Error Compensation Framework for Flow-Guided Video Inpainting

Jaeyeon Kang[1], Seoung Wug Oh[2], and Seon Joo Kim[1]

[1]Yonsei University, [2]Adobe

Abstract. The key to video inpainting is to use correlation information from as many reference frames as possible. Existing flow-based propagation methods split the video synthesis process into multiple steps: flow completion → pixel propagation → synthesis. However, there is a significant drawback that the errors in each step continue to accumulate and amplify in the next step. To this end, we propose an Error Compensation Framework for Flow-guided Video Inpainting (ECFVI), which takes advantage of the flow-based method and offsets its weaknesses. We address the weakness with the newly designed flow completion module and the error compensation network that exploits the error guidance map. Our approach greatly improves the temporal consistency and the visual quality of the completed videos. Experimental results show the superior performance of our proposed method with the speed up of ×6, compared to the state-of-the-art methods. In addition, we present a new benchmark dataset for evaluation by supplementing the weaknesses of existing test datasets.

Keywords: Video Inpainting, Object Removal, Video Restoration

1 Introduction

Video inpainting is the task of filling missing regions in frames with realistic content. Its applications cover object removal, watermark/logo/subtitle removal, and even corrupted video restoration. It is challenging as the hole regions should be filled with synthesized contents that are unnoticeable with the surrounding background and temporally consistent.

Existing video inpainting methods usually extract useful information from reference frames and merge them to fill the target frame. The key is to use as many frames as possible since most correlated pixels are somewhere in other frames. The common pipeline for video inpainting can be categorized into two groups: 1) direct synthesis, 2) flow-based propagation.

In general, direct synthesis methods adopt convolution-based [3, 4, 7, 23] and attention-based [13, 22] networks. They usually take corrupted frames and corresponding pixel-wise masks as input and directly output completed frames. However, due to the computational complexity, these methods have relatively small temporal windows resulting in only a few reference frames to be used (up to 10 frames). Serious temporal inconsistencies can occur if some key frames

(a) Small temporal window issue in direct synthesis methods

(b) Pixel misalignment and brightness inconsistency issues in flow-based methods

Fig. 1: Qualitative comparison with other video inpainting methods on object removal scenarios. (a) The rear camel is originally shown in all frames, but it disappears in the result of [22, 23] due to the small temporal window. (b) Wrong pixels are propagated, and the brightness inconsistency issue occurred in [6, 20].

that contain unique pixel values and textures needed to fill the hole are not selected. Fig. 1(a) illustrates failure cases of [13, 22] when an occluded object is not properly included in the small reference frame set.

Flow-based methods [6, 20] split the synthesis process as into three steps: flow completion, pixel propagation, and synthesis. During the flow completion, the corrupted optical flow maps are processed by inpainting the holes within the flow maps. In the pixel propagation step, pixel values from reference frames are propagated to the holes with the guidance of the inpainted flows. Finally, in the synthesis step, only the remaining holes are synthesized using an image inpainting method. Flow-based approaches show better long-term temporal consistency compared to direct synthesis methods as they explicitly propagate pixel values from other frames using flows.

Nevertheless, there are some drawbacks in flow-based methods. *(1) Flow completion.* In flow inpainting, it is crucial to infer spatio-temporal relationships from other corrupted flows. However, even adjacent flows can have different values and directions due to inaccurate flow estimation or non-linear motion between frames, resulting in temporally inconsistent flows. Also, without taking the underlying video contents, errors from the flow estimator like *From corrupted* in Fig. 2 cannot be handled. *(2) Pixel propagation.* Simply copy-and-pasting propagated pixels causes pixel misalignment and brightness inconsistency. As the pixel values/textures can be conveyed from distant frames, there are often perspective mismatch such as scale and shape. In addition, the brightness inconsistency may arise due to the changes in the lighting condition or camera exposure in the given video sequence. Models will output visual artifacts without the proper compensation for these errors. While existing methods attempt to address these issues

with checking flow consistency [6, 20] and poisson blending [6], visual artifacts still remain as shown in Fig. 1(b).

To this end, we propose a simple yet effective Error Completion Framework for Flow-guided Video Inpainting (ECFVI) that addresses both drawbacks. We follow the three steps in flow-based methods (i.e., flow completion, pixel propagation, synthesis) but significantly improve the first two steps by correcting errors, preventing the error accumulation. For the flow completion, we make the flow inpainting be aware of RGB values. Hallucinating missing flow values based on the underlying video contents can infer temporally coherent flows. Specifically, we first roughly complete RGB pixels on holes, considering the local temporal relationship. Then, the locally coherent RGB inpainting results guide the flow inpainting. Note that the local RGB inpainting here is only used for guiding the flow, and the actual inpainting results are obtained in later steps. For the pixel propagation step, we introduce additional compensation network to detect and correct the misalignment and brightness inconsistency errors from the pixel propagation. The network utilizes the error map outside the hole (called error guidance map) to predict errors inside the hole. Such carefully designed components can prevent errors from accumulating at the following stages, which significantly improves the visual quality and temporal consistency of completed videos.

