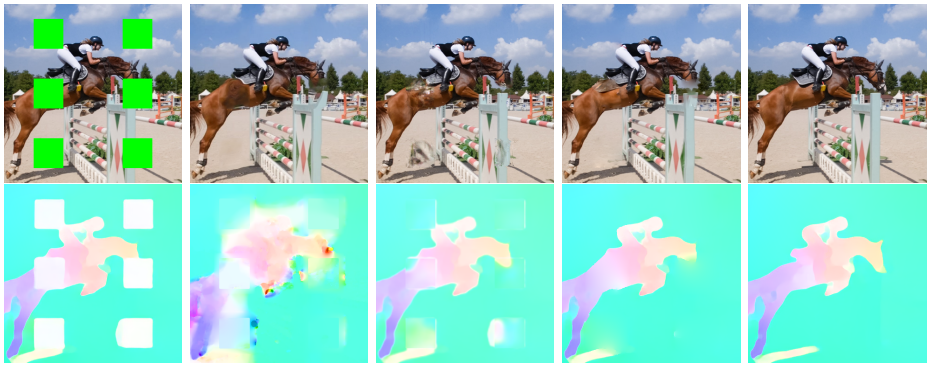
Flow-edge Guided Video Completion

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

[1] Virginia Tech [2] Facebook



(a) Input (b) Huang et al. (c) Xu et al. (d) Our result (e) Ground truth

Fig. 1: **Flow-edge guided video completion.** Our new flow-based video completion method synthesizes sharper motion boundaries than previous methods and can propagate content across motion boundaries using non-local flow connections.

Abstract. We present a new flow-based video completion algorithm. Previous flow completion methods are often unable to retain the sharpness of motion boundaries. Our method first extracts and completes motion edges, and then uses them to guide piecewise-smooth flow completion with sharp edges. Existing methods propagate colors among *local* flow connections between adjacent frames. However, not all missing regions in a video can be reached in this way because the motion boundaries form impenetrable barriers. Our method alleviates this problem by introducing *non-local* flow connections to temporally distant frames, enabling propagating video content over motion boundaries. We validate our approach on the DAVIS dataset. Both visual and quantitative results show that our method compares favorably against the state-of-the-art algorithms.

- Proposes a new "flow-based" video completion algorithm
- Introduces a new method "non-local" flow to overcome the limitations of the existing methods which propagate colours among local flow between adjacent frames.

1 Introduction

Video completion is the task of filling a given space-time region with newly synthesized content. It has many applications, including restoration (removing scratches), video editing and special effects workflows (removing unwanted objects), watermark and logo removal, and video stabilization (filling the exterior after shake removal instead of cropping). The newly generated content should embed seamlessly in the video, and the alteration should be as imperceptible

- Video completion is the task of filling space-time region with newly synthesized content.
- This task has many useful applications like removing watermarks, logos, etc. Though there can be some evil applications as well which we will discuss later, likely in the end.
- Video completion is challenging. Why? Not only you have to generate content that looks legit and embed seamlessly in the video, you also need to make sure that the results are temporally coherent i.e. the results shouldn't flicker and respect the dynamics of both the moving camera as well as the moving objects.

- Existing methods for video completion can be divided into three major categories: Patch based, Learning based, and Flow based
- Simply put, patch based methods are based on the idea of copying/pasting small video patches to the desired area in a greedy fashion. Though patches can provide rich information for encoding colours, textures, etc, these methods are really slow and have limited synthesis ability
- Learning based methods (you may have guessed GANs already), provide much reliable results compared to patch based methods but suffer from resolution issues.
- Flow based methods, like deep flow based DFCNet are most successful methods to date. They synthesize color and flow jointly.

as possible. This is challenging because we need to ensure that the result is temporally coherent (does not flicker) and respects dynamic camera motion as well as complex object motion in the video.

Up until a few years ago, most methods used patch-based synthesis techniques [14,26,39]. These methods are often slow and have limited ability to synthesize new content because they can only remix existing patches in the video. Recent learning-based techniques achieve more plausible synthesis [5, 38], but due to the high memory requirements of video, methods employing 3D spatial-temporal kernels suffer from resolution issues. The most successful methods to date [14, 42] are flow-based. They synthesize color and flow jointly and propagate color along flow trajectories to improve temporal coherence, which alleviates memory problems and enables high-resolution output. Our method also follows this general approach.

The key to achieving good results with the flow-based approach is accurate flow completion, in particular, synthesizing sharp *flow edges* along the object boundaries. However, the aforementioned methods are not able to synthesize sharp flow edges and often produce over-smoothed results. While this still works when removing *entire* objects in front of *flat* backgrounds, it breaks down in more complex situations. For example, existing methods have difficulty in completing *partially* seen *dynamic* objects well (Figure 1b–c). Notably, this situation is ubiquitous when completing *static screen-space masks*, such as logos or watermarks. In this work, we improve the flow completion by explicitly completing flow edges. We then use the completed flow edges to guide the flow completion, resulting in *piecewise-smooth* flow with sharp edges (Figure 1d).

Another limitation of previous flow-based methods is that chained flow vectors between adjacent frames can only form *continuous* temporal constraints. This prevents constraining and propagating to many parts of a video. For example, considering the situation of the periodic leg motion of a walking person: here, the background is repeatedly visible between the legs, but the sweeping motion prevents forming continuous flow trajectories to reach (and fill) these areas. We alleviate this problem by introducing additional flow constraints to a set of *non-local* (i.e., temporally distant) frames. This creates short-cuts across flow barriers and propagates color to more parts of the video.

Finally, previous flow-based methods propagate color values directly. However, the color often subtly changes over time in a video due to effects such as lighting changes, shadows, lens vignetting, auto exposure, and white balancing, which can lead to visible color seams when combining colors propagated from different frames. Our method reduces this problem by operating in the gradient domain.

In summary, our method alleviates the limitations of existing flow-based video completion algorithms through the following key contributions:

1. **Flow edges**: By explicitly completing flow edges, we obtain piecewise-smooth flow completion.
2. **Non-local flow**: We handle regions that cannot be reached through transitive flow (e.g., periodic motion, such as walking) by leveraging non-local flow.
3. **Seamless blending**: We avoid visible seams in our results through operating in the gradient domain.

