Diverse Generation from a Single Video Made Possible

Niv Haim[*][†], Ben Feinstein[*], Niv Granot, Assaf Shocher, Shai Bagon, Tali Dekel, and Michal Irani

Weizmann Institute of Science, Rehovot, Israel
Project webpage: https://nivha.github.io/vgpnn

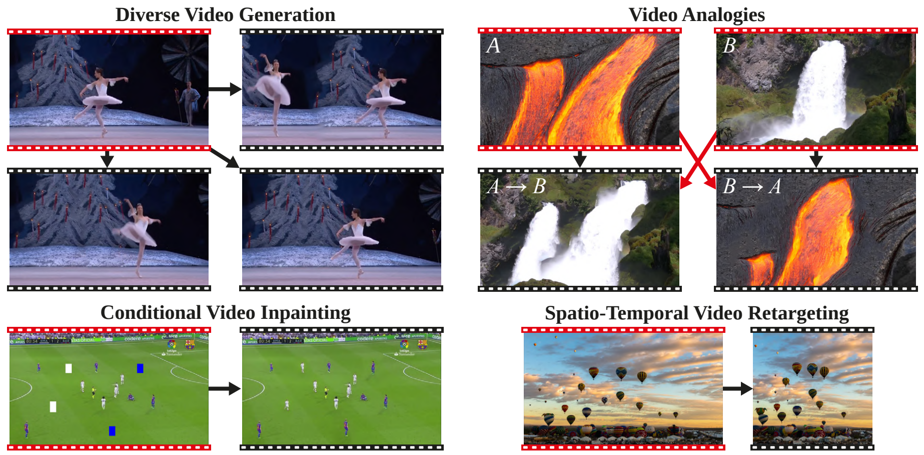
Diverse Video Generation
Video Analogies
A
B
A → B
B → A
Conditional Video Inpainting
Spatio-Temporal Video Retargeting

Fig. 1: We adapt classical patch-based approaches as a better, much faster non-parametric alternative to single video GANs, for a variety of video generation and manipulation tasks.

Abstract. GANs are able to perform generation and manipulation tasks, trained on a single video. However, these single video GANs require unreasonable amount of time to train on a single video, rendering them almost impractical. In this paper we question the necessity of a GAN for generation from a single video, and introduce a non-parametric baseline for a variety of generation and manipulation tasks. We revive classical space-time patches-nearest-neighbors approaches and adapt them to a scalable unconditional generative model, without any learning. This simple baseline surprisingly outperforms single-video GANs in visual quality and realism (confirmed by quantitative and qualitative evaluations), and is disproportionately faster (runtime reduced from several days to seconds). Other than diverse video generation, we demonstrate other applications using the same framework, including video analogies and spatio-temporal retargeting. Our proposed approach is easily scaled to Full-HD videos. These observations show that the classical approaches, if adapted correctly, significantly outperform heavy deep learning machinery for these tasks. This sets a new baseline for single-video generation and manipulation tasks, and no less important – makes diverse generation from a single video practically possible for the first time.

[*]Equal contribution
[†]Corresponding author: `niv.haim@weizmann.ac.il`

1 Introduction

Generation and editing of natural videos remain challenging, mainly due to their large dimensionality and the enormous space of motion they span. Most modern frameworks train generative models on a large collection of videos, producing high quality results for only a limited class of videos. These include extensions of GANs [23] to video data [2, 36, 48, 58, 62, 66], video to video translation [8, 16, 40, 63–65, 71] and autoregressive sequence prediction [3, 6, 7, 17, 22, 59–61].

While externally-trained generative models produce impressive results, they are restricted to the types of video dynamics in their training set. On the other side of the spectrum are *single-video GANs*. These video generative models train on a *single* input video, learn its distribution of space-time patches, and are then able to generate a diversity of new videos with the same patch distribution [5, 25]. However, these take very long time to train for each input video, making them applicable to only small spatial resolutions and to very short videos (typically, very few small frames). Furthermore, their output oftentimes shows poor visual quality and noticeable visual artifacts. These shortcomings render existing single-video GANs impractical and unscalable.

Video synthesis and manipulation of a single video sequence based on its distribution of space-time patches dates back to classical pre-deep learning methods. These classical methods demonstrated impressive results for various applications, such as video retargeting [30, 47, 55, 70], video completion [27, 34, 39, 41, 42, 69], video texture synthesis [14, 21, 28, 31–33] and more. With the rise of deep-learning, these methods gradually, perhaps unjustifiably, became less popular. Recently, Granot et al. [24] revived classical patch-based approaches for image synthesis, and was shown to significantly outperform *single-image* GANs in both run-time and visual quality.

In light of the above-mentioned deficiencies of single-video GANs, and inspired by [24], we propose a fast and practical method for video generation from a single video that we term VGPNN (*Video Generative Patch Nearest Neighbors*). In order to handle the huge amounts of space-time patches in a single video sequence, we use the classical fast approximate nearest neighbor search method PatchMatch by Barnes et al. [10]. By adding stochastic noise to the process, our approach can generate a large diversity of random different video outputs from a single input video in an unconditional manner.

Like single-video GANs, our approach enables the diverse and random generation of videos. However, in contrast to existing single-video GANs, we can generate *high resolution* videos, while reducing runtime by many orders of magnitude, thus making diverse unconditional video generation from a single video realistically possible for the first time.

In addition to diverse generation from a single video, by employing robust optical-flow based descriptors we use our framework to transfer the dynamics and motions between two videos with different appearance (which we call "video analogies"). We also show the applicability of our framework to spatio-temporal video retargeting and to conditional video inpainting.

