# Flow-edge Guided Video Completion Supplementary Material

Chen Gao[1], Ayush Saraf[2], Jia-Bin Huang[1], and Johannes Kopf[2]

[1] Virginia Tech [2] Facebook

## Overview

In this supplementary document, we provide additional implementation details and results to complement the main manuscript.

1. We summarize the complete pipeline of our algorithm in pseudo-code in Algorithm 1.
2. We show the runtime analysis and profiling to analyze the speed of our algorithm.
3. We describe the training details for the edge completion network.
4. We present additional visual examples of the ablation study, highlighting the effectiveness of our design choices.
5. We provide detailed per-sequence results in terms of PSNR, SSIM, and LPIPS on the DAVIS dataset.

## 1 Algorithm

We show our method pipeline in algorithm_illustration.mp4. We summarize our complete pipeline in Algorithm 1. Our pipeline consists of three main components: flow prediction, edge-guided flow completion, and video completion. We will release the pre-trained flow-edge completion model as well as the source code to facilitate future research.

**Algorithm 1:** Summary of our video completion algorithm.

**Input:** Color frames $\boldsymbol{I}_1 \dots \boldsymbol{I}_n$, mask frames $\boldsymbol{M}_1 \dots \boldsymbol{M}_n$.
**Output:** Completed frames $\boldsymbol{I}_1 \dots \boldsymbol{I}_n$ (updated in place).
**for** every frame $i \in 1 \dots n$ **do**
    Compute local and non-local optical flow $\boldsymbol{F}_{i \to j}$,
    $j \in \{i-1, i+1, 1, \lceil n/2 \rceil, n\}$ (Equations 1 and 2).
**for** each computed flow field $\boldsymbol{F}_{i \to j}$ **do**
    Extract flow edges $\boldsymbol{E}_{i \to j}$ using Canny edge detector [1].
    Complete flow edges $\tilde{\boldsymbol{E}}_{i \to j}$ using EdgeConnect [3] edge model.
    Complete flow $\tilde{\boldsymbol{F}}_{i \to j}$ with edge guidance (Equation 3).
    Compute flow error $\tilde{\boldsymbol{D}}_{i \to j}$ (Equation 4).
**while** any missing pixels exist in $\boldsymbol{M}_1 \dots \boldsymbol{M}_n$ **do**
    **for** every frame $i \in 1 \dots n$ **do**
        Obtain temporal neighbors through propagation.
        Fuse gradient images $\tilde{\boldsymbol{G}}_{x,i}$ and $\tilde{\boldsymbol{G}}_{y,i}$ (Equation 7).
        Reconstruct color image $\tilde{\boldsymbol{I}}_i$ (Equation 8).
        Update mask $\boldsymbol{M}_i(p) = 0$, where $|N(p)| \geq 1$.
    Select frame $\tilde{\boldsymbol{I}}_f$ with most remaining missing pixels.
    Complete $\tilde{\boldsymbol{I}}_f$ with DeepFill [5].
    Set $\boldsymbol{M}_f = 1$ (all pixels in this frame).
    Set $\boldsymbol{I} = \tilde{\boldsymbol{I}}$.

## 2 Runtime analysis and profiling

Following [2], we also show the detailed running time analysis of our method in Table 1. We report the time for each component of our method on the "CAMEL" video sequence under the object removal setting. The resolution is $960 \times 512$. There are 10721523 pixels being removed, which is 9.1% of the total pixels. Our method runs at 7.2 frames per minute.

Table 1: **Running time analysis.** We report the running time for each component of our method on the "CAMEL" video sequence under the object removal setting. The resolution is $960 \times 512$.

| | Component | Time (min.) |
|---|---|---|
| Flow completion | Flow prediction | 1.20 |
| | Edge extraction and completion | 0.45 |
| | Edge-guided flow completion | 4.20 |
| Video completion | Temporal propagation | 4.31 |
| | Spatial inpainting | 0.10 |
| | Poisson blending | 2.29 |
| Total | | 12.55 |

## 3 Training details

The only trainable component in our method is the flow edge completion network. We build our flow edge completion network upon the publicly available official implementation of EdgeConnect [3] edge model[1]. We load weights pretrained on the Places2 dataset [6], and then finetune on 60 sequences in DAVIS `2017-test-dev` and `2017-test-challenge` for three epochs.