For quantitative evaluation, many video inpainting methods construct their own datasets by randomly masking regions on a video. They usually set the unmasked video as the ground-truth and evaluate their models. However, when the randomly generated mask covers an entire object like in Fig. 6(a), a model will output object-removed results which are different from the unmasked video. Evaluation values from this case are not proportional to human perception. Therefore, we propose a new benchmark dataset by using an object segmentation algorithm [5] so that the mask only partially covers the object. The results on our dataset are identical to the unmasked video, which can provide comparative analysis into video inpainting methods.

In summary, the contributions of the paper are as follows:

- We propose a simple yet effective way to compensate for the limitations of the flow-based video inpainting. Specifically, we improve the flow completion with RGB-awareness and propose to compensate errors in pixel propagation.
- Our method outperforms previous state-of-the-art video inpainting methods in terms of PSNR/SSIM and visual quality. Compared with the state-of-the-art in flow-based method FGVC [6], we produce better results while reducing the computation time (×6 faster).
- We provide a new benchmark dataset for quantitative evaluation of video inpainting, which can be very useful for evaluating future work on this topic by providing a common ground.

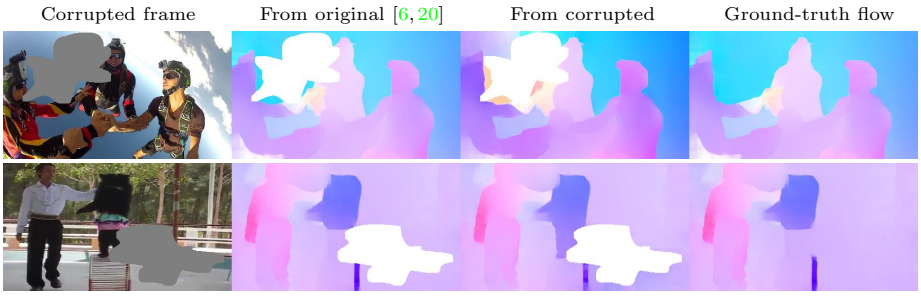


Fig. 2: This figure shows the corrupted flows with different settings. In [6, 20], they take *original* frames as input to the flow estimator and remove the values in the hole. However, if the original frames are unavailable, errors can occur when the corrupted frames are used as input. The flow values near the hole are different from the ground-truth flow.

2 Related Work

2.1 Direct synthesis methods

From the success of deep learning, many deep-based video inpainting methods have emerged. [3, 10, 17] proposed to use 3D encoder-decoder networks for enhancing the efficiency and the temporal consistency. [13, 22] exploited attention-based methods, which use transformer modules for matching similar patches on inter-intra frames. Zou et al. [23] used the optical flow to progressively merge target frames with reference frames to enrich feature representations on temporal relationships. Due to the memory constraints, above methods can only refer to a few frames. On the other hand, Oh et al. [14] exploited a non-local pixel matching to have a global temporal window by using the memory network. Although they refer to all frames, corruption of information occurs because the frames are continuously referenced in an implicit way. While Lee et al. [12] designed a network to take information from long-distance frames with global affine matrices, it cannot handle non-rigid complex motions.

2.2 Flow-based methods

Huang et al. [8] adopted spatial patch matching and flow-based method but suffered from mismatching issues caused by corrupted flows. Instead, [6, 19] first inpainted the corrupted flows and propagated pixels along their flow trajectories. Despite the success, they still have some limitations to produce better results: First, to estimate the corrupted flows, they entered *original* frames to the flow estimator and removed the values with masks. This procedure works well in object removal scenarios where the original frames exist. The problem is when the original frames are not available as in video restoration scenarios. If the flow estimator takes the corrupted frames as input, the masks can interfere with

estimating the motion of objects in the video (Fig. 2). The errors in corrupted flows will affect the flow completion stage, resulting in incorrect flow values. Second, it is difficult to use spatio-temporal information between the corrupted flows for flow completion as mentioned in previous section.

Wrong estimation from the flow completion would lead to the pixel misalignment issue. Both methods checked the flow consistency and only propagated trustworthy pixels to deal with the issue, but the consistency is not well preserved as the completed flow itself is incorrect. Our framework focuses on offsetting these limitations of flow-based methods.

3 Method

Let $X = [x_1, x_2, ..., x_T]$ be a set of corrupted video frames of length T. $M = [m_1, m_2, ..., m_T]$ denotes the corresponding frame-wise masks, where the spatial resolution is the same as X. For each mask, value 0 indicates valid regions, and 1 indicates hole (missing) regions. Video inpainting aims to predict the original or object removed video $\hat{Y} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_T]$ given X and M as inputs. We call the frame to be inpainted as the target frame and other frames as reference frames.

