity |
|---|---|---|---|---|
| SN-3dConv | $3 \times 5 \times 5$ | 64 | (1,2,2) | LeakyReLU(0.2) |
| SN-3dConv | $3 \times 5 \times 5$ | 128 | (1,2,2) | LeakyReLU(0.2) |
| SN-3dConv | $3 \times 5 \times 5$ | 256 | (1,2,2) | LeakyReLU(0.2) |
| SN-3dConv | $3 \times 5 \times 5$ | 256 | (1,2,2) | LeakyReLU(0.2) |
| SN-3dConv | $3 \times 5 \times 5$ | 256 | (1,2,2) | LeakyReLU(0.2) |
| SN-3dConv | $3 \times 5 \times 5$ | 256 | (1,2,2) | - |

**Table 2.** Details of the Temporal-PatchGAN (T-PatchGAN) discriminator [2]. The T-PatchGAN is composed of six 3D convolution layers. "SN-3dConv" denotes a 3D convolution layer that adopts spectral normalization to stabilize GAN's training [2].

0.1 every 150k iterations. Specifically, for each iteration, we sample five frames from a video in a consecutive or discontinuous manner with equal probability for training following Lee et al. [8,10].

**Computation complexity:** Our full model has a total of 12.6M trainable parameters. It costs about 3.9G GPU memory for completing a video from DAVIS dataset [1] by STTN on average. The proposed multi-scale patch-based video frame representations can enable fast training and inference. Specifically, our model runs at about 24.3fps with an NVIDIA V100 GPU and it runs at about 10.43 fps with an NVIDIA P100 GPU on average. Its total training time was about 3 days on YouTube-VOS dataset [11] and one day for fine-tuning on DAVIS dataset [1] with 8 Tesla V100 GPUs. The computation complexity of the proposed spatial-temporal transformers are denoted as:

$$\mathcal{O}(\sum_{l=1}^{D}\left[2\cdot(n\cdot\frac{HW}{p_w p_h})^2\cdot(p_w p_h C_l)+nk_l^2 HWC_{l-1}C_l\right])\approx\mathcal{O}(n^2), \tag{1}$$

where $D$ is the number of transformer layers, $n$ is the number of input frames, $HW$ is the feature size, $p_w p_h$ is the patch size, $k_l$ denotes for kernel size, and $C$ is the channel number of features. In Eq. (1), we focus on the computation complexity caused by the spatial-temporal transformers and leave out other computation costs (e.g., encoding and decoding costs) for simplification.

## E More ablation studies

To verify the effectiveness of the proposed Spatial-Temporal Transformer Networks (STTN) for video inpainting, this section presents extensive ablation studies on DAVIS dataset [1] with stationary masks.

**Effectiveness of utilizing distant frames:** we test our full model with different sample rates to prove the benefits of utilizing distant frames. Quantitative comparison results on DAVIS dataset [1] with stationary masks can be found in Table 3. The first row ($s > T$) means that the STTN takes only neighboring frames as input. Besides, the second row ($s = 20$) means that the STTN takes

both neighboring frames and distant frames that are uniformly sampled from the videos in a sampling rate of 20 frames.

Table 3 shows that leveraging visible contexts in distant frames helps in generating better results especially in terms of VFID with 5.70% relative improvements. Based on the observation that most videos in YouTube-VOS dataset [11] and DAVIS dataset [1] won't vary a lot within 10 frames on average, we set the sample rate as 10 in our full model to avoid sampling redundant frames and to save computation costs.

| Sample Rate | PSNR$^\star$ | SSIM(%)$^\star$ | $E_{warp}$ (%)$^\dagger$ | VFID$^\dagger$ |
|---|---|---|---|---|
| $s > T$ | 30.55 | 95.47 | 0.1802 | 0.158 |
| $s = 20$ | 30.62 | 95.55 | 0.1790 | 0.152 |
| $s = 10$ (ours) | **30.67** | **95.60** | **0.1779** | **0.149** |

**Table 3.** Ablation study by utilizing distant frames in different sampling rates. Our full model set $s = 10$. $^\star$ Higher is better. $^\dagger$ Lower is better.