Starting from the predicted flow between adjacent frames $i$ and $j$, $\boldsymbol{F}_{i\to j}$, we first calculate the flow magnitude image. We use the Canny edge detector [1] to extract a flow edge map $\boldsymbol{E}_{i\to j}$ from the flow magnitude image. We use the following parameters for the Canny edge detector [1]: Gaussian $\sigma = 1$, low threshold 0.1, high threshold 0.2. We randomly choose a mask from NVIDIA Irregular Mask Dataset testing split and resize it to $256 \times 256$.[2] We crop the flow edge map $\boldsymbol{E}_{i\to j}$ and the corresponding flow magnitude images to $256 \times 256$, and corrupt them with the mask. The input to the flow edge completion network is the mask (Figure 1a), the corrupted flow edge map (Figure 1b) and the corrupted flow magnitude image (Figure 1c). We train the network to complete the flow edge using batches of 8 randomly cropped $256 \times 256$ patches.

Note that our edge completion network does *not* receive any additional information regarding the stationary mask with a uniform grid of $5 \times 4$ square blocks during training.

## 4 Additional visual examples of the ablation study

In this section, we show additional visual examples of the ablation study to highlight the effectiveness of our design choices.

**Gradient domain processing.** We compare the proposed gradient propagation process with color propagation (used in [2,4]). Figure 2 shows a visual comparison. When filling the missing region with directly propagated colors, the result contains visible seams due to color differences in different source frames (Figure 2a). Our

[1] https://github.com/knazeri/edge-connect
[2] https://www.dropbox.com/s/01dfayns9s0kevy/test_mask.zip?dl=0

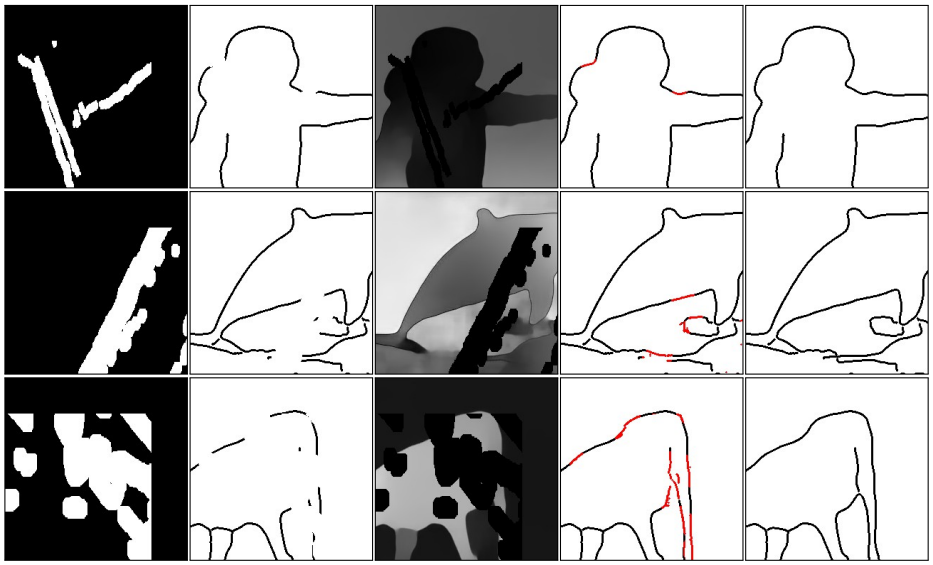



Fig. 1: **Flow edge completion.** Our flow edge completion network takes the mask, the corrupted flow edge map and the corrupted flow magnitude image as input, and complete the flow edge.

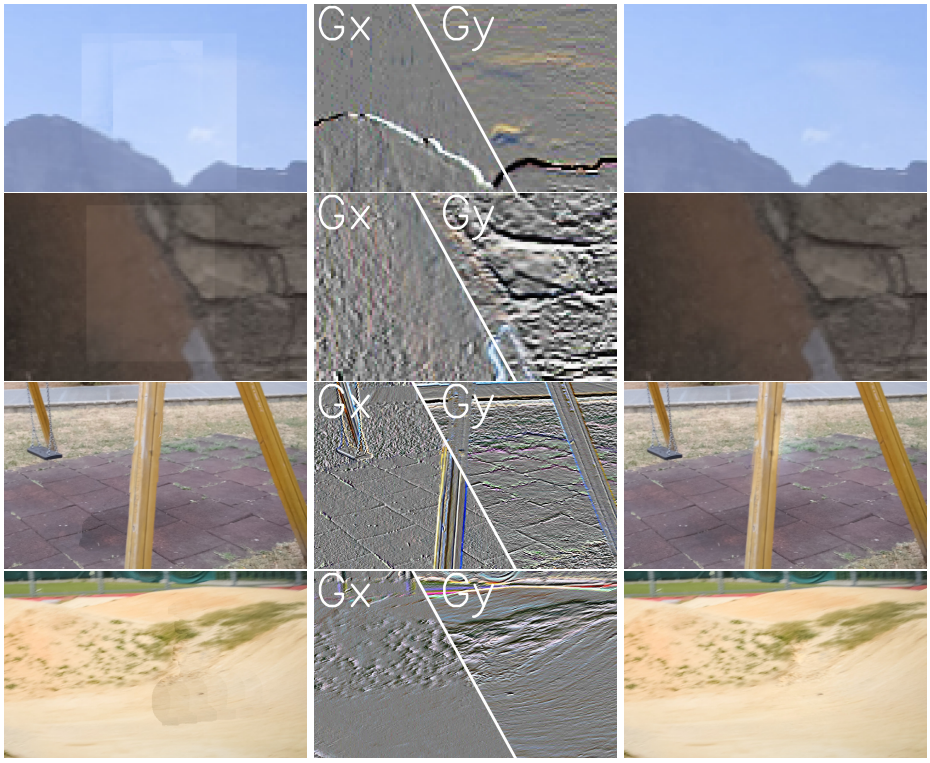
method operates in the gradient domain and does not suffers from such artifacts (Figure 2c).