3.1 Overview

We illustrate our video inpainting framework in Fig. 3. The overall procedure can be summarized as follows. **(1) Flow completion with RGB guidance:** Given the corrupted frames, we first generate locally coherent frames. Then, we estimate complete flows between adjacent frames using our flow estimator (Sec. 3.2). Before the propagation, we estimate bi-directional completed flows between all frames. **(2) Pixel Propagation with Error Compensation:** With the guidance of completed flows, we propagate valid pixels from reference frames to target holes. In this process, we prevent the errors from propagating to the following stages using our error compensation network (Sec 3.3). In the order close to the target frame, we iterate propagation and compensation procedures until the hole regions are entirely removed or all frames are referenced. **(3) Synthesis:** If there are remaining holes to fill, we synthesize using an existing video inpainting method. This case usually occurs due to occlusion, where some pixels cannot be found in any other frames. For the synthesis, we used FuseFormer [13] with the weights from their website.

3.2 Flow completion with RGB guidance

Previous methods [6,20] first estimate corrupted flows with the flow estimator [9, 16] and complete flow values with their methods. However, without considering the original video contents, there is no ability to handle the errors from the corrupted flows. Therefore, we design our flow completion module to be aware of RGB values and directly output completed flows from the flow estimator.

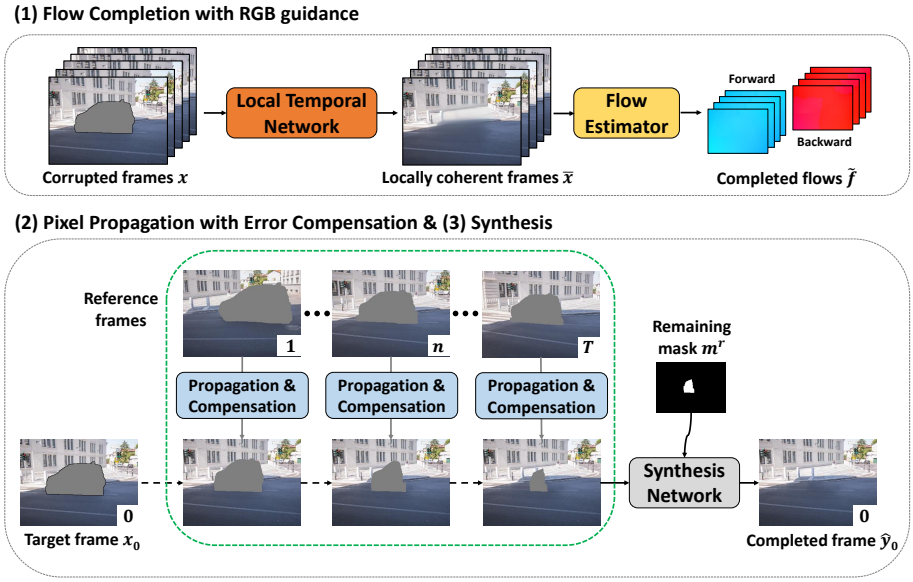


Fig. 3: Overview of our Error Compensation Framework for Flow-guided Video Inpainting (ECFVI). At first, we estimate all completed flows between adjacent frames using our flow completion module. With the guidance of the completed flows, we iteratively propagate and compensate pixels from reference frames to fill target holes. One iteration of the process is shown in Fig. 4. The remaining holes are further synthesized with the existing video inpainting method.

A simple approach would be to have the flow estimator take the corrupted frames and complete the flow values. However, since the flow estimator [9, 16] iteratively takes two frames to estimate entire flows, completed flows are created without considering spatio-temporal information of other flows. Also, it is more challenging to run flow estimation and completion simultaneously. Therefore, we roughly complete RGB pixels through local neighboring frames (larger than two frames) and pass the results to our flow estimator. This can further exploit temporal information for estimating completed flows. Here, our flow estimator takes locally coherent frames $\bar{x}$ from a local temporal network LTN. Note that the intermediate frames $\bar{x}$ are only used for guiding the flow completion.

We design the local temporal network LTN to consist of an encoder, multiple spatio-temporal transformer layers [13, 22] and a decoder. We first extract features of each input from the encoder and exploit transformer layers to merge the information from the reference in the deep encoding space. The decoder takes the output of the transformer layers to reconstruct locally coherent frames $\bar{x}$ as follows:

$$\bar{x}_i = LTN(x_{i-N:i+N}, m_{i-N:i+N}), \tag{1}$$

where N is the temporal radius and $N = 5$ is used in our experiment. Then, we estimate a completed flow between adjacent frames as follows:

$$\tilde{f}_{t\to t+1} = F(\bar{x}_t, \bar{x}_{t+1}, m_t, m_{t+1}), \tag{2}$$

where F is our flow estimator. Backward flow $\tilde{f}_{t\to t-1}$ can be estimated in the same manner. For classifying the regions where the original and coarsely completed content exist, we use masks m as additional input. The flow estimator is initialized with pretrained weights from RAFT [16] except for the first layer to take the masks as additional input. We jointly train our local temporal network and flow estimator as follows:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_{flow}\mathcal{L}_{flow}, \tag{3}$$

where $\mathcal{L}_{rec}$ is the reconstruction L1 loss defined as $||\bar{x} - y||_1$. λ_{flow} is a coefficient for the loss terms and we set it to 2 in, our experiment. For the flow loss $\mathcal{L}_{flow}$, inspired by [15], we employ hard example mining mechanism to weigh more on the difficult areas as follows:

$$\mathcal{L}_{flow} = ||h_t \odot (\tilde{f}_{t\to t+1} - f_{t\to t+1})||_1, \tag{4}$$

where $\odot$ is element-wise multiplication and ground-truth $f_{t\to t+1}$ is calculated from original frames Y. We set the hard mining weight h_t with $(1 + |\bar{x}_t - y_t|)^2$. It encourages the model to implicitly deal with errors from the local temporal network. More details of our network are shown in the supplementary material.