**Effectiveness of masked normalization:** As shown in Eq. (3) and Eq. (4) in the main paper, we normalize the value of similarity by the dimension of vectors and filter out unknown regions for similarities calculating. In this part, we conduct comparisons between models with or without such a masked normalization in Table 4. Results show that such an operation is necessary since it brings improvements with a significant margin comparing with the one without masked normalization.

| | PSNR$^\star$ | SSIM(%)$^\star$ | $E_{warp}$ (%)$^\dagger$ | VFID$^\dagger$ |
|---|---|---|---|---|
| w/o masked norm. | 30.39 | 95.32 | 0.1849 | 0.162 |
| w/ masked norm. | **30.67** | **95.60** | **0.1779** | **0.149** |

**Table 4.** Ablation study for the effectiveness of masked normalization operation on similarity calculation. $^\star$ Higher is better. $^\dagger$ Lower is better.

**Effectiveness of the Temporal PatchGAN Loss:** Recent state-of-the-art deep video inpainting models that adopt attention modules often include a perceptual loss [6] and a style loss [4] as optimization objectives for perceptually pleasing results [8,10]. However, they do not leverage specially-designed losses for ensuring temporal coherence. Chang et al. propose a novel Temporal PatchGAN (T-PatchGAN) loss for ensuring both perceptual rationality and spatial-temporal coherence of videos [2,3]. However, they only apply T-PatchGAN on consecutive frames while the attention-based deep video inpainting models take discontinuous frames as input for training. We are the first to introduce T-PatchGAN in video inpainting models that adopt attention modules and show

that T-PatchGAN is also powerful in discontinuous frames. Such a joint optimization encourages STTN to learn both local-global perceptual rationality and coherent spatial-temporal transformations for video inpainting.

We verify the effectiveness of the T-PatchGAN loss by quantitative comparisons in Table 5. Compared with the STTN optimized by a style loss [4] and a perceptual loss [6] following previous works [8,10], the STTN optimized by a T-PatchGAN loss performs better by a significant margin, especially in terms of VFID with 6.9% relative improvements. We also provide a visual comparison in Fig. 1. The visual results show that the STTN optimized by a T-PatchGAN loss can generate more coherent results than the one optimized by a perceptual loss and a style loss. The superior results show the effectiveness of the joint spatial-temporal adversarial learning in STTN.

| losses | PSNR$^\star$ | SSIM(%)$^\star$ | $E_{warp}$ (%)$^\dagger$ | VFID$^\dagger$ |
|---|---|---|---|---|
| w/ style [4], w/ perceptual [6] | 30.38 | 95.35 | 0.1821 | 0.160 |
| w/ T-PatchGAN [2] | **30.67** | **95.60** | **0.1779** | **0.149** |

**Table 5.** Ablation study for different losses. $^\star$ Higher is better. $^\dagger$ Lower is better.

Specifically, perceptual loss and style loss have shown great impacts in many image generation tasks since they were proposed [4,6,9]. A perceptual loss computes $L_1$ distance between the activation maps of real frames and generated frames. A style loss is similar to the perceptual loss but aims at minimizing the $L_1$ distance between Gram matrices of the activation maps of real frames and generated frames. In practice, the activation maps are extracted from layers (e.g., *pool1*, *pool2* and *pool3*) of a pre-trained classification network (more details see [8,9,10]). With the help of extracted low-level features, the perceptual loss and the style loss are helpful in generating high-frequency details.

Unfortunately, perceptual and style losses are calculated on the features of a single frame and they are unable to leverage temporal contexts. When filling in a large missing region in videos, the perceptual and style losses are hard to enforce the generator to synthesize rational contents due to limited contexts. As a result, they have to generate meaningless high-frequency textures to match ground truths' low-level features. For example, for filling the large missing regions in the second and the third frames in Fig. 1, the STTN optimized by perceptual and style losses tends to generate high-frequency artifacts in the large missing regions. Similar artifacts can be found in the failure cases of previous works [2,9]. Since the T-PatchGAN is able to leverage temporal contexts to optimize the generator, there are fewer artifacts in the results by using the T-PatchGAN. For the above considerations, we use the T-PatchGAN loss instead of the perceptual and style losses in our final optimization objectives. In the future, we plan to design video-based perceptual and style losses which are computed on spatial-temporal features to leverage temporal contexts for optimization.

