# Inserting Videos into Videos

Donghoon Lee[1,2] Tomas Pfister[2] Ming-Hsuan Yang[2,3]
[1]Electrical and Computer Engineering and ASRI, Seoul National University
[2]Google Cloud AI
[3]Electrical Engineering and Computer Science, University of California at Merced

## Abstract

*In this paper, we introduce a new problem of manipulating a given video by inserting other videos into it. Our main task is, given an object video and a scene video, to insert the object video at a user-specified location in the scene video so that the resulting video looks realistic. We aim to handle different object motions and complex backgrounds without expensive segmentation annotations. As it is difficult to collect training pairs for this problem, we synthesize fake training pairs that can provide helpful supervisory signals when training a neural network with unpaired real data. The proposed network architecture can take both real and fake pairs as input and perform both supervised and unsupervised training in an adversarial learning scheme. To synthesize a realistic video, the network renders each frame based on the current input and previous frames. Within this framework, we observe that injecting noise into previous frames while generating the current frame stabilizes training. We conduct experiments on real-world videos in object tracking and person re-identification benchmark datasets. Experimental results demonstrate that the proposed algorithm is able to synthesize long sequences of realistic videos with a given object video inserted.*

## 1. Introduction

Object insertion in images aims to insert a new object into a given scene such that the manipulated scene looks realistic. In recent years, there has been increasing interest in this problem as it can be applied to numerous vision tasks, including but not limited to training data augmentation for object detection [19], interactive image editing [10], and manipulating semantic layouts [14]. However, there remains a significant gap between its potential and real-world applications since existing methods focus on modifying a single image while either requiring carefully pre-processed inputs, *e.g*. segmented objects without backgrounds [16], or generating objects from a random vector which makes it difficult to control the resulting appearance of the object directly [10, 14, 19].

In this paper, we introduce a new problem of inserting existing videos into other videos. More specifically, as shown in Figure 1, a user can select a video of an object of interest, *e.g*. a walking pedestrian, and put it at a desired location in other videos, *e.g*. surveillance scenes. Then, an algorithm composes the object seamlessly while it moves in the scene video. Note that unlike previous approaches [10, 14, 19], we do not assume that the input videos have expensive segmentation annotations. This not only allows users to edit videos more directly and intuitively, but also opens the door to numerous applications from training data augmentation for object tracking, video person re-identification, and video object segmentation, to video content generation for virtual reality or movies.

We pose the problem as a video-to-video synthesis task where the synthesized video containing an object of interest should follow the distribution of existing objects in the scene video. This falls into an unsupervised video-to-video translation problem since we do not have paired data in general, *i.e*. we do not observe exactly the same motion of the same object at the location we want to insert in different videos. Nevertheless, without any supervision, we face challenging issues such as handling different backgrounds, occlusions, lighting conditions and object sizes. Existing methods are limited to addressing such issues when there exists a number of moving objects and complex backgrounds. For example, the performance of an algorithm that relies on object segmentation methods, which often fails to crop foreground objects accurately in a complex scene, will be bounded by the accuracy of the segmentation algorithm.

To address the problem, we first address the related problems in the image domain, *i.e*. we study how to insert a given object image into other frames from different videos. To alleviate the issue of unpaired data, we propose a simple yet effective way to synthesize fake data that can provide supervisory signals for object insertion. The key idea of this supervision approach using the fake data is, when training a network, the fake data is carefully rendered to closely match