This design choice has advantages in two aspects. First, it alleviates the problems of corrupted flows mentioned in Sec.2.2. Second, with the strong initialization of pretrained weights and no need to inpaint flow values from scratch, it can be trained much easier. Although there still could be errors from the two networks, we can further compensate it in the next stage.

3.3 Pixel Propagation with Error Compensation

The next step, pixel propagation with error compensation, is illustrated in Fig. 4. With the guidance of the completed flows, we iterate propagation and compensation steps so that the errors do not accumulate or amplify in the next iteration.

Propagation. Let x_i and x_j be the target and the reference frame respectively. With the completed flow $\tilde{f}_{i\to j}$, only the valid pixels in reference frame x_j are forwarded to target holes by using backward warping function w.

$$\begin{aligned} m_i^p &= m_i \odot (1 - w(m_j, \tilde{f}_{i\to j})), \\ \tilde{x}_i &= x_i + m_i^p \odot w(x_j, \tilde{f}_{i\to j}), \end{aligned} \tag{5}$$

where $\tilde{x}_i$ is the filled target frame. To find the regions where the matched and valid regions are in the reference frame, we warp the reference mask m_j and get valid regions $1 - w(m_j, \tilde{f}_{i\to j})$ in the warped reference frame. m_i^p denotes a propagation mask where the propagated pixels exist, shown by the green region in

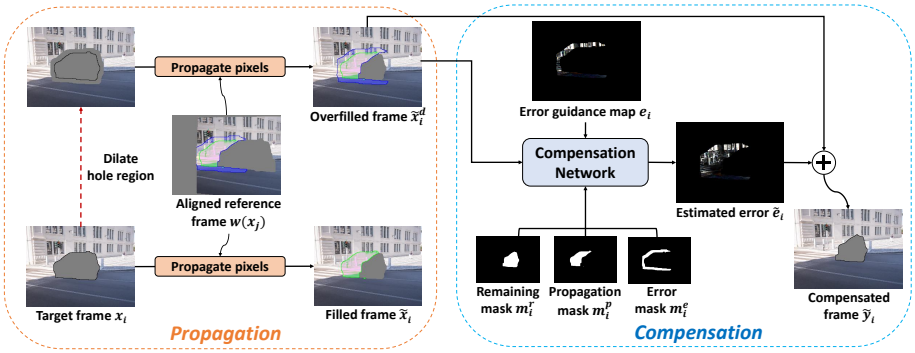


Fig. 4: For simplicity, we show the propagation and compensation stages from one reference frame x_j to the target frame x_i. We dilate the original hole region to get the overfilled frame $\tilde{x}_i^d$. Pixels corresponding to the blue region are further propagated from the aligned reference frame. The filled frame $\tilde{x}_i$ and overfilled frame $\tilde{x}_i^d$ are used to estimate the error guidance map e_i. Note that the green and blue regions correspond to propagation and error mask respectively.

Fig. 4. This mask is used at the compensation stage to know where to compensate for errors. After the propagation, there may be a remaining mask m_i^r, where $m_i^r = m_i - m_i^p$. The remaining mask is further used for the next propagation or input for the synthesis network.

Compensation. As shown in Fig. 5, we observe that directly propagating pixels from the reference frame may cause misalignment or brightness inconsistency issues. To remedy these issues, one solution is taking filled frame $\tilde{x}_i$ and corresponding mask m_i^p as input and processing them through a GAN framework. However, it is difficult to train such a network because it cannot easily detect what types of problem occurred.

We approach this problem by introducing the error guidance map e_i as input to our network. To know what is wrong with the propagation, we need valid (ground-truth) values. Since only regions outside the holes in the given corrupted frame have valid values, we propagate more pixels by dilating the original hole regions like in Fig. 4. Then, we calculate the errors on enlarged parts, which is the error guidance map e_i. This process assumes that the propagated pixels on the enlarged and the propagated regions have similar error tendencies. We set the dilation factor as 17 pixels in our experiment.

We set dilated mask as m_i^d and overfilled frame as $\tilde{x}_i^d$. The error guidance map e_i and the corresponding error regions (mask) m_i^e are computed as follows:

$$\begin{aligned} m_i^e &= (m_i^d - m_i) \odot (1 - w(m_j, \tilde{f}_{i \to j})), \\ e_i &= m_i^e \odot (\tilde{x}_i^d - \tilde{x}_i). \end{aligned} \tag{6}$$

In Fig. 5, this error guidance map intuitively shows the problems of pixel propagation. If it has no issue, it will be black (zero). On the other hand, if it has a