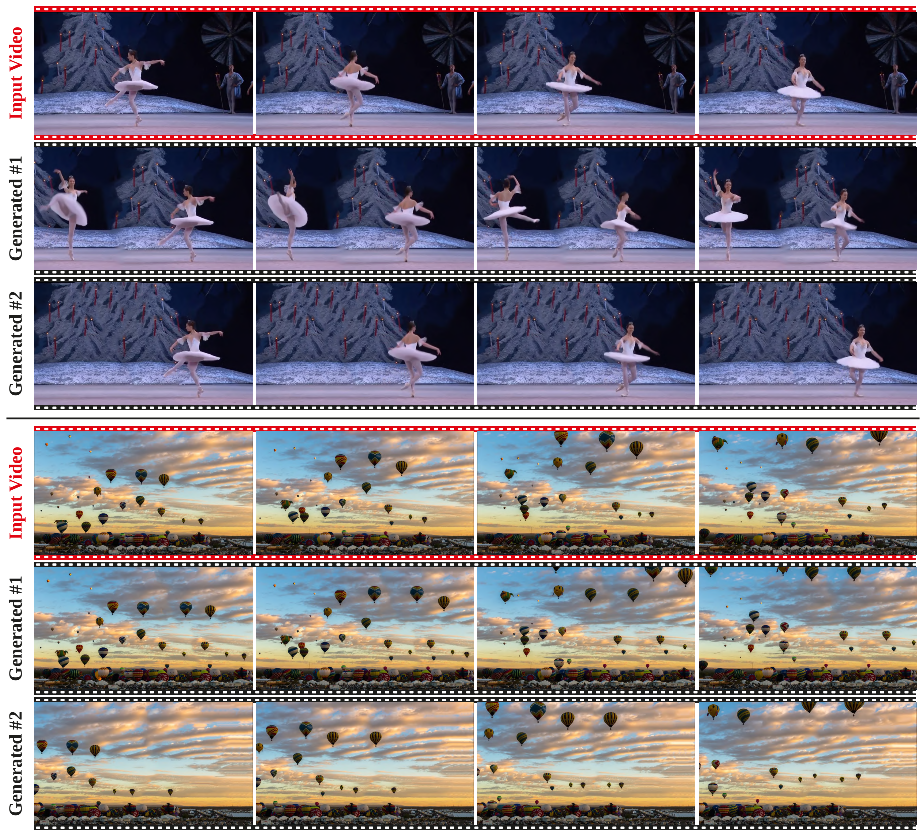

Fig. 2: **Diverse Single Video Generation:** Given an input video (red), we generate similarly looking videos (black) capturing both appearance of objects as well as their dynamics. Note the high quality of our generated videos. *Please watch the full resolution videos in the supplementary material.*

To summarize, our contributions are as follows:

– We show that our space-time patch nearest-neighbors approach, despite its simplicity, outperforms single-video GANs by a large margin, both in runtime and in quality.
– Our approach is the first to generate diverse high resolution videos (spatial or temporal) from a single video.
– We demonstrate the applicability of our framework to other applications: video analogies, sketch-to-video, spatio-temporal video retargeting and conditional video inpainting.

2 Related work

Classical video generation methods, many of whom inspired by similar *image* methods [19, 20, 67], include video texture synthesis [31–33], MRF-based controllable synthesis [51], flow-guided synthesis [14, 28, 33, 43, 49, 50] and more (see surveys by [9, 68]). While some used a generative model to model patch distribution, none of them considered unconditional generation of natural videos, beyond dynamic textures.

Classical methods typically involve comparing and matching of image/video patches. Efficient computation for such matching is therefore critical. *PatchMatch* [10] proposed a fast method for finding an approximate nearest-neighbor field (NNF) between patches of two images A, B. Namely, for each patch in image A, find its nearest neighbor in B. While solving NNF exhaustively takes $O(N^2)$ time (N being the number of patches to match), PatchMatch provides an approximation in $O(N \log N)$ time.

PatchMatch starts by a random guess for the NNF, then iteratively refines it for each patch. The main observation is that since natural images are smooth (as opposed to e.g., noise), w.h.p, the nearest neighbour patches of two (spatially) adjacent patches are also adjacent. Therefore, each patch can refine its own guess by either "peeking" at its neighbor's guess, keeping the current guess or sampling a new guess. At random guess, w.h.p at least one patch has a correct solution, and this is propagated to adjacent patches in further iterations.

Being very efficient, PatchMatch allowed for many applications of image/video manipulation ([9]). It was also extended for k-nearest neighbors search [11], faster search [12] and being differentiably learnable [18]. We use PatchMatch in order to dramatically reduce the running times of video generation from a single video.

The tasks of generation and inpainting are closely related. Both are required to "invent" new content. Wexler et al. [69] (and later extensions [34, 41, 42]) proposed a patch-based method for video completion. Missing space-time patches are replaced by their nearest neighbours from the rest of the video. This is also done in a multi-scale manner by computing a spatio-temporal pyramid. The task is first solved in the coarsest level, and the upsampled result is the initial guess for the next level in the pyramid. Our approach uses a different metric for patch similarity and much deeper spatio-temporal pyramids (higher down-scale ratio). More importantly, we focus on video generation and video analogies and their relation to recent deep learning methods.