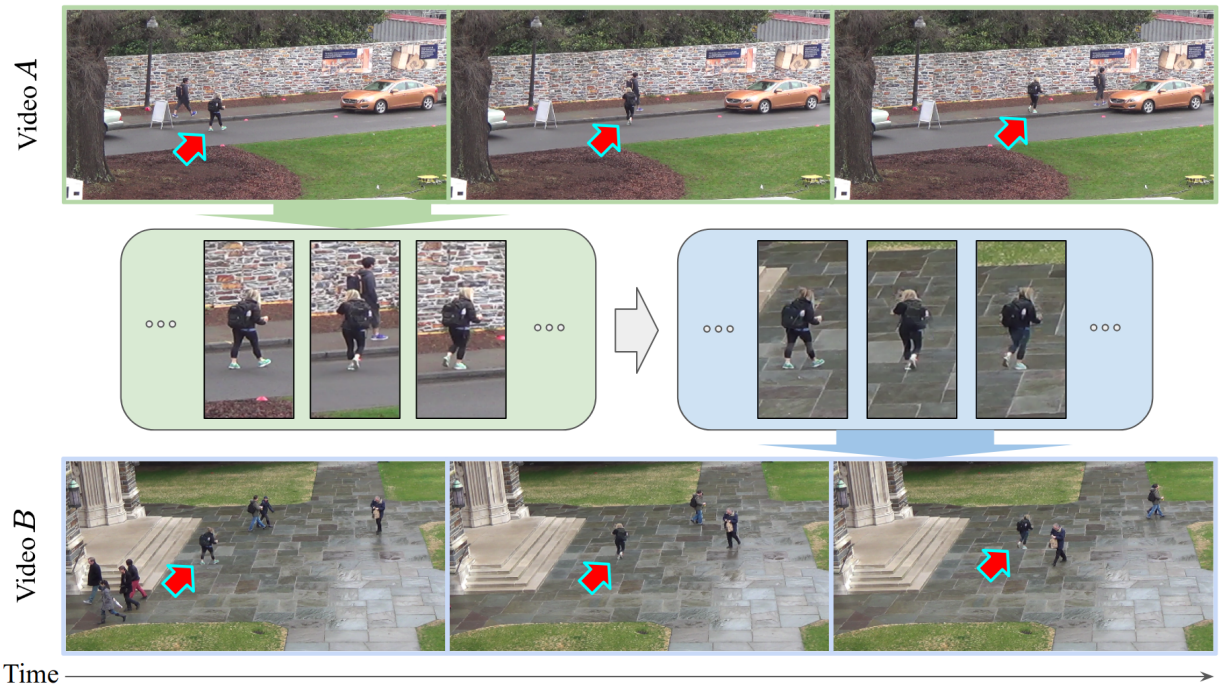


Figure 1: Given two videos, our algorithm aims to insert an object from one video to the other video. Red arrows point the inserted object originally exists in video $A$. We cast the problem as a video-to-video synthesis task. The proposed algorithm does not rely on any external domain knowledge such as the semantic segmentation mask or body pose.

the distribution of real data so that back-propagated gradient signals from the supervised fake data can help training the network with the unsupervised real data. In this work, the fake data is generated by blending an object image and a random background patch from each video. Then, the network learns how to reconstruct the object from the blended data. As the reconstruction errors provide strong supervisory signals, this approach facilitates the learning process of the generative adversarial framework [9] using unpaired real data. During inference, a new object is blended into a target location of the scene video and then fed to the trained network.

To extend the above-described algorithm to videos, we discuss how to utilize a history of synthesized frames to obtain a temporally consistent video. We observe that if we simply add a history of previous frames as a new source of input to the object insertion network trained on images, the network will easily collapse by relying only on the (clean) previous frames instead of the (blended) current frame. To avoid this pitfall, we use an idea from the denoising autoencoder [31]: a random noise is injected into previous frames before synthesizing the current frame. It forces the network to learn semantics between previous frames and the current input instead of blindly copy-and-pasting most of the information from the previous frames.

We conduct extensive experiments with strong baseline methods to evaluate the effectiveness of the proposed algorithm on real-world data. Experimental results show that the proposed algorithm can insert challenging objects, *e.g.* moving pedestrians under the cluttered backgrounds, into other videos. For quantitative evaluation, we carry out three experiments. First, we measure the recall of the state-of-the-art object detector [22] for the inserted object. It assesses the overall appearance of the inserted object given the surrounding context. Second, given the state-of-the-art segmentation algorithm [4], we measure pixel-level precision and recall of the inserted object. Third, we perform a human subjective study for evaluating the realism of inserted objects.

The main contributions of this work are summarized as follows:

- We introduce an important and challenging problem which broadens the domain of object insertion from images to videos.
- We propose a novel approach to synthesize supervised fake training pairs that can help a deep neural network to learn to insert objects without supervised real pairs.
- We develop a new conditional GAN model to facilitate the joint training of both unsupervised real and supervised fake training pairs.
- We demonstrate that the proposed algorithm can synthesize realistic videos based on challenging real-world input videos.

## 2. Related Work

**Inserting objects into images.** Given a pair of an object image and a scene image, the ST-GAN approach [16] learns a warping of the object conditioned on the scene. Based on the warping, the object is transformed to a new location without changing its appearance. As it focuses on geometric realism, they use carefully segmented object as an input.

Other approaches aim to insert an object by rendering its appearance. In [10], an object in a target category is inserted into a scene given a location and a size of a bounding box. It first predicts a shape of the object in the semantic space, after which an output image is generated from the predicted semantic label map and an input image. A similar approach is proposed in [19] without using a semantic label map. A bounding box of a pedestrian is replaced by random noise and then infilled with a new pedestrian based on the surrounding context.

To learn both placement and shape of a new object, the method in [5] removes existing objects from the scene using an image in-painting algorithm. Then, a network is trained to recover the existing objects. The results of this method rely significantly run script on whether the adopted image in-painting algorithm performs well, *e.g.* not generating noisy pixels. This issue is alleviated in [14] by learning the joint distribution of the location and shape of an object conditioned on the semantic label map. This method aims to find plausible locations and sizes of a bounding box by learning diverse affine transforms that warp a unit bounding box into the scene. Then, objects of different shapes are synthesized conditioned on the predicted location and its surrounding context.

In contrast to existing methods, our algorithm allows a user to specify both the appearance of an object to insert and its location. In addition, our algorithm does not require a segmentation map for training or test.

**Conditional video synthesis.** The future frame prediction task conditions on previous frames to synthesize image content [18, 7, 32, 6, 15, 29, 30]. Due to the future uncertainty and accumulated error in the prediction process, it typically can generate only short video sequences. On the other hand, we synthesize long video sequences by inserting one video into other videos.

The contents of a video can be transferred to other videos to synthesize new videos. In [3], given a source video of a person, the method transfers one's motion to another person in the target video. This method estimates the object motion using a detected body pose and trains a network to render a person conditioned on the pose. The trained network renders a new video as if the target subject follows the motion of the source video. Instead of following exactly the same motion, the approach in [1] transfers an abstract content of the source video while the style of the target video is preserved. A cyclic spatio-temporal constraint is proposed to address the task in an unsupervised manner. It translates a source frame to a target domain and predicts the next frame. Then, the predicted frame is translated back to the source domain. This work also forms a cyclic loop which can improve the video quality.