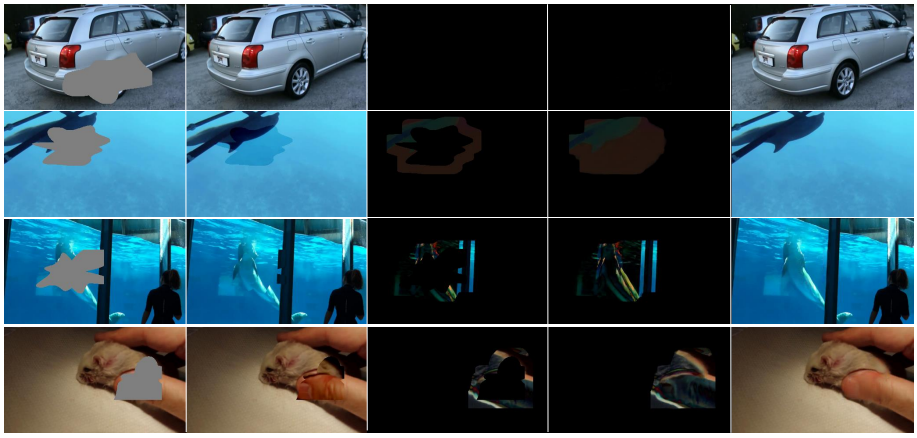


Fig. 5: Problems from the propagation stage and corresponding error maps. From top to bottom: no issue, brightness inconsistency, pixel misalignment, and combined issues. The estimated errors and compensated frames are computed using our method.

pixel misalignment issue, it will result in the edges being visible in the error map. This additional information helps our network be more aware of the problems from the propagation stage.

We design our error compensation network with a similar structure as our local temporal network LTN, but with different input and output settings:

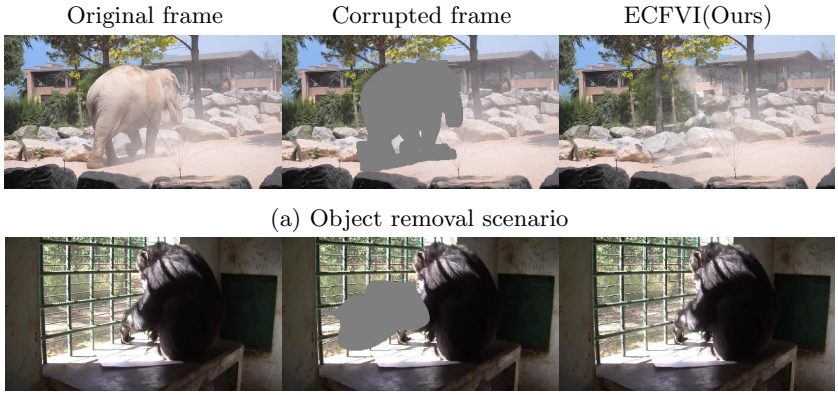
$$\tilde{e}_i = ECN(\tilde{x}^d_{i-N:i+N}, e_{i-N:i+N}, m^{[e,p,r]}_{i-N:i+N}), \tag{7}$$

where ECN is our error compensation network and $m^{[e,p,r]}$ denotes error, propagation, remaining mask respectively. For simplicity, we will omit the index from now on. $\tilde{x}^d, e, m^e$ are concatenated in the channel dimension and forwarded to our encoder, and each of the two masks m^p, m^r are used at the transformer layers. Note that the overfilled frames $\tilde{x}^d$ are used instead of filled frames $\tilde{x}$ as input. It makes our network to learn the relationship between overfilled frame $\tilde{x}^d$ and the error guidance maps e on the error regions m^e. Since the propagated regions have some structure information, based on the relationship, our network can estimate error values on the regions m^p with accuracy. A final compensated frame is $\tilde{y} = \tilde{x}^d + \tilde{e}$. The results and corresponding errors are shown in Fig. 5.

The loss to train our error compensation network consists of reconstruction loss $\mathcal{L}_{rec}$ and adversarial loss $\mathcal{L}_{adv}$:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}, \tag{8}$$

where λ_{adv} is a coefficient for the loss terms and we set 0.01 in our experiment. $\mathcal{L}_{rec}$ is a L1 loss between the ground-truth frame y_i and compensated output



(a) Object removal scenario

(b) Video restoration scenario

Fig. 6: Two common scenarios for video inpainting: object removal and video restoration. The gray region in the corrupted frame denotes a mask. In the object removal scenario, the result and the original frame are different. Only the video restoration scenario should be used for qualitative evaluation.

$\tilde{y}_i$. For $\mathcal{L}_{adv}$, we use the model from Temporal PatchGAN (T-PatchGAN) [3]. Further details on our network are included in the supplementary material.

4 Experiments

4.1 Training Details

Our whole networks are trained on Youtube-VOS [19] dataset. It consists of 4,453 videos, which are split into 3,471/474/508 for training, validation and testing respectively. We randomly select 256×256 frames and free-form masks from [22] as input. To train our compensation network, we freeze the weights of flow completion. And for handling the brightness inconsistency issue, we further modify the brightness, hue and saturation of reference frames. Adam optimizer [11] is used where the learning rate starts with 1e-4 and is divided by 2 every 40,000 iterations. For each module, the total iteration is 120,000 with mini-batch size 4, and it takes about 2 days using two NVIDIA RTX 2080 TI GPUs.