3 Method

Our main task is to generate diverse video samples based on a single input video, such that the generated outputs have similar appearance and motions as the original input video, but are also visually different from one another. We want our model to operate on natural input videos that can vary in their appearance and dynamics. In order to capture both spatial and temporal information of a single video, we start by building a spatio-temporal pyramid and operate coarse-to-fine to capture the internal statistics of the input video at multiple scales (Fig. 3). This multi-scale approach is extensively used in classical image synthesis

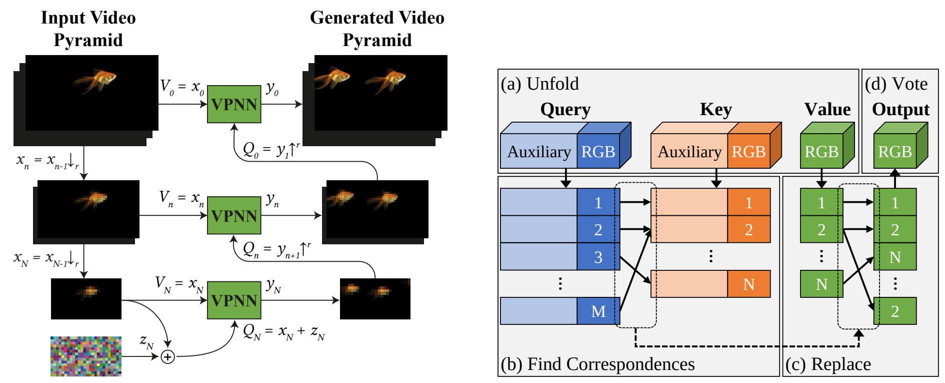

Fig. 3: **VGPNN Architecture** *Left*: given a single input video x_0, a spatio-temporal pyramid is constructed and an output video y_0 is generated coarse-to-fine. At each scale, VPNN module (*right*) is applied to transfer an initial guess Q_n to the output y_n which shares the same space-time patch distribution as the input x_n. At the coarsest scale, noise is injected to induce spatial and temporal randomness. *Right*: VPNN module gets as input query, key and value RGB videos (QKV respectively) and outputs an RGB video. Q and K can be concatenated to additional auxiliary channels. (a) Inputs are unfolded to patches (each position holds a concatenation of neighboring positions); (b) Each patch in Q finds its nearest neighbor patch in K. This is achieved by solving the NNF using PatchMatch [10]; (c) Each patch in Q is replaced with a patch from V, according to the correspondences found in stage (b); (d) Resulting patches are "folded" back to an RGB video output (using the *median* of all suggested votes).

methods as well as in modern GANs [e.g., 25, 29, 52]). At each scale we employ a Video-Patch-Nearest-Neighbor module (*VPNN*); VGPNN is in fact a sequence of VPNN layers. The inputs to each layer depend on the application, where we first focus on our main application of diverse video generation (see Sec. 5 for the specific details of the other applications).

Multi-scale approach: Given an input video x, we construct a spatio-temporal pyramid $\{x_0 \dots, x_N\}$, where $x_0 = x$, and $x_n = x_{n-1}\downarrow_r$ is a bicubically down-scaled version of x_{n-1} by factor r ($r = (r_H, r_W, r_T)$, where $r_H = r_W$ are the spatial factors and r_T is the temporal factor, which can be different).

At the coarsest level, the input to the first VPNN layer is an initial coarse guess of the output video. This is created by adding random Gaussian noise z_N to x_N. The noise z_N promotes high diversity in the generated output samples from the single input. The global structure (e.g., a head is above the body) and global motion (e.g., humans walk forward), is prompted by x_N, where such structure and motion can be captured by *small space-time* patches. The std of the noise z_n is much larger than that of x_n and is related to the typical distance between neighbouring patches in the video (for full details see our supplementary).

Each space-time patch of the initial coarse guess $(x_N + z_N)$ is then replaced with its nearest neighbor patch from the corresponding coarse input x_N. The coarsest-level output y_N is generated by choosing at each space-time position the median of all suggestions from neighboring patches (known as "voting" or "folding").

At each subsequent level, the input to the VPNN layer is the bicubically-upscaled output of the previous layer $(y_{n+1} \uparrow^r)$. Each space-time patch is replaced with its nearest neighbor patch from the corresponding input x_n (using the same patch-size as before, now capturing finer details). This way, the output y_n in each level is similar in structure and in motion to the initial guess, but contains the same space-time patch statistics of the corresponding input x_n. The output y_n is generated by median voting as described above.

To further improve the quality and sharpness of the generated output at each pyramid level (y_n), we iterate several times through the current level, each time using the current output y_n as input to the current VPNN layer (similar to the EM-like approach employed in many patch-based works [e.g., 10, 24, 55, 69]). Full implementation details (e.g., parameters of noise, pyramid, EM-iterations, etc.) are found in the supplementary material.

QKV scheme: In several cases it is necessary to compare patches in another search space than the original RGB input space. To this end we adopt a QKV scheme (query, key and value, respectively) as used by [24]. We denote $V = x_n$ (the corresponding level from the pyramid of the original video) and $Q = y_{n+1} \uparrow$ (the upscaled output of previous layer). Note that since Q is an *upscaled* version of previous output, its patches are blurry. Seeking their nearest neighbors in V (whose patches are sharp) often results in improper matches. This is mitigated by setting $K = x_{n+1}\uparrow^r$ (in the first iteration of each level), which has a similar degree of blur/degradation as Q. After finding its match in K_j, each patch Q_i is then replaced with a patch V_j (where i, j are spatio-temporal positions. Also note that K and V are of the same shape). The QKV scheme is especially important in our video analogies application where it is used to include additional temporal information in the queries and the keys. We discuss it in detail in Sec. 5.1.