The dynamic contents/textures in a video can also be used for conditional video synthesis. In [28], dynamic textures in a video such as water flow or fire flame are captured by learning a two-stream network. Then, the work animates an input image to a video with realistic dynamic motions. Artistic styles of a video is transferred to edit a target video while preserving its contents [11, 25].

For more generic video-to-video translations, the scheme in [33] formulates conditional generative adversarial networks (GANs) to synthesize photorealistic videos given a sequence of semantic label maps, sketches or human pose as an input. During training, the network takes paired data as input, *e.g.* sequences of a semantic label map and the corresponding RGB image sequence. The network is constrained to preserve the content of the input sequence in the output video.

## 3. Proposed Algorithm

In this work, we consider the problem where a user selects an object in video $A$ and wants to insert it at a desired location in video $B$. We assume that each video has annotations for bounding boxes and IDs of objects at every frame. From the bounding boxes of the selected object in $A$, we obtain a video $\mathbf{u}_A$ consisting of cropped images. The goal is to translate $\mathbf{u}_A$ to $\mathbf{v}_A$ so that the translated video is realistic when inserted into $B$. We first tackle this problem's image counterpart and then extend it to videos.

### 3.1. Inserting images into images

Let $u_A$ denote a frame in $\mathbf{u}_A$ which will be inserted into a user-defined region $r_B$ in $B$. We train a generator network $G_I$ which takes $u_A$ and $r_B$ as inputs to render an output $v_A$. Note that this is different from existing image-to-image translation tasks [12, 13, 17, 35, 36] since they aim to preserve the content of an input image while changing it to different attributes or styles, *e.g.* a semantic map is translated to RGB images that have the same semantic layout. In contrast, we need to translate two different images into a single image while learning which part of the content in each image should be preserved.

One challenging issue is that we do not have a training tuple $(u_A, r_B, v_A)$. To address this issue, we first cast the problem as a conditional image in-painting task. More specifically, we corrupt $r_B$ by blending $u_A$ using pixel-wise multiplications with a fixed binary mask $m$, *i.e.* $u_A \oplus r_B = u_A m/2 + r_B(1-m/2)$, as shown in the Figure 2. Then, the

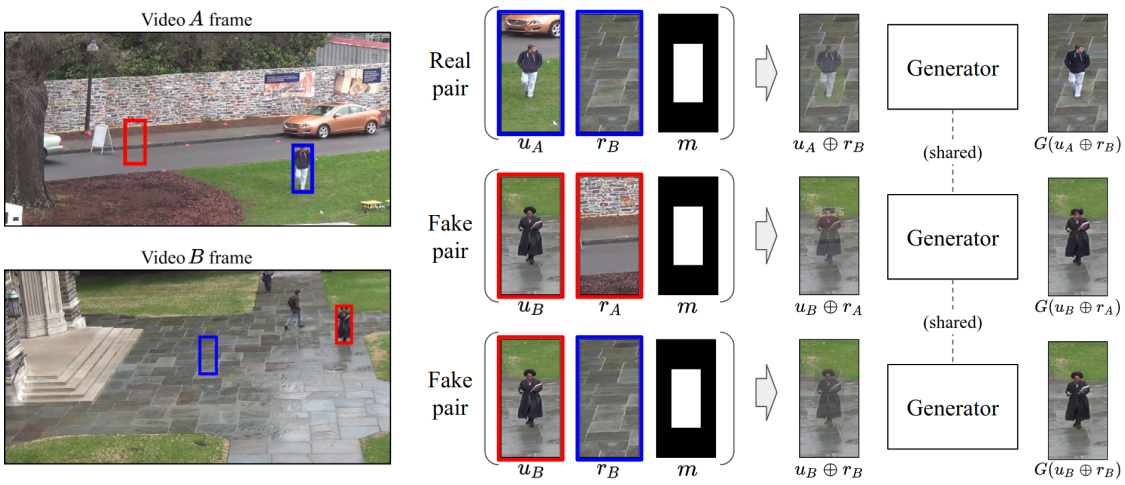


Figure 2: Main steps of the proposed algorithm for inserting an object into an image. Given the video $A$ and $B$, we crop objects and backgrounds from each video and synthesize blended images $u_A \oplus r_B$, $u_B \oplus r_A$, and $u_B \oplus r_B$. By learning how to reconstruct $u_B$ from $u_B \oplus r_A$ and $u_B \oplus r_B$, we can guide the network to insert $u_A$ into the center of $r_B$. In addition to reconstruction losses from fake pairs, we have additional objective functions as described in (6).

(a) $u_A$ (b) $r_B$ (c) (1) + (2) (d) (2) + (3) (e) (6)

Figure 3: Object insertion results using different objective functions. Inputs are $u_A$ and $r_B$.

generator learns a mapping $G_I : (u_A \oplus r_B) \rightarrow v_A$ to synthesize realistic $v_A$. To this end, the generator learns how to render the object while suppressing mismatched backgrounds based on the context of surrounding non-blended regions. The key advantage of this formulation is that it is easy to synthesize fake training pairs that are similar to $(u_A \oplus r_B, v_A)$.