Existing methods have several limitations:

1. The first limitation is that they cannot produce sharp flow edges along the object boundaries which is necessary for accurate flow completion
2. Second limitations is that the chained flow vectors between adjacent frames can only form continuous temporal constraints which isn't an ideal scenario for many cases.
3. Third, they propagate color values directly which is problematic because color depends on many external factors like lightning, shadows, and tend to change over time by these factors.

Proposed solutions for those limitations:

1. To overcome the first limitation by explicitly completing the flow edges
2. The authors also introduce another flow constraint to a set of non-local, temporally distant frames to address the second limitation.
3. To address the color propagation problem, the authors propose to operate in the gradient domain

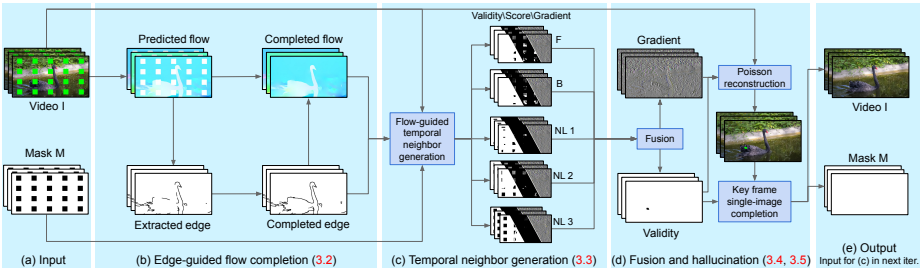


Fig. 2: **Algorithm overview.** (a) The input to our video completion method is a color video and a binary mask video that indicates which parts need to be synthesized. (b) We compute forward and backward flow between adjacent frames as well as a set of non-adjacent frames, extract and complete flow edges, and then use the completed edges to guide a piecewise-smooth flow completion (Section 3.2). (c) We follow the flow trajectories to compute a set of candidate pixels for each missing pixel. For each candidate, we estimate a confidence score as well as a binary validity indicator (Section 3.3). (d) We fuse the candidates in the gradient domain for each missing pixel using a confidence-weighted average. We pick a frame with most missing pixels and fill it with image inpainting (Section 3.4). (e) The result will be passed into the next iteration until there is no missing pixel (Section 3.5).

4. **Memory efficiency**: Our method handles videos with up to 4K resolution, while other methods fail due to excessive GPU memory requirements.

We validate the contribution of individual components to our results and show clear improvement over the prior methods in both quantitative evaluation and the quality of visual results.

2 Related Work

Image completion aims at filling missing regions in images with plausibly synthesized content. *Example-based* methods exploit the redundancy in natural images and transfer patches or segments from known regions to unknown (missing) regions [7, 9]. These methods find correspondences for content transfer either via patch-based synthesis [1, 39] or by solving a labeling problem with graph cuts [12, 32]. In addition to using only verbatim copied patches, several methods improve the completion quality by augmenting patch search with geometric and photometric transformations [8, 13, 15, 24]. *Learning-based* methods have shown promising results in image completion thanks to their ability to synthesize new content that may not exist in the original image [16, 29, 43, 45]. Several improved architecture designs have been proposed to handle free-form holes [23, 40, 44] and leverage predicted structures (e.g., edges) to guide the content [25, 33, 41]. Our work leverages a pre-trained image inpainting model [45] to fill in pixels that are not filled through temporal propagation.

Video completion inherits the challenges from the image completion problems and introduces new ones due to the additional time dimension. Below, we only discuss the video completion methods that are most relevant to our work. We refer the readers to a survey [17] for a complete map of the field.

Patch-based synthesis techniques have been applied to video completion by using 3D (spatio-temporal) patches as the synthesis unit [26, 39]. It is, however, challenging to handle dynamic videos (e.g., captured with a hand-held camera) with 3D patches, because they cannot adapt to deformations induced by camera motion. For this reason, several methods choose to fill the hole using 2D spatial patches and enforce temporal coherence with homography-based registration [10, 11] or explicit flow constraints [14, 34, 36]. In particular, Huang et al. [14] propose an optimization formulation that alternates between optical flow estimation and flow-guided patch-based synthesis. While the impressive results have been shown, the method is computationally expensive. Recent work [3, 28] shows that the speed can be substantially improved by (1) decoupling the flow completion step from the color synthesis step and (2) removing patch-based synthesis (i.e., relying solely on flow-based color propagation). These flow-based methods, however, are unable to infer sharp flow edges in the missing regions and thus have difficulties synthesizing dynamic object boundaries. Our work focuses on overcoming the limitations of these flow-based methods.

How does it work?

Limitations

Proposed solutions to the above limitations

Limitations of the proposed solutions

Driven by the success of learning-based methods for visual synthesis, recent efforts have focused on developing CNN-based approaches for video completion. Several methods adopt 3D CNN architectures for extracting features and learning to reconstruct the missing content [5, 38]. However, the use of 3D CNNs substantially limits the spatial (and temporal) resolution of the videos one can process due to the memory constraint. To alleviate this issue, the methods in [20, 22, 27] sample a small number of nearby frames as references. These methods, however, are unable to transfer temporally distant content due to the fixed temporal windows used by the method. Inspired by flow-based methods [3, 14, 28], Xu et al. [42] explicitly predict and complete dense flow field to facilitate propagating content from potentially distant frames to fill the missing regions. Our method builds upon the flow-based video completion formulation and makes several technical contributions to substantially improve the visual quality of completion, including completing edge-preserving flow fields, leveraging non-local flow, and gradient-domain processing for seamless results.