Completeness score: In the applications of video analogies, spatio-temporal video retargeting and conditional video inpainting we use the normalized similarity score [24] that encourages visual completeness. The score between a query patch Q_i and a key patch K_j is defined as:

$$S\left(Q_i, K_j\right) := \frac{1}{\alpha + \min_\ell D(Q_\ell, K_j)} D(Q_i, K_j) \tag{1}$$

where D is mean square error, and α controls the degree of completeness (smaller α encourages more completeness). S is essentially a weighted version of D, whose weights depend *globally* on K and Q.

Finding Correspondences: We find the nearest neighbors between Q and K (Fig. 3right-b) using PatchMatch (Barnes et al. [10]). To cope with the completeness score, we apply PatchMatch twice. First we find a "rareness" score for the keys - for each *key* we find its closest *query*. Then, for each *query* we find its closest *key* while factoring in the rareness of the keys as weights in the PatchMatch search. Namely, we solve for:

$$\mathrm{NNF}(\mathbf{p}) = \arg\min_{\mathbf{v}} W(\mathbf{p}+\mathbf{v}) \cdot D(Q(\mathbf{p}), K(\mathbf{p}+\mathbf{v})) \tag{2}$$

where D is a distance function, W are per-patch weights, $\mathbf{p} = (t, x, y)$ a position in Q and $\mathbf{v} = (t', x', y')$ are possible NNF candidates (such as the NNF at the current position $\mathrm{NNF}(t, x, y)$ or at a neighbor position $\mathrm{NNF}(t, x-1, y)$ in the propagation step).

This requires a slight modification of PatchMatch to support per-key weights. This additional support makes it possible to approximately solve Eq. 1 with two passes of PatchMatch. Even though this gives an approximation of Eq. 1, we do not suffer loss in quality or lack of completeness, as apparent from our results.

The algorithm is implemented on GPU using PyTorch [44], with time complexity $O(n \times d)$ and $O(n)$ additional memory (where n is the video size and d is the patch size; also see Fig. 5).

Temporal Diversity and Consistency: A simple but effective trick to enhance the temporal diversity of our samples is to generate outputs with less frames than in the input video. Generating samples with similar number of frames as in the input video result in outputs that are "in sync". Intuitively, the motion of the input video is the only motion that is coherent for this amount of frames. By generating shorter videos allow for shorter motions from different times in the input video, to occur simultaneously in the generated outputs (see for example how the generated dancers in Fig. 2 are not "synced"). We also found that the temporal consistency is best preserved in the generated output when the initial noise z_N is randomized for each spatial position, but is the same (replicated) in the temporal dimension.

4 Experimental Results

In this section we evaluate and compare the performance of our main application – diverse video generation from a single input video. Figs. 1 and 2 illustrate diverse videos generated from a single input video, all sharing the same space-time patch distribution. The diversity is both spatially (e.g., number of dancers and their positions are different from the input video) and temporally (generated dancers are not synced). **Please refer to the supplementary material** to view the full resolution videos and many more examples.

Evaluation of video generation from a single video: We compare our results to recently published methods for diverse video generation from single video: HP-VAE-GAN [25] and SinGAN-GIF [5]. We show that our results are both qualitatively and quantitatively superior while reducing the runtime by a factor of 3×10^4 (from 8 days training on one video to 18 seconds for new generated video). Since SinGAN-GIF did not make their code available, and the training time of HP-VAE-GAN for a single video is roughly 8 days, we are only able to compare to the videos published by these methods (we use all published videos).

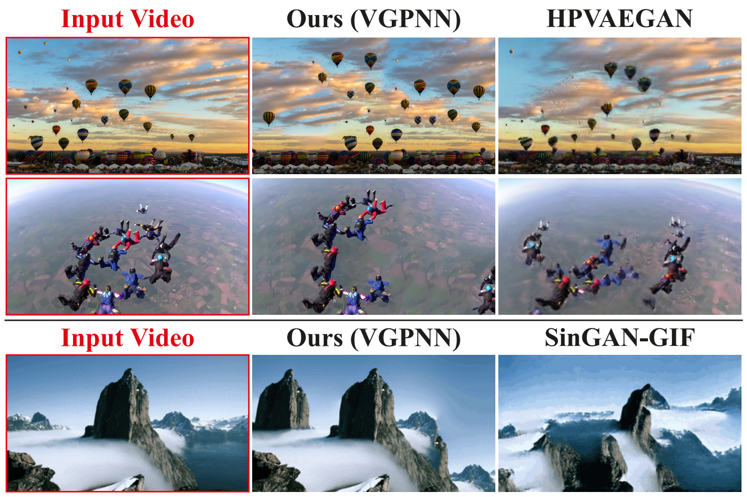

Fig. 4: **Comparing Visual Quality** between our generated frames and those of HP-VAE-GAN [25] and SinGAN-GIF [5] (please **zoom in** on the frames). Note that our generated frames are sharper and also exhibit more coherent and plausible arrangements of the scene. For details see Section 4. See supplementary for full videos and more comparisons.

Evaluation set: "HP-VAE-GAN dataset" comprises of 10 input videos with 13 frames each, and of spatial resolution of 144×256 pixels. "SinGAN-GIF dataset" has 5 input videos with maximal resolution of 168×298 pixels and 8-16 frames.

Qualitative comparison: In Fig. 4 we show a side-by-side comparison of representative generated frames of our method to frames generated by HP-VAE-GAN [25] and SinGAN-GIF [5]. Note that while [5, 25] are limited to generated outputs of small resolution (144×256), we can generate outputs in the same resolution of the input video (full-HD 1280×1920, shown in the figure). The full videos (as well as a comparison to our generated outputs of similar low resolution) can be viewed in the supplementary material. As can be seen, our generated samples (in low and high resolution) are more spatially and temporally coherent, as well as having higher visual quality. It is evident that generating videos using the space-time patches of the original input video, rather than regressing output RGB values, gives rise to high quality outputs.