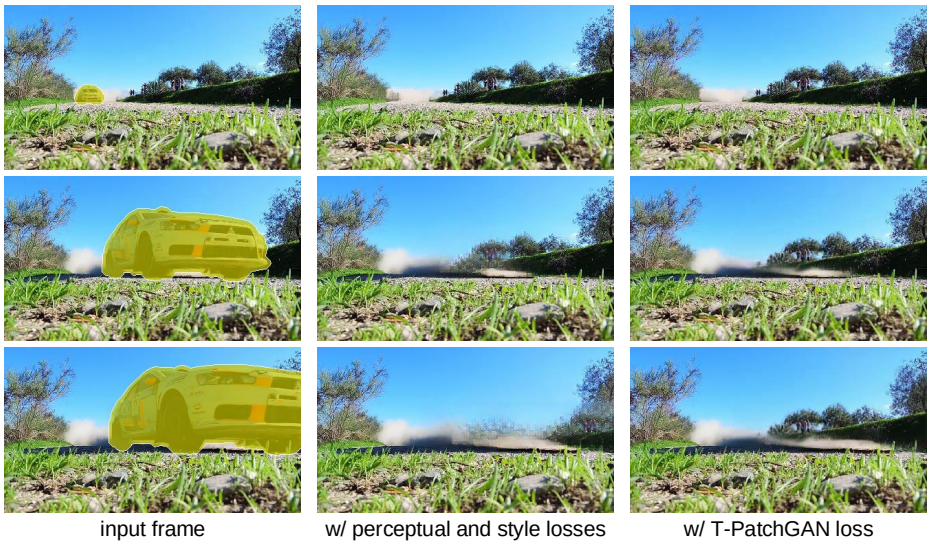


**Fig. 1.** Visual comparisons between an STTN optimized by a perceptual loss [6] and a style loss [4] and an STTN optimized by a T-PatchGAN loss [2]. These two models perform similarly in small missing regions, while in large missing regions, the model optimized by perceptual and style losses tends to generate artifacts in the missing regions. [Best viewed with zoom-in]

## References

1. Caelles, S., Montes, A., Maninis, K.K., Chen, Y., Van Gool, L., Perazzi, F., Pont-Tuset, J.: The 2018 davis challenge on video object segmentation. arXiv (2018)
2. Chang, Y.L., Liu, Z.Y., Lee, K.Y., Hsu, W.: Free-form video inpainting with 3d gated convolution and temporal patchgan. In: ICCV. pp. 9066–9075 (2019)
3. Chang, Y.L., Liu, Z.Y., Lee, K.Y., Hsu, W.: Learnable gated temporal shift module for deep video inpainting. In: BMVC (2019)
4. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR. pp. 2414–2423 (2016)
5. Huang, J.B., Kang, S.B., Ahuja, N., Kopf, J.: Temporally coherent completion of dynamic video. TOG **35**(6), 1–11 (2016)
6. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV. pp. 694–711 (2016)
7. Kim, D., Woo, S., Lee, J.Y., Kweon, I.S.: Deep video inpainting. In: CVPR. pp. 5792–5801 (2019)
8. Lee, S., Oh, S.W., Won, D., Kim, S.J.: Copy-and-paste networks for deep video inpainting. In: ICCV. pp. 4413–4421 (2019)
9. Liu, G., Reda, F.A., Shih, K.J., Wang, T.C., Tao, A., Catanzaro, B.: Image inpainting for irregular holes using partial convolutions. In: ECCV. pp. 85–100 (2018)
10. Oh, S.W., Lee, S., Lee, J.Y., Kim, S.J.: Onion-peel networks for deep video completion. In: ICCV. pp. 4403–4412 (2019)

11. Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J., Huang, T.: Youtube-vos: A large-scale video object segmentation benchmark. arXiv (2018)
12. Xu, R., Li, X., Zhou, B., Loy, C.C.: Deep flow-guided video inpainting. In: CVPR. pp. 3723–3732 (2019)
13. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. In: ICCV. pp. 4471–4480 (2019)