Limitations when using CNNs

Proposed solutions

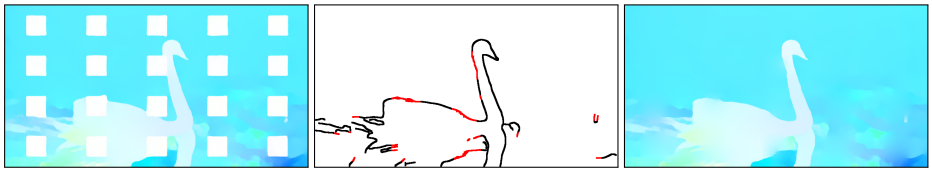
Limitations of the proposed solutions

Gradient-domain processing techniques are indispensable tools for a wide variety of applications, including image editing [2, 31], image-based rendering [21], blending stitched panorama [37], and seamlessly inserting moving objects in a video [6]. In the context of video completion, Poisson blending could be applied as a post-processing step to blend the synthesized content with the original video and hide the seams along the hole boundary. However, such an approach would not be sufficient because the propagated content from multiple frames may introduce visible seams *within* the hole that cannot be removed via Poisson blending. Our method alleviates this issue by propagating gradients (instead of colors) in our flow-based propagation process.

Simply put, operating on the gradients rather than the pixel values directly

Limitations of using Poisson blending for video completion tasks

Solution to overcome it: instead of colors, propagate gradients



(a) Input flow (b) Extracted/completed edges (c) Completed flow

Fig. 3: **Flow completion.** (a) Optical flow estimated on the input video. Missing regions tend to have zero value (white). (b) **Extracted** and **completed** flow edges. (c) Piecewise-smooth completed flow, using the edges as guidance.

3 Method

3.1 Overview

Inputs

The input to our video completion method is a color video and a binary mask video indicating which parts need to be synthesized (Figure 2a). We refer to the masked pixels as the *missing* region and the others as the *known* region. Our method consists of the following three main steps. **(1) Flow completion**: We first compute forward and backward flow between adjacent frames as well as a set of non-adjacent ("non-local") frames, and complete the missing region in these flow fields (Section 3.2). Since edges are typically the most salient features in flow maps, we extract and complete them first. We then use the completed edges to produce piecewise-smooth flow completion (Figure 2b). **(2) Temporal propagation**: Next, we follow the flow trajectories to propagate a set of candidate pixels for each missing pixel (Section 3.3). We obtain two candidates from chaining forward and backward flow vectors until a known pixel is reached. We obtain three additional candidates by checking three temporally distant frames with the help of non-local flow vectors. For each candidate, we estimate a confidence score as well as a binary validity indicator (Figure 2c). **(3) Fusion**: We fuse the candidates for each missing pixel with at least one valid candidate using a confidence-weighted average (Section 3.4). We perform the fusion in the gradient domain to avoid visible color seams (Figure 2d).

If there are still missing pixels after this procedure, it means that they could not be filled via temporal propagation (e.g., being occluded throughout the entire video). To handle these pixels, we pick a single key frame (with most remaining missing pixels) and fill it completely using a single-image completion technique (Section 3.5). We use this result as input for another iteration of the same process described above. The spatial completion step guarantees that we are making progress in each iteration, and its result will be propagated to the remainder of the video for enforcing temporal consistency in the next iteration. In the following sections, we provide more details about each of these steps.

How to handle pixels that are still missing even after these three steps?

3.2 Edge-guided Flow Completion

The first step in our algorithm is to compute optical flow between adjacent frames as well as between several non-local frames (we explain how we choose the set of

The first step Edge-guided Flow Completion consists of three steps:

- Flow computation: Flow are computed for local as well as non-local frames. For local frames, a pretrained network FlowNet2 is used to compute the flow (eqn 1) The same can't be used for non-adjacent frames as the estimates are very poor. To overcome this, first holography warp is used to tackle the large motion problem. This will give us the flow estimates in holographic aligned frames. But we need to estimate the flows in original frames, hence we add another inverse mapping (eqn 2).

non-local connections in Section 3.3) and to complete the missing regions in the flow fields in an edge-guided manner.

Flow computation. Let $\boldsymbol{I}_i$ and $\boldsymbol{M}_i$ be the color and mask of the i-th frame, respectively (we drop the subscript i if it is clear from the context), with $\boldsymbol{M}(p) = 1$ if pixel p is missing, and 0 otherwise.

We compute the flow between adjacent frames i and j using the pretrained FlowNet2 [18] network $\mathcal{F}$:

$$\boldsymbol{F}_{i\to j} = \mathcal{F}(\boldsymbol{I}_i, \boldsymbol{I}_j), \quad |i-j| = 1. \tag{1}$$

Flow estimate

White values in flow indicates the missing pixels in the video, denoted by black in the videos

Note that we set the missing pixels in the color video to black, but we do not treat them in any special way except during flow computation. In these missing regions, the flow is typically estimated to be zero (white in visualizations, e.g., in Figure 3a).

We notice that the flow estimation is *substantially degraded* or even *fails* in the presence of large motion, which frequently occurs in non-local frames. To alleviate this problem, we use a homography warp $\mathcal{H}_{j\to i}$ to compensate for the large motion between frame i and frame j (e.g., from camera rotation) before estimating the flow:

$$\boldsymbol{F}_{i\to j} = \mathcal{F}(\boldsymbol{I}_i, \mathcal{H}_{j\to i}(\boldsymbol{I}_j)) + \boldsymbol{H}_{i\to j}, \quad |i-j| > 1. \tag{2}$$

Flow estimates for non-adjacent or non-local frames

Since we are not interested in the flow between the homography-aligned frames but between the original frames, we add back the flow field $\boldsymbol{H}_{i\to j}$ of the *inverse* homography transformation, i.e., mapping each flow vector back to the original pixel location in the unaligned frame j. We estimate the aligning homography using RANSAC on ORB feature matches [35]. This operation takes about 3% of the total computational time.

Flow edge completion. After estimating the flow fields, our next goal is to replace missing regions with plausible completions. We notice that the influence of missing regions extends slightly outside the masks (see the bulges in the white regions in Figure 3a). Therefore, we dilate the masks by 15 pixels for flow completion. As can be seen in numerous examples throughout this paper, flow fields are generally piecewise-smooth, i.e., their gradients are small except along distinct motion boundaries, which are the most salient features in these maps. However, we observed that many prior flow-based video completion methods are unable to preserve sharp boundaries. To improve this, we first extract and complete the flow edges, and then use them as guidance for a piecewise-smooth completion of the flow values.