Quantitative comparison: In Table 1 we report the Single-Video-FID (SVFID) [5] of our generated samples, compared to those generated by HP-VAE-GAN [25] and SinGAN-GIF [5][*]. SVFID was proposed by [25] to measure the patch statistics similarity between the input video and a generated video. It computes the Fréchet distance between the statistics of the input video and the generated video using pre-computed C3D [57] features (Lower SVFID is better). As can be seen in Table 1, our generated samples bear more substantial similarity to the

[*]All quantitative comparisons were done on generated samples of the same resolution and video length as that of the other method.

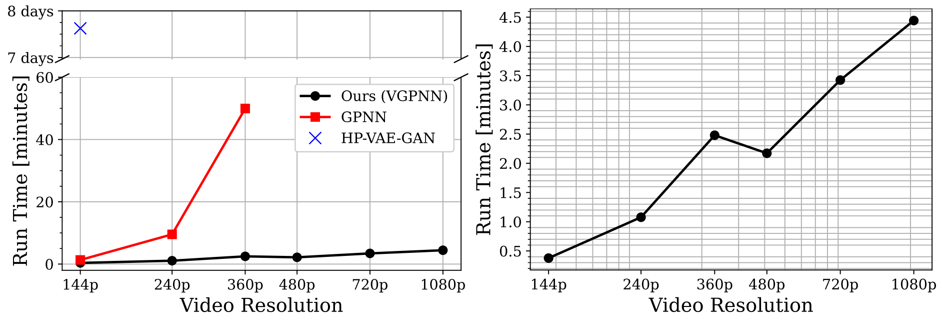

Fig. 5: **Generation Runtime. Left:** Comparing between our approach (VGPNN), a naïve extension of GPNN [24] from 2D to 3D and HP-VAE-GAN [25]. We compared the generation time of 13-frames videos with different spatial resolutions (X-axis). All videos have 16:9 aspect ratio (e.g., 144p is 144x256 and 1080p is 1080x1920 – full-HD). **Right:** Close-up of our generation runtime (black line in left). Our approach takes seconds/minutes to generate low-res/high-res video outputs. The drop in 480p is the result of decreasing the patch-size in finer-levels of high-res videos. See Section 4 for details, and supplementary for implementation details.

input videos (indicated by lower SVFID). [52] proposed a diversity index to make sure that generated outputs are indeed different (and not simply "copying" the input). We adapt the index for videos. The index is zero if all generated outputs are the same, and higher otherwise. While our and HP-VAE-GAN generated samples have similar index (0.45/0.41 respectively), those of SinGAN-GIF have higher index (0.86 vs. our 0.6). Such high diversity is not an advantage, when paired with SVFID about twice worse than ours. It stems from low quality appearance with out-of-distribution patches. All inputs and generated videos can be found in the supplementary material.

User study: We conducted a user study evaluation using Amazon Mechanical Turk (AMT). For each dataset, 100 subjects were shown multiple pairs of videos, each consisting of a video generated by our method, and a video generated by the other method (both were generated from the same input video). The subjects were asked to judge which sample is better in terms of sharpness, natural look and coherence. In Table 1 we report the percentage of users who favored our method over the other. Compared to videos generated from HP-VAE-GAN dataset, there is a clear preference in favor of our patch-based method. The results on the SinGAN-GIF dataset are not that clear-cut, this might be due to the somewhat restricted nature of the videos in that particular dataset (as mentioned above, it was not possible to check SinGAN-GIF on other samples, since the authors did not publish their code, nor stated the amount of time it took to generate their samples).

Method	**SVFID** [25] ↓	**Head-on comparison** (User study) [%]↑	**Runtime** ↓
HP-VAE-GAN [25]	0.0081	**67.84**±1.77	7.625 days
VGPNN (Ours)	**0.0072**		**18** secs
SinGAN-GIF [5]	0.0119	**50.57**± 3.27	Unpublished
VGPNN (Ours)	**0.0058**		**10** secs

Table 1: **Quantitative Evaluation:** A comparison of our generated video samples to that of HP-VAE-GAN [25] and SinGAN-GIF [5], conducted on input videos provided in their papers. Our diverse samples have more resemblance to the input videos (indicated by lower SVFID). In a user study, users scored in favor of our method (see Section 4 for details).

Reducing running times: In Fig. 5 we show a comparison of the runtime taken to generate random video samples using our method, compared to a naïve extension of GPNN [24] (from 2D to 3D patches) and compared to the training time of HP-VAE-GAN [25]. As discussed in Sec. 3, the use of efficient PatchMatch algorithm for nearest neighbors search, as opposed to the exhaustive search done in GPNN, dramatically reduces both run time and memory footprint used for video generation, making it possible to generate high-resolution videos (including Full-HD 1080p). All experiments were conducted on Quadro RTX 8000 GPU.

5 Applications

Other than unconditional diverse generation, we demonstrate the utility of our framework on several other video manipulation applications.

5.1 Video Analogies

Video to video translation methods typically train on large datasets and are either conditioned on human poses or keypoint detection [e.g. 15, 40, 63–65], or require knowledge of a human/animal model [e.g. 1, 16, 37, 46, 53, 54, 71]. We show that when videos' dynamics are similar in both their motion and semantic context within their video, one can use our framework to transfer the motion and appearance between the two (see Fig. 6). We term this task "video analogies" (inspired by *image analogies* [13, 26, 38]). More formally, we generate a new video whose spatio-temporal layout is taken from a content video C, and overall appearance and dynamics from a style video S.