In this paper, we propose two types of fake pairs $(u_B \oplus r_A, u_B)$ and $(u_B \oplus r_B, u_B)$ to learn object insertion. The intuition behind it is that these pairs contain two separate tasks that the generator has to perform during inference: rendering consistent backgrounds based on the context, and recovering the object region overlapped with $r_B$. We design two objective functions for fake pairs using $G_I$ and an image discriminator $D_I$. First,

$$\begin{aligned}\mathcal{L}_{\mathcal{A}}^{fake}(G_I, D_I) &= \mathbb{E}_{(u_B, r_A)}[\log D_I(u_B, u_B \oplus r_A)] \\ &+ \mathbb{E}_{(u_B, r_B)}[\log D_I(u_B, u_B \oplus r_B)] \\ &+ \mathbb{E}_{(u_B, r_A)}[\log(1 - D_I(G_I(u_B \oplus r_A), u_B \oplus r_A))] \\ &+ \mathbb{E}_{(u_B, r_B)}[\log(1 - D_I(G_I(u_B \oplus r_B), u_B \oplus r_B))],\end{aligned} \tag{1}$$

is a conditional adversarial loss to make the reconstructed image sharper and realistic[1]. Second,

$$\mathcal{L}_{\mathcal{R}}(G_I) = \|u_B - G_I(u_B \oplus r_A)\| + \|u_B - G_I(u_B \oplus r_B)\|, \tag{2}$$

is a content loss to reconstruct $u_B$.

We present results on the real pair using a network trained with fake pairs in Figure 3(c). Although some parts are blurry, the overall shape and appearance of inserted objects are preserved. In addition, most of the background pixels from $A$ are removed and replaced by $r_B$, showing that fake pairs provide meaningful signals to the network to insert unseen objects. Thus, we expect that the network can be trained well with both of real and fake pairs. We update the adversarial loss to consider real pairs as follows:

$$\begin{aligned}\mathcal{L}_{\mathcal{A}}(G_I, D_I) &= \mathcal{L}_{\mathcal{A}}^{fake}(G_I, D_I) \\ &+ \mathbb{E}_{(u_A, r_B)}[\log(1 - D_I(G_I(u_A \oplus r_B), u_A \oplus r_B))].\end{aligned} \tag{3}$$

However, as shown in Figure 3(d), the synthesized results become unstable when we naively train the network using (2) and (3). We attribute this to different distributions of the fake pair and real pair. Although their similar distributions make it possible to generalize the network to unseen images, when the network actually learns with both pair types, it is able to distinguish between them, thus limiting generalization. We address this issue by making it more difficult for the network to distinguish these pairs. In particular, we make it uncertain about whether the input is sampled from the fake pair or real pair. To this end, we add a discriminator $D_E$ that aims to distinguish the input type

[1] We denote $\mathbb{E}_{(\cdot)} \triangleq \mathbb{E}_{(\cdot) \sim p_{data}(\cdot)}$ for notational simplicity.

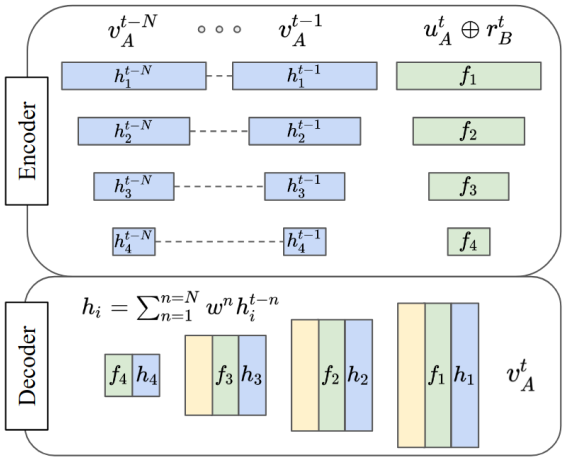


Figure 4: Network structure of the video insertion network $G_V$. As an illustrative example, we show the case where the number of layers is four. The network takes previous frames $(v_A^{t-N}, \ldots, v_A^{t-1})$ and a blended image $u_A^t \oplus r_B^t$ as an input to render $v_A^t$. Each square denotes a layer in the network. The dashed lines indicates shared weights, and layers next to each other represent channel concatenations.

based on its embedded vector as follows:

$$\begin{aligned}\mathcal{L}_{\mathcal{A}}(G_I, D_E) = &\mathbb{E}_{(u_B, r_A)}[\log D_E(e_{u_B \oplus r_A})] \\ &+ \mathbb{E}_{(u_B, r_B)}[\log D_E(e_{u_B \oplus r_B})] \\ &+ \mathbb{E}_{(u_A, r_B)}[\log(1 - D_E(e_{u_A \oplus r_B}))],\end{aligned} \tag{4}$$

where $e_x$ denotes an embedded vector from the encoder in $G_I$ with an input $x$. The encoder is trained to fool the discriminator by embedding the fake pair and real pair into the same space. This embedding vector is fed to discriminators as a conditional input. We tile the vector to the same size of the input image and concatenate them to the input channel. The objective function $\mathcal{L}_{\mathcal{A}}(G_I, D_I)$ is modified as follows:

$$\begin{aligned}\mathcal{L}_{\mathcal{A}}(G_I, D_I) = &\mathbb{E}_{(u_B, r_A)}[\log D_I(u_B, e_{u_B \oplus r_A})] \\ &+ \mathbb{E}_{(u_B, r_B)}[\log D_I(u_B, e_{u_B \oplus r_B})] \\ &+ \mathbb{E}_{(u_B, r_A)}[\log(1 - D_I(G_I(u_B \oplus r_A), e_{u_B \oplus r_A}))] \\ &+ \mathbb{E}_{(u_B, r_B)}[\log(1 - D_I(G_I(u_B \oplus r_B), e_{u_B \oplus r_B}))] \\ &+ \mathbb{E}_{(u_A, r_B)}[\log(1 - D_I(G_I(u_A \oplus r_B), e_{u_A \oplus r_B}))].\end{aligned} \tag{5}$$