We use the Canny edge detector [4] to extract a flow edge map $\boldsymbol{E}_{i\to j}$ (Figure 3b, black lines). Note that we remove the edges of missing regions using the masks. We follow *EdgeConnect* [25] and train a flow edge completion network (See Section 4.1 for details). At inference time, the network predicts a completed edge map $\tilde{\boldsymbol{E}}_{i\to j}$ (Figure 3b, red lines).

Flow completion. Now that we have hallucinated flow *edges* in the missing region, we are ready to complete the actual flow *values*. Since we are interested in a smooth completion except at the edges, we solve for a solution that minimizes

- Flow-edge completion: once the flow has been computed, Canny detector is used to extract the edges. A network is then trained to predict and complete the missing edges. This helps in getting better flow values which is the last step
- Flow completion: To compute the actual flow values, the problem is stated as an optimization problem that minimizes the gradients everywhere except for the edges (Eqn 3). This problem is solved using a standard linear least-square solver.

the gradients everywhere (except at the edges). We obtain the completed flow $\tilde{\boldsymbol{F}}$ by solving the following problem:

$$\underset{\tilde{\boldsymbol{F}}}{\operatorname{argmin}} \sum_{p|\tilde{\boldsymbol{E}}(p)=1} \left\|\Delta_x \tilde{\boldsymbol{F}}(p)\right\|_2^2 + \left\|\Delta_y \tilde{\boldsymbol{F}}(p)\right\|_2^2, \quad \text{subject to} \quad \tilde{\boldsymbol{F}}(p) = \boldsymbol{F}(p) \mid \boldsymbol{M}(p) = 0, \tag{3}$$

where Δ_x and Δ_y respectively denote the horizontal and vertical finite forward difference operator. The summation is over all non-edge pixels, and the boundary condition ensures a smooth continuation of the flow outside the mask. The solution to Equation 3 is a set of sparse linear equations, which we solve using a standard linear least-squares solver. Figure 3c shows an example of flow completion.

3.3 Local and Non-local Temporal Neighbors

Now we can use the *completed* flow fields to guide the completion of the color video. This proceeds in two steps: for each missing pixel, we (1) find a set of known *temporal neighbor* pixels (this section), and (2) resolve a color by *fusing* the candidates using weighted averaging (Section 3.4).

The flow fields establish a connection between related pixels across frames, which are leveraged to guide the completion by propagating colors from known pixels through the missing regions along flow trajectories. Instead of *push*-propagating colors to the missing region (and suffering from repeated resampling), it is more desirable to transitively follow the forward and backward flow links for a given missing pixel, until known pixels are reached, and *pull* their colors.

We check the validity of the flow by measuring the forward-backward cycle consistency error,

$$\tilde{\boldsymbol{D}}_{i\to j}(p) = \left\|\boldsymbol{F}_{i\to j}(p) + \boldsymbol{F}_{j\to i}(p + \boldsymbol{F}_{i\to j}(p))\right\|_2^2, \tag{4}$$

and stop the tracing if we encounter an error of more than $\tau = 5$ pixels. We call the known pixels that can be reached in this manner *local* temporal neighbors because they are computed by chaining flow vector between adjacent frames.

Sometimes, we might not be able to reach a local known pixel, either because the missing region extends to the end of the video, because of invalid flow, or because we encounter a *flow barrier*. Flow barriers occur at every major motion boundary because the occlusion/dis-occlusion breaks the forward/backward cycle consistency there. A typical example is shown in Figure 4. Barriers can lead to large regions of *isolated* pixels without local temporal neighbors. Previous methods relied on hallucinations to generate content in these regions. However, hallucinations are more artifact-prone than propagation.

In particular, even if the synthesized content is plausible, it will most likely be different from the actual content visible *across* the barrier, which would lead to temporarily inconsistent results.

We alleviate this problem by introducing *non-local* temporal neighbors, i.e., computing flow to a set of temporally distant frames that short-cut across flow

- For color propagation for the missing pixel, we can leverage the connection established by the flow fields across different frames
- But instead of push propagating the color through the trajectories obtained above, we can use consistency cycle (eqn 4) to locate known pixels and pull their colors.
- As you might have guessed, it is not necessary that the flow is consistent and we always find the local pixels through this setup. Flow barriers break the cycle consistency and can lead to large regions of isolated pixels without any local temporal neighbors
- In that case, we can use eqn 2. to compute flow between non-local frames.