Our goal is to find a mapping of dynamic elements (3D patches) between the two videos, which can be very different in their appearance (RGB space). This is achieved by using the magnitude of the optical flow (extracted via RAFT [56]), quantized into few bins (using k-means)[†]. We term this the *dynamic structure* of the video. By concatenating the dynamic structure to the RGB values of the video (along the channels axis), each patch can now be compared using its RGB values and its dynamic values. This provides a good mapping between the dynamic elements of the two input videos.

[†]Each cluster has an integer cluster index. We divide each index by the total number of clusters/bins to be in $[0, 1]$

We compute spatio-temporal pyramids from the style video S, as well as from the dynamic structure of the content video $\mathrm{dyn}(C)$, and the dynamic structure the style video $\mathrm{dyn}(S)$. The output video is generated by setting Q, K, V at each level as follows:

Level	Q	K	V
N (coarsest)	$\mathrm{dyn}(C)_N$	$\mathrm{dyn}(S)_N$	S_N
n (any other)	$\mathrm{dyn}(C)_n \| Q_{n+1} \uparrow$	$\mathrm{dyn}(S)_n \| S_n$	S_n

where $\|$ denotes concatenation along the channels axis, and n denote the current level in the pyramid. Note that in the coarsest level, the two videos are only compared by their dynamic structure. In finer levels, the dynamic structure of C (the content video) is used to "guide" the output to the desired spatio-temporal layout.

In Fig. 1 we show a snapshot of the analogies between a waterfall and a lava stream, and in Fig. 6 we show snapshots of the analogies of all possible pairs between three videos (the lava stream, a waterfall and a meat grinder). *The full videos are in the supplementary material.*

We can use the above mentioned mechanism for "sketch-to-video" transfer, where the dynamic structure is given by a sketch video instead of an actual video. See Fig. 6 for a few snapshots of transfering the motion of morphed MNIST [35] digits to a video of marching soldiers, and *please see the full videos and many more results in the supplementary material.*

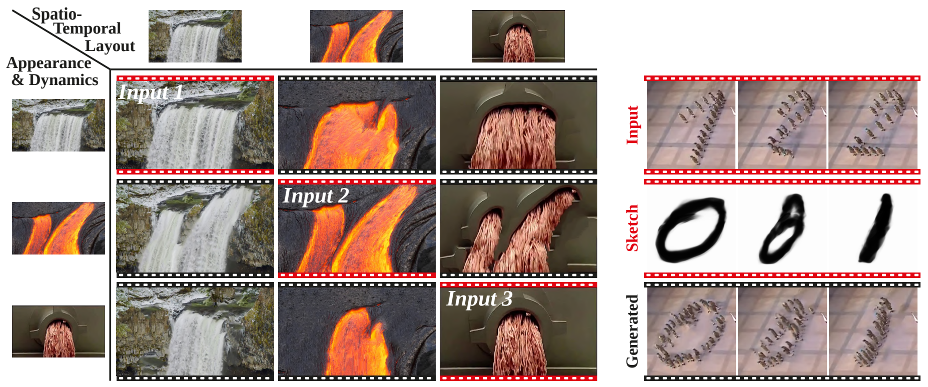

Fig. 6: **Video Analogies:** *Left:* an example of video analogies between all pairs of three input videos (red). Each generated video (black) takes the spatio-temporal layout from the input video in its row, and the appearance and dynamics of the input video from its column. *Right:* an example of sketch-to-video – the generated video (bottom) takes its spatio-temporal layout from the sketch video of morphed MNIST digits (middle) and its appearance and dynamics from the input video of parading soldiers (top). *Please find full videos and additional examples in the supplementary material.*

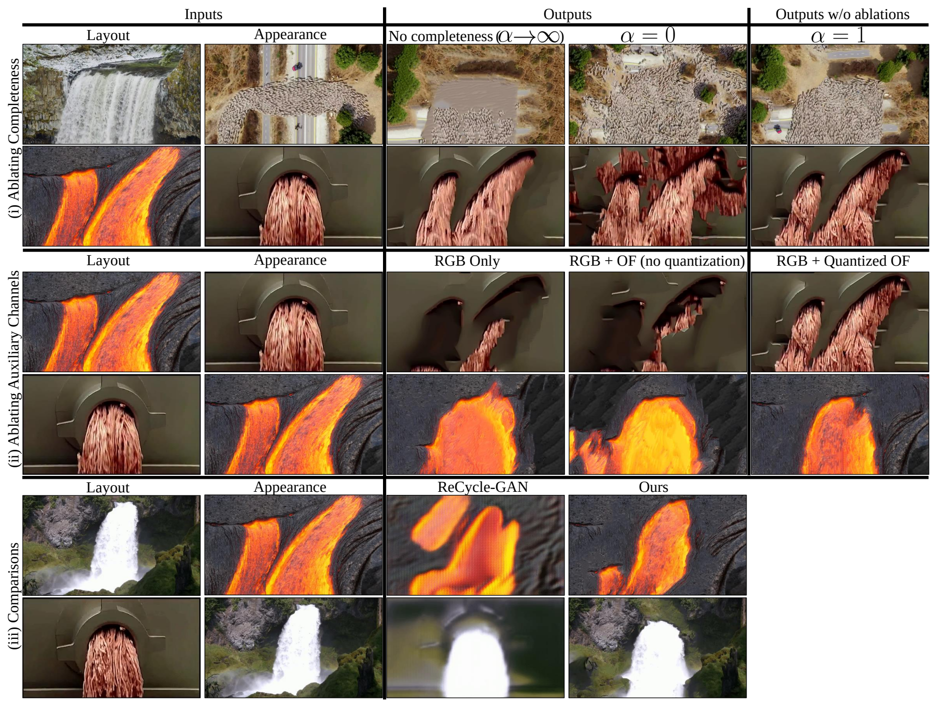

Fig. 7: **Video Analogies:** (i) Ablations for the choice of α (completeness score); (ii) Ablations of choice of auxiliary channel; (iii) Comparisons our results to that of ReCycle-GAN [8].