Finally, the overall objective function for object insertion on the image domain is formulated as follows:

$$\mathcal{L}(G_I, D_I, D_E) = \mathcal{L}_{\mathcal{A}}(G_I, D_I) + \mathcal{L}_{\mathcal{A}}(G_I, D_E) + \mathcal{L}_{\mathcal{R}}(G_I). \tag{6}$$

Figure 3(e) shows that the inserted objects using the loss function in (6) are sharp and realistic.

### 3.2. Inserting videos into videos

In this section, we discuss how to extend the object insertion model from images to videos. To this end, we make two major modifications. First, when rendering the current frame, we also look up previous frames. Second, we add a new term in the objective function to synthesize temporally consistent videos.

Let $G_V$ denote a video generator that learns a mapping $G_V : (\mathbf{u}_A \oplus \mathbf{r}_B) \rightarrow \mathbf{v}_A$[2]. One simple mapping is to apply $G_I$ for each frame. However, as the mapping of a frame is independent from neighboring frames, the resulting sequence becomes temporally inconsistent. Therefore, we let $G_V$ to additionally look up $N$ previous frames while synthesizing each frame from the blended input. This Markov assumption is useful for generating long sequence videos [33]. Figure 4 shows the proposed U-net [24] style encoder-decoder network architecture. If the network operates without blue layers, which correspond to the feature maps of previous frames, then it is identical to $G_I$ in Section 3.1. The network encodes all previous frames using a shared encoder. Then, the feature map is linearly combined with a scalar weight $w^n$ which represents the importance of each frame. We use $N = 2$ and $w^1 = w^2 = 0.5$ for experiments in this work.

To learn $G_V$, we calculate an error signal for the generated sequence using the following objective function:

$$\begin{aligned}\mathcal{L}(G_V, D_I, D_V, D_E) = &\mathcal{L}_{\mathcal{A}}(G_V, D_I) + \mathcal{L}_{\mathcal{A}}(G_V, D_V) \\ &+ \mathcal{L}_{\mathcal{A}}(G_V, D_E) + \mathcal{L}_{\mathcal{R}}(G_V),\end{aligned} \tag{7}$$

where $D_V$ is a video discriminator. The first term is defined similarly to (5) while we select a random frame from the generated sequence to calculate the loss; this term focuses on the realism of the selected frame. The second term assesses the rendered sequence as follows:

$$\begin{aligned}\mathcal{L}_{\mathcal{A}}(G_V, D_V) = &\mathbb{E}_{(\mathbf{u}_B, \mathbf{r}_A)}[\log D_V(\mathbf{u}_B, e_{\mathbf{u}_B \oplus \mathbf{r}_A})] \\ &+ \mathbb{E}_{(\mathbf{u}_B, \mathbf{r}_B)}[\log D_V(\mathbf{u}_B, e_{\mathbf{u}_B \oplus \mathbf{r}_B})] \\ &+ \mathbb{E}_{(\mathbf{u}_B, \mathbf{r}_A)}[\log(1 - D_V(G_V(\mathbf{u}_B \oplus \mathbf{r}_A), e_{\mathbf{u}_B \oplus \mathbf{r}_A}))] \\ &+ \mathbb{E}_{(\mathbf{u}_B, \mathbf{r}_B)}[\log(1 - D_V(G_V(\mathbf{u}_B \oplus \mathbf{r}_B), e_{\mathbf{u}_B \oplus \mathbf{r}_B}))] \\ &+ \mathbb{E}_{(\mathbf{u}_A, \mathbf{r}_B)}[\log(1 - D_V(G_V(\mathbf{u}_A \oplus \mathbf{r}_B), e_{\mathbf{u}_A \oplus \mathbf{r}_B}))].\end{aligned} \tag{8}$$

The third and fourth terms are defined similarly to (4) and (2), respectively.

In addition, while training the network, we observe that the predicted frame $v_A^t$ heavily relies on the previous frames rather than the current input. The main reason is that the current input is corrupted by a blending $u_A^t \oplus r_B^t$ which makes it more difficult to process. Therefore, instead of learning to recover the current frame, the network gradually ignores the current input and depends more on the previous frame. It is a critical problem when generating long videos as the error from the previous frame is accumulated. As a result, the generated sequence contains severe artifacts after a number of frames. To address this issue, we degrade

[2]We denote $(\mathbf{u}_A \oplus \mathbf{r}_B)$ as a sequence of blended inputs $((u_A^1 \oplus r_B^1, \ldots, (u_A^T \oplus r_B^T))$ where $T$ is the number of frames.

previous frames as well using random noise before render the current frame. By blocking this easy cheating route, the network has to learn semantic relationships between the two inputs instead of relying on one side. It makes the network significantly stable during training.

## 4. Experimental Results

We evaluate our method on the multi-target tracking or person re-identification databases such as the DukeMTMC [23], TownCenter [2], and UA-DETRAC [34] to show applicability of our algorithm on real-world examples. These datasets record challenging scenarios where pedestrians or cars move naturally. We split 20% of the data as a test set and present experimental results on the test set. Additional results, including sample generated videos and a user study, are included in the supplementary material.