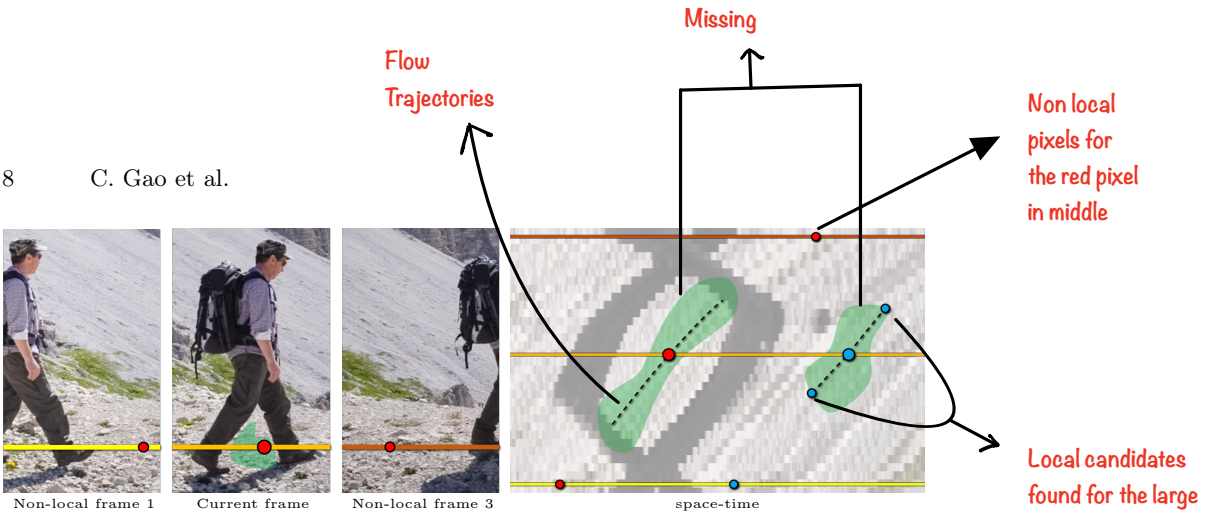


Fig. 4: **Non-local completion candidates.** The right figure shows a *space-time* visualization for the highlighted scanlines in the left images. Green regions are missing. The yellow, orange and brown line in the right subfigure represents the scanline at the first non-local frame, the current frame and the third non-local frame, respectively. The figure illustrates the completion candidates for the **red** and **blue** pixels (large discs on the orange line). By following the flow trajectories (dashed black lines) until the edge of the missing region, we obtain *local* candidates for the **blue** pixel (small discs), but not for the **red** pixel, because the sweeping legs of the person form impassable flow barriers. With the help of the non-local flow that connects to the temporally distant frames, we obtain extra *non-local* neighbors for the **red** pixel (red discs on the yellow and brown line). As a result, we can reveal the true background that is covered by the sweeping legs.

barriers, which dramatically reduces the number of isolated pixels and the need for hallucination. For every frame, we compute non-local flow to three additional frames using the homography-aligned method (Equation 2). For simplicity, we always select the first, middle, and last frames of the video as non-local neighbors. Figure 5 shows an example.

Discussion: We experimented with adaptive schemes for non-local neighbor selection, but found that the added complexity was hardly justified for the relatively short video sequences we worked with in this paper. When working with longer videos, it might be necessary to resort to more sophisticated schemes, such as constant frame offsets, and possibly adding additional non-local frames.

3.4 Fusing Temporal Neighbors

Now that we have computed temporal neighbors for the missing pixels, we are ready to fuse them to synthesize the completed color values. For a given missing pixel p, let $k \in N(p)$ be the set of valid local and non-local temporal neighbors (we reject neighbors with flow error exceeding τ, and will explain how to deal with pixels that have no neighbors in Section 3.5). We compute the completed color as a weighted average of the candidate colors c_k,

$$\tilde{\boldsymbol{I}}(p) = \frac{\sum_k w_k c_k}{\sum_k w_k}. \tag{5}$$

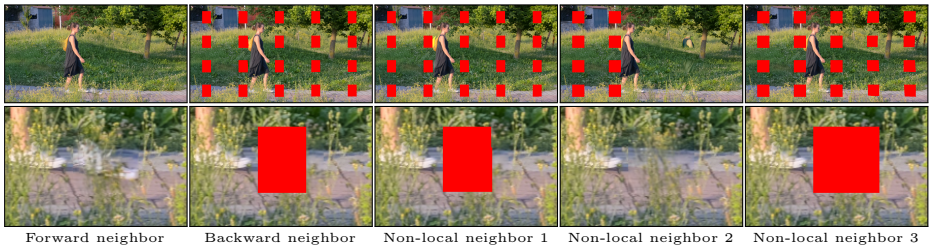
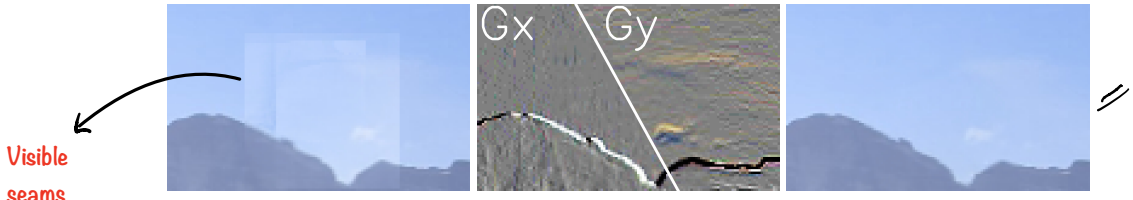


Fig. 5: **Temporal neighbors.** Non-local temporal neighbors (the second non-local neighbor in this case) are useful when correct known pixels cannot be reached with local flow chains due to flow barriers (**red**: invalid neighbors.)



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

Fig. 6: **Gradient domain reconstruction.** Previous methods operate directly in the color domain, which results in seams (a). We propagate in the gradient domain (b), and reconstruct the results by Poisson reconstruction (c).

The weights, w_k are computed from the flow cycle consistency error:

$$w_k = \exp(-d_k/T), \tag{6}$$

where d_k is the consistency error $\tilde{\boldsymbol{D}}_{i\to j}(p)$ for *non-local* neighbors, and the maximum of these errors along the chain of flow vectors for *local* neighbors. We set $T = 0.1$ to strongly down-weigh neighbors with large flow error.

Gradient-domain processing. We observed that directly propagating color values often yields visible seams, even with the correct flow. This is because of subtle color shifts in the input video (Figure 6a). These frequently occur due to effects such as lighting changes, shadows, lens vignetting, auto exposure, and white balancing, etc. We address this issue by changing Equation 5 to compute a weighted average of color *gradients*, rather than color values,

Solution for visible seams

$$\tilde{\boldsymbol{G}}_x(p) = \frac{\sum_k w_k \, \Delta_x c_k}{\sum_k w_k}, \quad \tilde{\boldsymbol{G}}_y(p) = \frac{\sum_k w_k \, \Delta_y c_k}{\sum_k w_k}, \tag{7}$$

and obtain the final image by solving a Poisson reconstruction problem,

$$\underset{\tilde{\boldsymbol{I}}}{\operatorname{argmin}} \left\| \Delta_x \tilde{\boldsymbol{I}} - \tilde{\boldsymbol{G}}_x \right\|_2^2 + \left\| \Delta_y \tilde{\boldsymbol{I}} - \tilde{\boldsymbol{G}}_y \right\|_2^2,$$

$$\text{subject to} \tilde{\boldsymbol{I}}(p) = \boldsymbol{I}(p) \mid M(p) = 0, \tag{8}$$

which can be solved using a standard linear least-squares solver. By operating in the gradient domain (Figure 6b), the color seams are suppressed (Figure 6c).