4.2 Youtube-VI Dataset

Most previous works created their own random masks and measured the performance on Youtube-VOS [19] and DAVIS [2] datasets. However, when a randomly generated mask coincidentally covers an entire object, a model will create the object-removed result like in Fig. 6(a). The result is reasonable, but it is far from the original frame. Only the case in Fig. 6(b) should be used for quantitative evaluation. Also, if there is no motion in the mask and its vicinity, it is not suitable for video inpainting, which cannot trace any pixels from other frames.

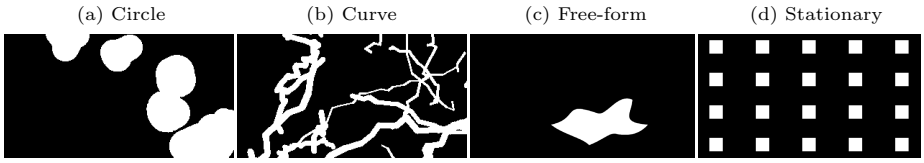


Fig. 7: Four different types of mask. In FVI [23] dataset, they use (a),(b) as a moving mask and (c) as a stationary mask. In our Youtube-VI dataset, (c) and (d) is used as a moving mask and stationary mask respectively.

TSAM [23] publicly provides frame-wise masks for the FVI [21] dataset. It consists of 100 scenes, each with 15 frames. For each scene, three types of masks are provided, some of which are shown in Fig. 7. While the masks cover diverse scenarios, 15 frames in a video are still limited compared to real-world videos. Also, the problems mentioned above on the mask still exist.

To this end, we design the new Youtube Video Inpainting (Youtube-VI) dataset, which will be publicly available. 100 scenes are selected in order of name from the beginning of Youtube-VOS dataset, and each scene has 50 frames. We generated two types of masks, which are shown in Fig. 7 and described as follows:
(1) Moving mask. We follow the same rules for generating free-form mask in STTN [22]. But we use a video segmentation network STCN [5] so that an object is not fully covered. With the help of STCN, we randomly generate one moving mask that partially overlaps with objects. Then we modify the mask again if the scene and mask do not look visually proper.
(2) Stationary mask. This case is usually used to cover the logo/subtitle removal scenarios. We find that using free-form masks is unsuitable since there may be no motion near the mask. Instead, we use 5x4 small square masks like FGVC [6].

4.3 Comparisons

We quantitatively compare with others on published FVI [23] and our new dataset for video restoration scenarios. Like [13,22,23], we use DAVIS [2] dataset for qualitative comparison in object removal scenarios. DAVIS dataset consists of 150 videos in total, in which 90 videos are annotated with the pixel-wise object masks. In addition, DAVIS-shadow dataset [8] where the mask covers both an object and its shadows are used for more visual comparison.

DFC-Net [20] and FGVC [6] only take original frames as input to the flow estimator from their published code. It is acceptable in object removal scenarios, but it is not fair to evaluate quantitative performance where the original frame is equal to ground-truth. Therefore, we also experiment with corrupted frames as input, and denote them as DFC-Net* and FGVC* respectively.

Quantitative Results. We evaluate video inpainting quality on PSNR and SSIM. In Table 1, we achieve the best performance on PSNR and SSIM except for the curve mask on FVI dataset. It is worth noting that the curve mask is a particular case in real-world scenarios. As we assume our error guidance map for

	FVI [23]						Youtube-VI				
	Stationary Mask		Circle Mask		Curve Mask		Stationary Mask		Moving Mask		
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	Time(s)
DFC-Net [20]	29.20	0.9632	24.77	0.9219	28.03	0.9436	29.03	0.9609	32.94	0.9771	95
DFC-Net* [20]	28.51	0.9609	24.12	0.9141	26.60	0.9295	27.73	0.9536	32.14	0.9747	95
OPN [14]	31.59	0.9689	26.37	0.9264	29.79	0.9519	31.91	0.9721	33.56	0.9767	154
CPN [12]	31.76	0.9693	26.90	0.9315	31.09	0.9664	31.32	0.9687	34.15	0.9783	24
STTN [22]	32.18	0.9699	27.04	0.9288	30.01	0.9515	31.96	0.9716	34.66	0.9789	17
FGVC [6]	30.88	0.9690	26.14	0.9369	**31.50**	**0.9676**	32.25	0.9750	35.68	0.9841	734
FGVC* [6]	30.60	0.9682	25.55	0.9302	29.90	0.9565	31.21	0.9701	34.05	0.9795	734
TSAM [23]	31.74	0.9695	26.66	0.9250	31.18	0.9578	-	-	-	-	-
FuseFormer [13]	33.19	0.9733	27.85	0.9368	30.85	0.9577	33.04	0.9755	35.29	0.9806	18
ECFVI(Ours)	**33.27**	**0.9749**	**28.24**	**0.9469**	31.15	0.9638	**33.53**	**0.9795**	**36.80**	**0.9860**	121

Table 1: Quantitative evaluation on FVI and Youtube-VI datasets. Best values are shown in bold and second best values are underlined. For PSNR and SSIM, the higher is better. Missing entries indicate the method impossible to run at those settings.

common scenarios, the error values in dilated regions can be overlapped, which worsens our results in such thin and overlapped masks. On the other hand, on the moving and stationary mask, our framework clearly outperforms by a large margin compared to DFC-Net [20] and FGVC [6], where the flow-based propagation is used. Such results show that our overall framework plays critical role in addressing the limitations of existing methods. More comparisons on other metrics, Video-based Fréchet Inception Distance (VFID) [18] and optical flow based warping error (EWarp) [18], are shown in the supplementary material.