Flow-based appearance transfer of fluids has been studied by [14, 28, 33, 43, 49, 50]. Most similar to us is [28] that uses a patch nearest neighbor approach to transfer the appearance of a fluid exemplar (a still image) into a video given a human annotated flow+alpha mask. Our method differs in how we model the flow guidance and in the mapping we have between two flows of two videos (instead of a still image exemplar). Also similar to us is Recycle-GAN by Bansal et al. [8] that pose unsupervised video-to-video translation as a domain transfer problem (each video is a domain). They train convolutional encoders to map between the two videos using adversarial loss with cyclic constraints.

In Fig. 7 we compare our results to [8]. As can be seen, [8] results generally fail to converge to the visual quality of the original inputs (partially due to the difficulty of training a parameteric model on small amounts of data), and in many cases converge to the input style video (probably due to the instability of training a GAN).

In Fig. 7 we show ablations for the main parameters used for video analogies. Fig. 7(i) we show the role of using the completeness term α (see Eq. 1). No completeness term (equivalent to $\alpha \rightarrow \infty$) results in over smoothed outputs due to many patches in Q being mapped to a similar patch in K. On the other hand, too “strong” completeness term results in many undesired visual details. We found $\alpha = 1$ to be a good balance for our results. In Fig. 7(ii) we ablate the use of quantized optical flow in the auxiliary channel. Without employing temporal features (RGB only) or using optical flow without quantizing to $[0-1]$, the resulting mapping fails to match similar dynamical elements to those with similar semantics in the context of their video. *Please find full videos in the supplementary material.*

5.2 Spatial Retargeting

The goal of video retargeting is to change the dimensions of a video without distorting its visual contents (e.g., fit a portrait video to a wide screen display). It can be performed in a very similar manner to our video generation described in Sec.3. Given a target shape, we first resize (bicubically) the input video to the target shape, then compute two pyramids (for the input and resized videos) with the same depth and downscale factor. The initial guess at the coarsest level Q_N would be the coarsest level of the resized pyramid (without any additional noise). We then compute the rest of the output video in the same manner as in Sec.3. Note that at each level, V_n are unchanged, hence no distortion is introduced to the patches reconstructing the retargeted video.

As can be seen in Fig. 1 and in the supplementary material, the results preserve the original size and aspect ratio of objects from the input videos while keeping the overall appearance coherent even though the aspect ratio is significantly altered. The dynamics and motions in the videos are also preserved. For instance, the balloons are not “squashed” but rather packed more compactly in the sky and more members were added to the choir instead of stretching them. Nevertheless, the motion of the balloons or the sway of the choir members are preserved. Other classical works for video retargeting, such as [30, 47, 55, 70] did not make their implementation available, therefore we were unable to provide a comparison.

5.3 Temporal Retargeting

Similar to spatial retargeting, one can generate a realistic video with a different *temporal* length. One possible use is generating a shorter summary of the video. While most deep video summarization techniques are achieved by selecting a subset of frames (see survey [4]), classical methods have demonstrated summaries that consist of *novel frames* in which dynamics that are originally sequential can be parallelized or vice versa [45, 55]. By applying the retargeting approach to the temporal dimenstion, we are able to generate summaries with novel frames. The temporal retargeting section in the supplementary material shows several examples. For example, in the dog training summarized video, the trainer and dog turn around simultaneously as opposed to sequentially in the original video. Moreover, we can, in a similar manner, extend the temporal duration of a video creating longer dynamics while preserving the speed of the individual actions.

In the ballet dancer video for example, the choreography is longer, but the pace of the dance motions remains the same.

5.4 Video Conditional Inpainting

In this task we are given an input video with some occluded space-time volume, where the missing parts should be completed based on crude color cues placed by the user in the occluded space (similar to conditional image inpainting [24]). Here we set the number of levels in the pyramid such that the occluded part in the coarsest level is roughly in the size of a single patch. The masked part is then coherently reconstructed using other space-time patches of similar colors to that of the cue. In finer levels, details and dynamic elements are added correctly. The conditional inpainting section in the supplementary material shows how different cues are completed with different elements from the non-occluded parts. See for instance, how a blue cue will be replaced by a player from Barcelona while a white cue by a player from Real Madrid. See more examples in the supplementary material.

6 Limitations

Generation of local patches lacks high-level semantic or global geometric understanding of the scene. For example, This is apparent when scenes with significant depth variations are introduced with large camera motion. While each frame is plausible, different patches are not being transformed consistently, resulting in non-rigid deformations to entities that are realistically rigid. See the generated videos of mountains in the supplementary.

7 Conclusion

We demonstrated that random diverse video generation from a single video can be efficiently done by simple patch-based methods. We also demonstrated how small modifications to our framework give rise to other tasks such as video analogies and spatio-temporal retargeting. We showed that our non-parametric approach outperforms existing single-video GANs in the visual quality of the generated outputs, while being orders of magnitude faster. The low run time required for generating videos using our approach makes it a good baseline for future works in the field.