Why iterative completion is required?
Problem 1: While propagating colors, it might happen with non-local flow, some pixels won't have valid temporal neighbors.
Solution: Treat it as a single image completion task and use Deepfill to do it.
Problem 2: Completing other missing pixels in all frames using this method would result in temporal incoherency.
Solution: Select only that frame which has the highest number of missing pixels and complete it with single image completion method. For rest of the frames, this will be the input in the subsequent iterations where each iteration is over the whole pipeline, except for the flow computation step.

3.5 Iterative Completion

In each iteration, we propagate color gradients and obtain (up to) five candidate gradients. Then we fuse all candidate gradients and obtain missing pixel color values by solving a Poisson reconstruction problem (Equation 8). This will fill all the missing pixels that have valid temporal neighbors. Some missing pixels might not have any valid temporal neighbors, even with the non-local flow, which, for example, happens when the pixel is occluded in all non-local frames, or when the flow is incorrectly estimated. Similar to past work [14], we formulate this problem as a single-image completion task, and solve it with Deepfill [45]. However, if we would complete the remaining missing regions in *all* frames with this single-image method, the result would not be temporally coherent. Instead, we select only *one* frame with the most remaining missing pixels and complete it with the single-image method. Then, we feed the inpainting result as input to another iteration of our whole pipeline (with the notable exception of flow computation, which does not need to be recomputed). In this subsequent iteration, the single-image completed frame is treated as a known region, and its color gradients are coherently propagated to the surrounding frames.

The iterative completion process ends when there is no missing pixel. In practice, our algorithm needs around 5 iterations to fill all missing pixels in the video sequences we have tried. We have included the pseudo-code in the supplementary material, which summarizes the entire pipeline.

4 Experimental Results

4.1 Experimental setup

Scenarios. We consider two application scenarios for video completion: (1) screen-space mask inpainting and (2) object removal. For the inpainting setting, we generate a stationary mask with a uniform grid of 5×4 square blocks (see an example in Figure 7). This setting simulates the tasks of watermark or subtitle removal. Recovering content from such holes is particularly challenging because it often requires synthesizing *partially* visible dynamic objects over their background. For the object removal setting, we aim at recovering the missing content from a dynamically moving mask that covers the *entire* foreground object. This task is relatively easier because, typically, the dominant dynamic object is removed entirely. Results in the object removal setting, however, are difficult to compare and evaluate due to the lack of ground truth content behind the masked object. For this reason, we introduce a further *synthetic* object mask inpainting task. Specifically, we take a collection of free-form object masks and randomly pair them with other videos, pretending there is an object occluding the scene.

Evaluation metrics. For tasks where the ground truth is available (stationary mask inpainting and object mask inpainting), we quantify the quality of the completed video using PSNR, SSIM, and LPIPS [46]. For LPIPS, we follow the default setting; we use Alexnet as the backbone, and we add a linear calibration on top of intermediate features.

Table 1: **Video completion results with two types of synthetic masks.** We report the average PSNR, SSIM and LPIPS results with comparisons to existing methods on DAVIS dataset. The best performance is in **bold** and the second best is underscored. Missing entries indicate the method fails at the respective resolution.

	720 × 384 resolution						960 × 512 resolution					
	Stationary masks			Object masks			Stationary masks			Object masks		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Kim et al. [20]	25.19	0.8229	0.301	28.07	0.8673	0.283	-	-	-	-	-	-
Newson et al. [26]	27.50	0.9070	0.067	32.65	0.9648	0.023	-	-	-	-	-	-
Xu et al. [42]	27.69	0.9264	0.077	39.67	0.9894	0.008	27.17	0.9216	0.085	38.88	0.9882	0.009
Lee et al. [22]	28.47	0.9170	0.111	35.76	0.9819	0.021	28.08	0.9141	0.117	35.34	0.9814	0.022
Huang et al. [14]	28.72	0.9256	0.070	34.64	0.9725	0.018	-	-	-	-	-	-
Oh et al. [27]	30.28	0.9279	0.082	33.78	0.9630	0.058	-	-	-	-	-	-
Ours	**31.38**	**0.9592**	**0.042**	**42.72**	**0.9917**	**0.007**	**30.91**	**0.9564**	**0.048**	**41.89**	**0.9910**	**0.007**

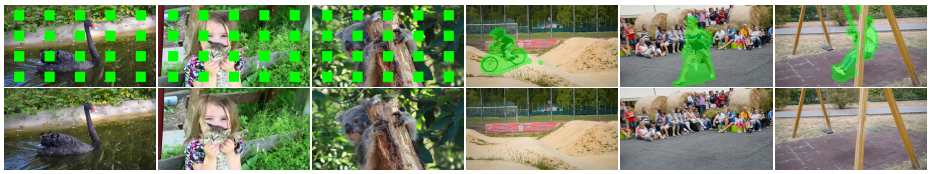


Fig. 7: **Qualitative results.** We show the results of stationary screen-space inpainting task (first three columns) and object removal task (last three columns).

Dataset. We evaluate our method on the DAVIS dataset [30], which contains a total of 150 video sequences. Following the evaluation protocol in [42], we use the 60 sequences in `2017-test-dev` and `2017-test-challenge` for training our flow edge completion network. We use the 90 sequences in `2017-train` and `2017-val` for testing the stationary mask inpainting task. For the object removal task, we test on the 29 out of the 90 sequences for which refined masks provided by Huang et al. [14] are available (these masks include shadows cast by the foreground object). For the object mask inpainting task, we randomly pair these 29 video sequences with mask sequences from the same set that have the same or longer duration. We resize the object masks by a uniform random factor in [0.8, 1], and trim them to match the number of frames. We resize all sequences to 960 × 512.

Implementation details. We build our flow edge completion network upon the publicly available official implementation of EdgeConnect [25][1]. We use the following parameters for the Canny edge detector [4]: Gaussian $\sigma = 1$, low threshold 0.1, high threshold 0.2. We run the Canny edge detector on the flow magnitude image. In addition to the mask and edge images, EdgeConnect takes a "grayscale" image as additional input; we substitute the flow magnitude image for it. We load weights pretrained on the Places2 dataset [47], and then finetune on 60 sequences in DAVIS `2017-test-dev` and `2017-test-challenge` for 3 epochs.