**Implementation details.** For all experiments, the network architecture, parameters, and initialization are similar to DCGAN [21]. We use transposed convolutional layers with 64 as a base number of filters for both of the generator and discriminator. The batch size is set to 1 and instance normalization is used instead of batch normalization. Input videos are resized to $1024 \times 2048$ pixels. We crop $u_{(\cdot)}$ and $r_{(\cdot)}$ from the video and resize to $256 \times 128$ pixels. Then, we render an object on the $256 \times 128$ pixels patch. It is transformed to $512 \times 256$ pixels image or video for visualization. For each iteration, we pick a random location in $A$ to put a new object since we want to cover various location and size input of a user.

**Baseline models and qualitative evaluations.** As the problem introduced in this paper is a new problem, we design strong baselines for performance evaluation.

For object insertion in images, we present six baseline models. First, we apply the state-of-the-art semantic segmentation algorithm [4] to segment the interested object region in video $A$, *e.g.* a pedestrian in the DukeMTMC dataset. Then, object pixels are copied to a region in video $B$ using the predicted segmentation mask as shown in Figure 5(c). However, the predicted segmentation mask is inaccurate due to the complex background and articulated human pose. Therefore, some parts of the object are often missing and undesired background pixels from video $A$ are included in the synthesized frame. In addition, the brightness of the inserted pixels does not match with surrounding pixels in video $B$. Second, we apply the Poisson blending [20] method to the predicted object mask as shown in Figure 5(d). Although the boundary of object becomes smoother, the blended image still contains artifacts. In addition, the results depend on the performance of the segmentation algorithm.

Third, we design four GAN-based methods. One naive approach focuses on synthesizing realistic example using the following objective function:

$$\begin{aligned}\mathcal{L}_{\mathcal{A}}^{base}(G, D) = \mathbb{E}_{u_B}[\log D(u_B)] \\ + \mathbb{E}_{(u_A, r_B)}[\log D(G(u_A \oplus r_B))].\end{aligned} \tag{9}$$

In this case, the generator easily collapses as it is not guided to preserve the content of the input object as shown in Figure 5(e). To alleviate this issue, we add an objective function that checks the content in the generated image, *e.g.* a pixel-wise reconstruction loss or the perceptual loss [8] as shown in Figure 5(f) and Figure 5(g). The objective functions are defined as follows:

$$\mathcal{L}_{pixel}^{base}(G, D) = \mathcal{L}_{\mathcal{A}}^{base}(G, D) + \|u_A m - v_A m\|, \tag{10}$$

$$\begin{aligned}\mathcal{L}_{perceptual}^{base}(G, D) = \mathcal{L}_{\mathcal{A}}^{base}(G, D) \\ + \sum_l \frac{1}{C_l H_l W_l} \|\phi_l(u_A m) - \phi_l(v_A m)\|_2^2,\end{aligned} \tag{11}$$

where $\phi_l$ is $l$-th activation map of the VGG19 network [27] with a shape of $C_l \times H_l \times W_l$. We use activation maps of relu2_2 and relu3_3 layers of the VGG19 network which is pre-trained on the ImageNet dataset [26] to calculate the perceptual loss. The main limitation of these approaches is that the network is trained to preserve all pixels around the object in $u_A$. As a result, a large number of undesired background pixels appear in $v_A$.

The final baseline model uses the cycle consistency loss [35] which has been used to train networks with unpaired training data. For the cyclic loss, we learn two mapping functions $G\colon (u_A, r_B) \rightarrow v_A$ and $F\colon (u_B, r_A) \rightarrow v_B$. By taking the conditional inputs into account, the objective function is defined by:

$$\begin{aligned}\mathcal{L}_{cyc}^{base}(G, F, D_A, D_B) = \mathcal{L}_{\mathcal{A}}(G, D_B) + \mathcal{L}_{\mathcal{A}}(F, D_A) \\ + \mathbb{E}_{(u_A, r_B)}[\|F(G(u_A, r_B), u_A(1-m)) - u_A\|_1] \\ + \mathbb{E}_{(u_B, r_A)}[\|G(F(u_B, r_A), u_B(1-m)) - u_B\|_1] \\ + \mathbb{E}_{(u_A, r_B)}[\|G(u_A, r_B)(1-m) - r_B(1-m)\|_1] \\ + \mathbb{E}_{(u_B, r_A)}[\|F(u_B, r_A)(1-m) - r_A(1-m)\|_1],\end{aligned} \tag{12}$$

where $D_A$ and $D_B$ are discriminators for each video and $\mathcal{L}_{\mathcal{A}}(G, D_B)$ and $\mathcal{L}_{\mathcal{A}}(F, D_A)$ are typical adversarial losses. The last two terms are added to force the network to insert an object at a given $r_A$ or $r_B$. Although the formulation has the potential to learn unpaired mappings, it still cannot guide the network to preserve the same object while translating images as shown in Figure 5(h). In addition, we observe that this makes the network unstable during training. In contrast, the proposed algorithm inserts an object with its sharp shape and renders less noisy background pixels as shown in Figure 5(i).