We measure the execution time on 864x480 resolution for inpainting 50 frames to compare the efficiency. Since we design our entire components with deep frameworks, ours is $\times 6$ faster compared to FGVC [6] where the optimization-based methods are used.

Qualitative Results. In Fig. 8, we show our model's qualitative results compared with other methods, including STTN [22], FGVC [6] and FuseFormer [13]. To visualize the temporal consistency, we show the temporal profile [1] of the resulting videos below the completed frames. Sharp and smooth edges in the temporal profile indicate that the video has much less flickering. As can be seen, our method shows more visually pleasing results compared to others.

For a more comprehensive comparison, we conducted a user study to subjectively compare our method against others on the DAVIS dataset through Amazon Mechanical Turk. The experiments were performed on 30 videos that excludes easy (Tennis, Flamingo, etc.) and difficult (India, Drone, etc.) cases where every methods failed. The videos like Fig. 8 were shown to 20 participants. We unlabeled and shuffled the order in all videos, and asked to rank the results from 1 to 4 (1 is the highest preference). As shown in Fig. 9, the results of our model showed higher preference over others, supporting that our approach can produce more visually pleasing outputs.

4.4 Ablation Study

Effectiveness of Flow Completion. In Table 2(a), we show the flow end-point-error (EPE) metric to compare our flow completion with others. We set

Fig. 8: Qualitative results compared with [6,13,22]. The temporal profiles of the red scan line are shown below the results. **Best viewed in zoom.**

the estimated flows from RAFT [16] as pseudo ground-truth flows. In the table, FGVC/* means FGVC and FGVC* respectively. In DFC-Net [20] and FGVC [6], the performance is significantly reduced when the corrupted frames are used as input to the flow estimator. It verifies that the errors in corrupted flows prevent estimating correct flows. In the same setting where the corrupted frames are used, ours achieves the best performance.

In addition, we conduct a series of ablation studies to analyze the effectiveness of our flow completion module. First, we sequentially run pretrained FuseFormer [13] followed by original RAFT [16] (FF + RAFT). To demonstrate the effectiveness of local temporal network LTN, we show the results only using the flow estimator (w/o LTN). The flow estimator is trained with the same setting as ours. In addition, we train our flow estimator with equal weights in Eq. (4) instead of h (w/o Hard). Such results verify that our jointly trained flow completion module can deal with the errors from the local temporal network LTN and produce more accurate completed flows.

Effectiveness of Compensation. In Table 2(b), we show the performance of the error compensation network by replacing our flow completion with FGVC's method (FGVC* + Comp). Although the flow completion from FGVC* is worse than ours, it outperforms the full procedure of FGVC*. We also show the pseudo ground-truth flow from RAFT [16] (GT flow + Comp). These two experiments demonstrate our error compensation network is robust to errors from the flow completion. Also, we test our model without compensation stage (w/o Comp)

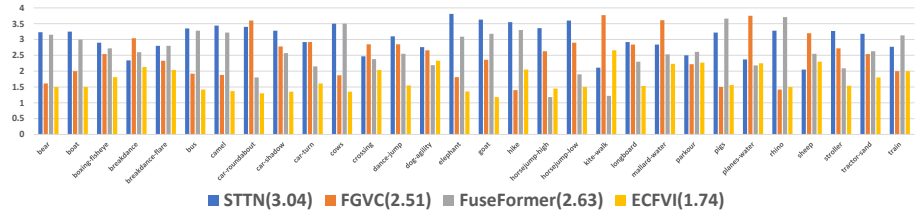


Fig. 9: A user study on DAVIS [2] object removal. The lower is better.

(a) Flow completion

	Stationary Mask	Moving Mask	
	Flow-EPE	Flow-EPE	Time(s)
DFC-Net/* [19]	0.32/0.89	0.38/0.86	43
FGVC/* [6]	0.12/0.57	0.28/0.80	412
FF [13] + RAFT [16]	0.49	0.80	52
w/o LTN	0.71	0.83	24
w/o Hard	0.40	0.58	52
Ours	0.33	0.49	52

(b) Video inpainting

	Stationary Mask		Moving Mask	
	PSNR	SSIM	PSNR	SSIM
FGVC* + Comp	32.87	0.9762	35.59	0.9823
GT flow + Comp	33.64	0.9796	37.17	0.9865
w/o Comp	30.05	0.9691	34.04	0.9810
w/o Guide	30.48	0.9692	33.81	0.9797
Ours	33.53	0.9795	36.80	0.9860

Table 2: Ablation studies on Youtube-VI dataset. For Flow-EPE, the lower is better. FGVC/* means FGVC and FGVC* respectively.

and without error guidance map (*w/o* Guide). In *w/o* Guide experiment, we use naive GAN where filled frames with propagation masks and remaining masks are only used as input. The results show severe performance drops compared to our method. These two experiments demonstrate that our compensation network using the error guidance map is necessary to handle the errors from the previous stages.