Backbone network

[1] https://github.com/knazeri/edge-connect

Input Diffusion Xu et al. [42] Ours

Fig. 8: **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.

We adopt masks from NVIDIA Irregular Mask Dataset testing split [2]. During training, we first crop the edge images and corresponding flow magnitude images to 256×256 patches. Then we corrupt them with a randomly chosen mask, which is resized to 256×256. We use the ADAM optimizer with a learning rate of 0.001. Training our network takes 12 hours on a single NVIDIA P100 GPU.

Reasonable

4.2 Quantitative evaluation

We report quantitative results under the stationary mask inpainting and object mask inpainting setting in Table 1. Because not all methods were able to handle the full 960×512 resolution due to memory constraint, we downscaled all scenes to 720×384 and reported numbers for both resolutions. Our method substantially improves the performance over state-of-the-art algorithms [14, 20, 22, 26, 27, 42] on the three metrics. Following [14], we also show the detailed running time analysis of our method in the supplementary material. We report the time for each component of our method on the "CAMEL" video sequence under the object removal setting. Our method runs at 7.2 frames per minute.

4.3 Qualitative evaluation

Figure 7 shows sample completion results for a diverse set of sequences. In all these cases, our method produces temporally coherent and visually plausible content. Please refer to the supplementary video results for extensive qualitative comparison to the methods listed in Table 1.

4.4 Ablation study

In this section, we validate the effectiveness of our design choices.

Gradient domain processing.

We compare the proposed gradient propagation process with color propagation (used in [14, 42]). Figure 6 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 6a), which are removed

[2] https://www.dropbox.com/s/01dfayns9s0kevy/test_mask.zip?dl=0

Without non-local neighbors With non-local neighbors

Fig. 9: **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.

Table 2: **Ablation study.** We report the average scores on DAVIS.

(a) **Domain and non-local**

Gradient	Non-local	Stationary masks			Object masks		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
-	-	28.28	0.9451	0.067	39.29	0.9893	0.009
-	✓	28.47	0.9469	0.069	39.67	0.9897	0.009
✓	-	30.78	0.9552	0.049	41.55	0.9907	0.007
✓	✓	30.91	0.9564	0.048	41.89	0.9910	0.007

(b) **Flow completion methods**

	Stationary masks				Object masks			
	Flow EPE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Flow EPE ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Diffusion	1.79	30.18	0.9526	0.049	0.04	41.12	0.9902	0.008
Xu et al. [42]	2.01	27.17	0.9216	0.085	0.26	38.88	0.9882	0.009
Ours	1.63	30.91	0.9564	0.048	0.03	41.89	0.9910	0.007

when operating in the gradient domain (Figure 6c). Table 2(a) analyzes the contribution of the gradient propagation quantitatively.

Non-local temporal neighbors. We study the effectiveness of the non-local temporal neighbors. Table 2(a) shows the quantitative comparisons. The overall quantitative improvement is somewhat subtle because, in many simple scenarios, the forward/backward flow neighbors are sufficient for propagating the correct content. In challenging cases, the use of non-local neighbors helps substantially reduce artifacts when both forward and backward (transitively connected) flow neighbors are incorrect due to occlusion or not available. Figure 9 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 8, we show two examples of flow completion results using diffusion (essentially Equation 3 without edge guidance), a trained flow completion network [42], and our proposed edge-guided flow completion. The diffusion-based method maximizes smoothness in the flow field everywhere and thus cannot create motion boundaries. The learning-based flow completion network [42] 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. Table 2(b) reports the endpoint error (EPE) between the pseudo ground truth flow (i.e., flow computed from the original, uncorrupted videos using FlowNet2) and the completed flow. The results show that the proposed flow completion achieves significantly lower EPE errors than diffusion and the trained flow completion network [42]. As a result, our proposed flow completion method helps improve the quantitative results.

Large stationary hole Semantic structure Fast motion

Fig. 10: **Failure cases.** Left, middle: hallucinated content in large missing regions (i.e., not filled by propagation) is sometimes not plausible. Right: fast motion might lead to poorly estimated flow, which results in a poor color completion.

4.5 Limitations

Failure results. Video completion remains a challenging problem. We show and explain several failure cases in Figure 10.

Processing speed. Our method runs at 0.12 fps, which is comparable to other flow-based methods. End-to-end models are relatively faster, e.g., Lee et al. [22] runs at 0.405 fps, but with worse performance. We acknowledge our slightly slower running time to be a weakness. 👌👌

4.6 Negative results

We explored several alternatives to our design choices to improve the quality of our video completion results. Unfortunately, these changes either ended up degrading performance or not producing clear improvement.

Flow completion network. As many CNN-based methods have shown impressive results on the task of image completion, using a CNN for flow completion seems a natural approach. We modified and experimented with several inpainting architectures, including partial conv [23] and EdgeConnect [25] for learning to complete the missing flow (by training on flow fields extracted from a large video dataset [19]). However, we found that in both cases, the network fails to generalize to unseen video sequences and produce visible seams along the hole boundaries.

Learning-based fusion. We explored using a U-Net based model for learning the weights for fusing the candidate (Section 3.4). Our model takes a forward-backward consistency error maps and the validity mask as inputs and predict the fusion weights so that the fused gradients are as similar to the ground truth gradients as possible. However, we did not observe a clear improvement from this learning-based method over the hand-crafted weights.

Final note: Remember that I said in the beginning that this can be used in an "evil" way?
Although this is a very rare example but it is completely legit IMO. Think of a video as the only evidence of a hate crime. What If someone uses it to remove the suspected person from that video?