For video object insertion, we consider two baseline models. First, frames are synthesized without using previous frames. As the model only processes the current frame as an input, the overall video may contain flickering or inconsistent content. Second, a video is generated without

(a) $u_A$ (b) $r_B$ (c) [4] (d) [4] + [20] (e) (9) (f) (10) (g) (11) (h) (12) (i) Ours (6)

Figure 5: Object insertion results for different baseline models given input $u_A$ and $r_B$.

Table 1: Recall of the state-of-the-art object detector [22] on the DukeMTMC database. B1: Adobe Premiere blending mode. B2: Segmentation-based composition [4].

| Method | B1 | B2 | (9) | (10) | (11) | Our |
|---|---|---|---|---|---|---|
| Recall | 0.39 | 0.76 | 0.73 | 0.80 | 0.78 | **0.86** |

injecting noise into previous frames. In such cases, as small errors in each frame accumulate over frame, the synthesized images are likely noisy.

Figure 6 shows video object insertion results with baseline comparisons. We use an automatic blending mode of a commercial video editing software (Adobe Premier CC Pro) as one baseline. The other baseline uses DeepLabv3+ [4] to copy and paste the predicted segment along frames. It shows that the proposed algorithm can synthesize more realistic videos than other baseline methods. In addition, as shown in Figure 7, our algorithm is capable of inserting videos across databases and different objects such as a car.

**Quantitative evaluations.** To quantify the realism of the inserted object, an object detector is often used to locate the inserted object [14, 19, 5]. The premise is that a detector is likely to locate only well-inserted objects since state-of-the-art methods take both of the object and its surrounding context into account. We use the YOLOv3 detector [22] to determine whether it can correctly detect the inserted object or not. We fix the detection threshold and measure the recall of the detector by calculating the intersection over union (IoU) between the inserted object and detected bounding boxes, using an IoU threshold of 0.5. Table 1 shows the average recall using a network trained with five different iterations. For each experiment, we sample one thousand images at random. It shows that the proposed algorithm achieves the highest recall value on average. In addition, we accidentally found an interesting corner case of this experiment. While (9) generates non-realistic images in a similar mode as shown in Figure 5(e), this method once achieves the highest recall value. It reveals one limitation of assess-

Table 2: Object insertion score on the DukeMTMC dataset. B1: Adobe Premiere blending mode.

| Method | B1 | (9) | (10) | (11) | Our |
|---|---|---|---|---|---|
| Precision | 0.32 | 0.61 | 0.70 | 0.76 | **0.85** |
| Recall | 0.28 | 0.26 | 0.47 | 0.61 | **0.72** |
| OIS | 0.30 | 0.36 | 0.56 | 0.68 | **0.78** |

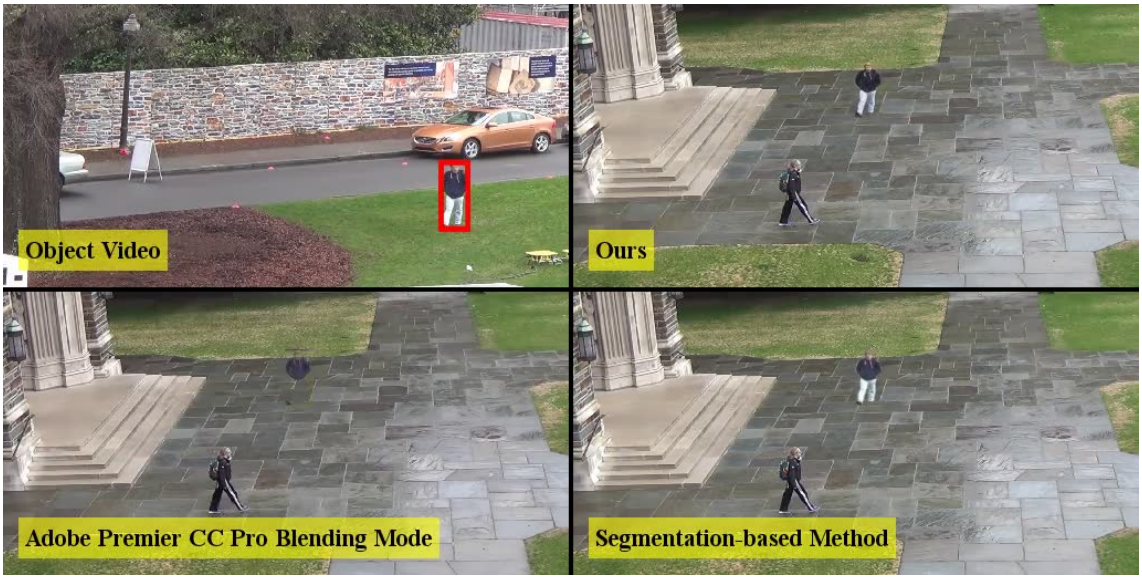


Figure 6: Inserting a pedestrian video on the DukeMTMC dataset. Click the image to play the video.

Figure 7: Results of cross dataset pedestrian insertion (from the DukeMTMC dataset to TownCenter dataset) and a car video insertion on the UA-DETRAC dataset.

ing the synthesized image using a detector, *i.e.*, if a trained detector mistakenly returns positive detection result for a non-realistic fake image, then it is highly likely that other non-realistic images in the same mode will be detected as positive samples as well.