5 Conclusion

In this paper, we have proposed a simple yet effective video inpainting framework, which takes advantage of the flow-based methods while compensating for its shortcomings. We can propagate valid pixels with our flow completion and error compensation network, showing high-quality completed videos. Especially with the help of the error guidance map, we can prevent the errors from accumulating and amplifying at the following stages. We show that our method achieves state-of-the-art performance and demonstrate the benefits of our framework through extensive experiments. We also provide a new benchmark dataset, which provides comparative analysis into video inpainting methods.

Acknowledgement This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2014-3-00123, Development of High Performance Visual BigData Discovery Platform for LargeScale Realtime Data Analysis), and No. 2020-0-01361, Artificial Intelligence Graduate School Program (Yonsei University).

References

1. Caballero, J., Ledig, C., Aitken, A., Acosta, A., Totz, J., Wang, Z., Shi, W.: Real-time video super-resolution with spatio-temporal networks and motion compensation. In: CVPR. pp. 4778–4787 (2017) 12
2. 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 preprint arXiv:1803.00557 (2018) 10, 11, 14
3. 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) 1, 4, 10
4. Chang, Y.L., Liu, Z.Y., Lee, K.Y., Hsu, W.: Learnable gated temporal shift module for deep video inpainting". BMVC (2019) 1
5. Cheng, H.K., Tai, Y.W., Tang, C.K.: Rethinking space-time networks with improved memory coverage for efficient video object segmentation. In: NeurIPS (2021) 3, 11
6. Gao, C., Saraf, A., Huang, J.B., Kopf, J.: Flow-edge guided video completion. In: ECCV. pp. 713–729. Springer (2020) 2, 3, 4, 5, 11, 12, 13, 14
7. Hu, Y.T., Wang, H., Ballas, N., Grauman, K., Schwing, A.G.: Proposal-based video completion. In: ECCV. pp. 38–54. Springer (2020) 1
8. Huang, J.B., Kang, S.B., Ahuja, N., Kopf, J.: Temporally coherent completion of dynamic video. ACM Transactions on Graphics (TOG) **35**(6), 1–11 (2016) 4, 11
9. 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. pp. 2462–2470 (2017) 5, 6
10. Kim, D., Woo, S., Lee, J.Y., Kweon, I.S.: Deep video inpainting. In: CVPR. pp. 5792–5801 (2019) 4
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 10
12. Lee, S., Oh, S.W., Won, D., Kim, S.J.: Copy-and-paste networks for deep video inpainting. In: ICCV. pp. 4413–4421 (2019) 4, 12
13. Liu, R., Deng, H., Huang, Y., Shi, X., Lu, L., Sun, W., Wang, X., Dai, J., Li, H.: Fuseformer: Fusing fine-grained information in transformers for video inpainting. In: ICCV. pp. 14040–14049 (2021) 1, 2, 4, 5, 6, 11, 12, 13, 14
14. Oh, S.W., Lee, S., Lee, J.Y., Kim, S.J.: Onion-peel networks for deep video completion. In: ICCV. pp. 4403–4412 (2019) 4, 12
15. Shrivastava, A., Gupta, A., Girshick, R.: Training region-based object detectors with online hard example mining. In: CVPR. pp. 761–769 (2016) 7
16. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: European conference on computer vision. pp. 402–419. Springer (2020) 5, 6, 7, 13, 14
17. Wang, C., Huang, H., Han, X., Wang, J.: Video inpainting by jointly learning temporal structure and spatial details. In: AAAI. vol. 33, pp. 5232–5239 (2019) 4
18. Wang, T.C., Liu, M.Y., Zhu, J.Y., Liu, G., Tao, A., Kautz, J., Catanzaro, B.: Video-to-video synthesis. arXiv preprint arXiv:1808.06601 (2018) 12
19. Xu, N., Yang, L., Fan, Y., Yang, J., Yue, D., Liang, Y., Price, B., Cohen, S., Huang, T.: Youtube-vos: Sequence-to-sequence video object segmentation. In: ECCV. pp. 585–601 (2018) 4, 10, 14
20. Xu, R., Li, X., Zhou, B., Loy, C.C.: Deep flow-guided video inpainting. In: CVPR. pp. 3723–3732 (2019) 2, 3, 4, 5, 11, 12, 13

21. 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) 11
22. Zeng, Y., Fu, J., Chao, H.: Learning joint spatial-temporal transformations for video inpainting. In: ECCV. pp. 528–543. Springer (2020) 1, 2, 4, 6, 10, 11, 12, 13
23. Zou, X., Yang, L., Liu, D., Lee, Y.J.: Progressive temporal feature alignment network for video inpainting. In: CVPR. pp. 16448–16457 (2021) 1, 2, 4, 11, 12