Again, this is a very rare and highly unlikely scenario, I have to point it out because it was bugging my mind

References

1. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. In: ACM TOG (Proc. SIGGRAPH). vol. 28, p. 24 (2009)
2. Bhat, P., Zitnick, C.L., Cohen, M.F., Curless, B.: Gradientshop: A gradient-domain optimization framework for image and video filtering. ACM TOG (Proc. SIGGRAPH) **29**(2), 10–1 (2010)
3. Bokov, A., Vatolin, D.: 100+ times faster video completion by optical-flow-guided variational refinement. In: ICIP (2018)
4. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. pp. 679–698 (1986)
5. Chang, Y.L., Liu, Z.Y., Hsu, W.: Free-form video inpainting with 3d gated convolution and temporal patchgan. In: ICCV (2019)
6. Chen, T., Zhu, J.Y., Shamir, A., Hu, S.M.: Motion-aware gradient domain video composition. TIP **22**(7), 2532–2544 (2013)
7. Criminisi, A., Perez, P., Toyama, K.: Object removal by exemplar-based inpainting. In: CVPR (2003)
8. Darabi, S., Shechtman, E., Barnes, C., Goldman, D.B., Sen, P.: Image melding: Combining inconsistent images using patch-based synthesis. ACM TOG (Proc. SIGGRAPH) **31**(4), 82–1 (2012)
9. Drori, I., Cohen-Or, D., Yeshurun, H.: Fragment-based image completion. In: ACM TOG (Proc. SIGGRAPH). vol. 22, pp. 303–312 (2003)
10. Gao, C., Moore, B.E., Nadakuditi, R.R.: Augmented robust pca for foreground-background separation on noisy, moving camera video. In: 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP) (2017)
11. Granados, M., Kim, K.I., Tompkin, J., Kautz, J., Theobalt, C.: Background inpainting for videos with dynamic objects and a free-moving camera. In: ECCV (2012)
12. He, K., Sun, J.: Image completion approaches using the statistics of similar patches. TPAMI **36**(12), 2423–2435 (2014)
13. Huang, J.B., Kang, S.B., Ahuja, N., Kopf, J.: Image completion using planar structure guidance. ACM TOG (Proc. SIGGRAPH) **33**(4), 129 (2014)
14. Huang, J.B., Kang, S.B., Ahuja, N., Kopf, J.: Temporally coherent completion of dynamic video. ACM Transactions on Graphics (TOG) (2016)
15. Huang, J.B., Kopf, J., Ahuja, N., Kang, S.B.: Transformation guided image completion. In: ICCP (2013)
16. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM TOG (Proc. SIGGRAPH) **36**(4), 107 (2017)
17. Ilan, S., Shamir, A.: A survey on data-driven video completion. In: Computer Graphics Forum. vol. 34, pp. 60–85 (2015)
18. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: CVPR (2017)
19. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al.: The kinetics human action video dataset. arXiv preprint arXiv:1705.06950 (2017)
20. Kim, D., Woo, S., Lee, J.Y., Kweon, I.S.: Deep video inpainting. In: CVPR (2019)
21. Kopf, J., Langguth, F., Scharstein, D., Szeliski, R., Goesele, M.: Image-based rendering in the gradient domain. ACM TOG (Proc. SIGGRAPH) **32**(6), 199 (2013)

22. Lee, S., Oh, S.W., Won, D., Kim, S.J.: Copy-and-paste networks for deep video inpainting. In: ICCV (2019)
23. 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 (2018)
24. Mansfield, A., Prasad, M., Rother, C., Sharp, T., Kohli, P., Van Gool, L.J.: Transforming image completion. In: BMVC (2011)
25. Nazeri, K., Ng, E., Joseph, T., Qureshi, F., Ebrahimi, M.: Edgeconnect: Generative image inpainting with adversarial edge learning. In: ICCVW (2019)
26. Newson, A., Almansa, A., Fradet, M., Gousseau, Y., Pérez, P.: Video inpainting of complex scenes. SIAM Journal on Imaging Sciences (2014)
27. Oh, S.W., Lee, S., Lee, J.Y., Kim, S.J.: Onion-peel networks for deep video completion. In: ICCV (2019)
28. Okabe, M., Noda, K., Dobashi, Y., Anjyo, K.: Interactive video completion. IEEE computer graphics and applications (2019)
29. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: CVPR (2016)
30. Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: CVPR (2016)
31. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. ACM TOG (Proc. SIGGRAPH) **22**(3), 313–318 (2003)
32. Pritch, Y., Kav-Venaki, E., Peleg, S.: Shift-map image editing. In: ICCV (2009)
33. Ren, Y., Yu, X., Zhang, R., Li, T.H., Liu, S., Li, G.: Structureflow: Image inpainting via structure-aware appearance flow. In: CVPR. pp. 181–190 (2019)
34. Roxas, M., Shiratori, T., Ikeuchi, K.: Video completion via spatio-temporally consistent motion inpainting. IPSJ Transactions on Computer Vision and Applications (2014)
35. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: ICCV (2011)
36. Strobel, M., Diebold, J., Cremers, D.: Flow and color inpainting for video completion. In: German Conference on Pattern Recognition (2014)
37. Szeliski, R., Uyttendaele, M., Steedly, D.: Fast poisson blending using multi-splines. In: ICCP. pp. 1–8 (2011)
38. Wang, C., Huang, H., Han, X., Wang, J.: Video inpainting by jointly learning temporal structure and spatial details. In: AAAI (2019)
39. Wexler, Y., Shechtman, E., Irani, M.: Space-time completion of video. TPAMI (3), 463–476 (2007)
40. Xie, C., Liu, S., Li, C., Cheng, M.M., Zuo, W., Liu, X., Wen, S., Ding, E.: Image inpainting with learnable bidirectional attention maps. In: ICCV (2019)
41. Xiong, W., Yu, J., Lin, Z., Yang, J., Lu, X., Barnes, C., Luo, J.: Foreground-aware image inpainting. In: CVPR (2019)
42. Xu, R., Li, X., Zhou, B., Loy, C.C.: Deep flow-guided video inpainting. In: CVPR (2019)
43. Yan, Z., Li, X., Li, M., Zuo, W., Shan, S.: Shift-net: Image inpainting via deep feature rearrangement. In: ECCV (2018)
44. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. arXiv preprint arXiv:1806.03589 (2018)
45. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Generative image inpainting with contextual attention. In: CVPR (2018)
46. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)

47. 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)