While detection results give an idea of how realistic (or at least, detectable) the inserted object is, it does not indicate pixel-level accuracy of the object insertion, *i.e.* whether the object pixels in the input are preserved in the output. To this end, we introduce a new metric based on pixel-level precision and recall for object insertion. Given a semantic segmentation algorithm, let $s_A$ denote a binary segmentation mask of the input object image. Also let $s_\Delta$ be a binary mask where $s_\Delta(i,j) = 1$ when $v_A(i,j)$ is closer to $u_A(i,j)$ than $r_B(i,j)$. Thus, $s_\Delta$ represents pixel locations of the inserted object. We then define the precision $P$, recall $R$, and object insertion score (OIS) as follows:

$$P = \frac{|s_A \odot s_\Delta|}{|s_\Delta|}, \; R = \frac{|s_A \odot s_\Delta|}{|s_A|}, \; \text{OIS} = 2\frac{PR}{P+R}, \quad (13)$$

where $\odot$ is an element-wise multiplication, $|s|$ is an area of non-zero region in $s$, and OIS is defined using the $F_1$ score. We calculate the score based on randomly generated one thousand samples and segmentation masks are obtained by the DeepLabv3+ [4] method. Table 2 shows that the proposed algorithm achieves the highest OIS against other baseline algorithms. We also note that the OIS of the baseline model based on (9) is the lowest.

In order to show potential application for data augmentation, we train a detector using synthesized objects by our algorithm. We detect pedestrians on the DukeMTMC dataset using YOLOv3 initialized on the ImageNet. For training and evaluation, we pick 100 and 1,000 frames at random from the video of camera 5 in the dataset. In addition, 3,000 frames are augmented by inserting pedestrians from camera 1. It boosts mAP from 53.1% to 68.3%.

## 5. Conclusion

In this paper, we have introduced an algorithm to a new problem: manipulating a given video by inserting other videos into it. It is a challenging task as it is inherently an unsupervised (unpaired) problem. Unlike existing approaches, we propose an algorithm that converts the problem to a paired problem by synthesizing fake training pairs and corresponding loss functions. We conducted experiments on real-world videos and demonstrated that the proposed algorithm is able to render long realistic videos with a given object video inserted. As a future work, it is interesting to make the inserted object interact with the new video, *e.g.*, path navigation or occlusion handling.

## References


[1] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-GAN: Unsupervised video retargeting. In *European Conference on Computer Vision*, 2018. 3

[2] Ben Benfold and Ian Reid. Stable multi-target tracking in real-time surveillance video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011. 6

[3] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. *arXiv preprint arXiv:1808.07371*, 2018. 3

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision*, 2018. 2, 6, 7, 8

[5] Jui-Ting Chien, Chia-Jung Chou, Ding-Jie Chen, and Hwann-Tzong Chen. Detecting nonexistent pedestrians. In *IEEE International Conference on Computer Vision*, 2017. 3, 7

[6] Emily Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In *Neural Information Processing Systems*, 2017. 3

[7] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Neural Information Processing Systems*, 2016. 3

[8] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 6

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Neural Information Processing Systems*, 2014. 2

[10] Seunghoon Hong, Xinchen Yan, Thomas Huang, and Honglak Lee. Learning hierarchical semantic image manipulation through structured representations. In *Neural Information Processing Systems*, 2018. 1, 3

[11] Haozhi Huang, Hao Wang, Wenhan Luo, Lin Ma, Wenhao Jiang, Xiaolong Zhu, Zhifeng Li, and Wei Liu. Real-time neural style transfer for videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 3

[12] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *European Conference on Computer Vision*, 2018. 3

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 3

[14] Donghoon Lee, Sifei Liu, Jinwei Gu, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Context-aware synthesis and placement of object instances. In *Neural Information Processing Systems*, 2018. 1, 3, 7

[15] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P Xing. Dual motion gan for future-flow embedded video prediction. In *IEEE International Conference on Computer Vision*, 2017. 3

[16] Chen-Hsuan Lin, Ersin Yumer, Oliver Wang, Eli Shechtman, and Simon Lucey. ST-GAN: Spatial transformer generative adversarial networks for image compositing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 3

[17] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Neural Information Processing Systems*, 2017. 3

[18] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *International Conference on Learning Representations*, 2016. 3

[19] Xi Ouyang, Yu Cheng, Yifan Jiang, Chun-Liang Li, and Pan Zhou. Pedestrian-Synthesis-GAN: Generating pedestrian data in real scene and beyond. *arXiv preprint arXiv:1804.02047*, 2018. 1, 3, 7

[20] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, 2003. 6, 7

[21] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 6

[22] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2, 7

[23] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision Workshops*, 2016. 6

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015. 5

[25] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos and spherical images. *International Journal of Computer Vision*, pages 1–21, 2018. 3

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 6

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 6

[28] Matthew Tesfaldet, Marcus A. Brubaker, and Konstantinos G. Derpanis. Two-stream convolutional networks for dynamic texture synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 3

[29] Ruben Villegas, Jimei Yang, Seunghoon Hang, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *International Conference on Learning Representations*, 2017. 3

[30] Ruben Villegas, Jimei Yang, Yuliang Zou, Seunghoon Hang, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. In *International Conference on Machine Learning*, 2017. 3

[31] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust

features with denoising autoencoders. In *International Conference on Machine Learning*, 2008. 2

[32] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, 2016. 3

[33] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Neural Information Processing Systems*, 2018. 3, 5

[34] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *arXiv preprint arXiv:1511.04136*, 2015. 6

[35] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision*, 2017. 3, 6

[36] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Neural Information Processing Systems*, 2017. 3