**Non-local temporal neighbors.** We study the effectiveness of the non-local temporal neighbors. Figure 3 shows such an example. Using non-local neighbors enables us to transfers the correct contents from temporally distant frames.

**Edge-guided flow completion.** We evaluate the performance of completing the flow field with different methods. In Figure 4, we show examples of flow completion results using diffusion, a trained flow completion network [4], and our proposed edge-guided flow completion. The diffusion-based method maximizes smoothness in the flow field everywhere and thus cannot create sharp motion boundaries. The learning-based flow completion network [4] fails to predict a smooth flow field and sharp flow edges. In contrast, the proposed edge-guided flow completion fills the missing region with a piecewise-smooth flow and no visible seams along the hole boundary.

## 5 Per-sequence results on the DAVIS dataset

We show the detailed per-sequence results in terms of PSNR, SSIM, and LPIPS under stationary screen-space masks setting in Figure 5. Our proposed method



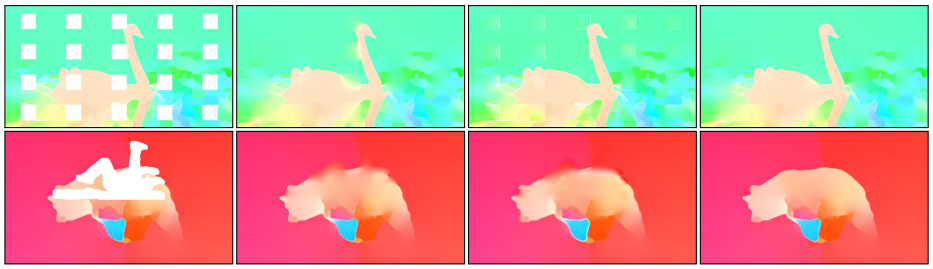

(a) Color propagation (b) Propagated x/y gradient (c) Reconstruction

Fig. 2: **Gradient domain reconstruction.** Previous methods operate directly in the color domain, which results in visible seams in the completed video (a). We propagate in the gradient domain (b), and reconstruct the results via Poisson reconstruction (c).

improves the performance over state-of-the-art methods for most of the video sequences, under all three metrics.

Without non-local neighbors With non-local neighbors

Fig. 3: **Non-local temporal neighbor ablation.** Video completion results *with* and *without* non-local temporal neighbors. The result without non-local neighbors (left) does not recover well from the lack of well-propagated content.



Input Diffusion Xu et al. [4] Ours

Fig. 4: **Flow completion.** Comparing different methods for flow completion. Our method has better ability to retain the piecewise-smooth nature of flow fields (sharp motion boundaries, smooth everywhere else) than the other two methods.


Fig. 5: **Per-sequence PSNR, SSIM and LPIPS on DAVIS under the stationary masks inpainting setting.**

## References

1. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. pp. 679–698 (1986)
2. Huang, J.B., Kang, S.B., Ahuja, N., Kopf, J.: Temporally coherent completion of dynamic video. ACM Transactions on Graphics (TOG) (2016)
3. Nazeri, K., Ng, E., Joseph, T., Qureshi, F., Ebrahimi, M.: Edgeconnect: Generative image inpainting with adversarial edge learning. In: ICCVW (2019)
4. Xu, R., Li, X., Zhou, B., Loy, C.C.: Deep flow-guided video inpainting. In: CVPR (2019)
5. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Generative image inpainting with contextual attention. In: CVPR (2018)
6. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE transactions on pattern analysis and machine intelligence **40**(6), 1452–1464 (2017)