Acknowledgements This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 788535), from the D. Dan and Betty Kahn Foundation, and from the Israel Science Foundation (grant 2303/20). Dr. Bagon is a Robin Chemers Neustein AI Fellow.

Bibliography

[1] Kfir Aberman, Yijia Weng, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Unpaired motion style transfer from video to animation. *ACM Transactions on Graphics (TOG)*, 39(4):64–1, 2020.

[2] Sandra Aigner and Marco Körner. Futuregan: Anticipating the future frames of video sequences using spatio-temporal 3d convolutions in progressively growing gans. *arXiv preprint arXiv:1810.01325*, 2018.

[3] Emre Aksan and Otmar Hilliges. Stcn: Stochastic temporal convolutional networks. *arXiv preprint arXiv:1902.06568*, 2019.

[4] Evlampios Apostolidis, Eleni Adamantidou, Alexandros I Metsai, Vasileios Mezaris, and Ioannis Patras. Video summarization using deep neural networks: A survey. *arXiv preprint arXiv:2101.06072*, 2021.

[5] Rajat Arora and Yong Jae Lee. Singan-gif: Learning a generative video model from a single gif. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1310–1319, 2021.

[6] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017.

[7] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.

[8] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. In *Proceedings of the European conference on computer vision (ECCV)*, pages 119–135, 2018.

[9] Connelly Barnes and Fang-Lue Zhang. A survey of the state-of-the-art in patch-based synthesis. *Computational Visual Media*, 3(1):3–20, 2017.

[10] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[11] Connelly Barnes, Eli Shechtman, Dan B Goldman, and Adam Finkelstein. The generalized patchmatch correspondence algorithm. In *European Conference on Computer Vision*, pages 29–43. Springer, 2010.

[12] Connelly Barnes, Fang-Lue Zhang, Liming Lou, Xian Wu, and Shi-Min Hu. Patchtable: Efficient patch queries for large datasets and applications. *ACM Transactions on Graphics (ToG)*, 34(4):1–10, 2015.

[13] Sagie Benaim, Ron Mokady, Amit Bermano, and L Wolf. Structural analogy from a single image pair. In *Computer Graphics Forum*, volume 40, pages 249–265. Wiley Online Library, 2021.

[14] Kiran S Bhat, Steven M Seitz, Jessica K Hodgins, and Pradeep K Khosla. Flow-based video synthesis and editing. In *ACM SIGGRAPH 2004 Papers*, pages 360–363. 2004.

[15] Andreas Blattmann, Timo Milbich, Michael Dorkenwald, and Björn Ommer. ipoke: Poking a still image for controlled stochastic video synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14707–14717, 2021.

[16] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5933–5942, 2019.

[17] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, pages 1174–1183. PMLR, 2018.

[18] Shivam Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4384–4393, 2019.

[19] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001.

[20] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999.

[21] Jakub Fišer, Ondřej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sỳkora. Stylit: illumination-guided example-based stylization of 3d renderings. *ACM Transactions on Graphics (TOG)*, 35(4): 1–11, 2016.

[22] Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. Stochastic latent residual video prediction. In *International Conference on Machine Learning*, pages 3233–3246. PMLR, 2020.

[23] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[24] Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, and Michal Irani. Drop the gan: In defense of patches nearest neighbors as single image generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13460–13469, 2022.

[25] Shir Gur, Sagie Benaim, and Lior Wolf. Hierarchical patch vae-gan: Generating diverse videos from a single sample. *arXiv preprint arXiv:2006.12226*, 2020.

[26] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001.

[27] Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. Temporally coherent completion of dynamic video. *ACM Transactions on Graphics (TOG)*, 35(6):1–11, 2016.

[28] Ondřej Jamriška, Jakub Fišer, Paul Asente, Jingwan Lu, Eli Shechtman, and Daniel Sỳkora. Lazyfluids: appearance transfer for fluid animations. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015.

[29] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[30] Philipp Krähenbühl, Manuel Lang, Alexander Hornung, and Markus Gross. A system for retargeting of streaming video. In *ACM SIGGRAPH Asia 2009 papers*, 2009.

[31] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *Acm transactions on graphics (tog)*, 22(3):277–286, 2003.

[32] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. In *ACM SIGGRAPH 2005 Papers*, pages 795–802. 2005.

[33] Vivek Kwatra, David Adalsteinsson, Theodore Kim, Nipun Kwatra, Mark Carlson, and Ming Lin. Texturing fluids. *IEEE transactions on visualization and computer graphics*, 13(5):939–952, 2007.

[34] Thuc Trinh Le, Andrés Almansa, Yann Gousseau, and Simon Masnou. Motion-consistent video inpainting. In *2017 IEEE international conference on image processing (ICIP)*, pages 2094–2098. IEEE, 2017.

[35] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[36] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

[37] Jessica Lee, Deva Ramanan, and Rohit Girdhar. Metapix: Few-shot video retargeting. *arXiv preprint arXiv:1910.04742*, 2019.

[38] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *arXiv preprint arXiv:1705.01088*, 2017.

[39] Ming Liu, Shifeng Chen, Jianzhuang Liu, and Xiaoou Tang. Video completion via motion guided spatial-temporal global optimization. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 537–540, 2009.

[40] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. *arXiv preprint arXiv:2007.08509*, 2020.

[41] Alasdair Newson, Andrés Almansa, Matthieu Fradet, Yann Gousseau, and Patrick Pérez. Towards fast, generic video inpainting. In *Proceedings of the 10th European Conference on Visual Media Production*, pages 1–8, 2013.

[42] Alasdair Newson, Andrés Almansa, Matthieu Fradet, Yann Gousseau, and Patrick Pérez. Video inpainting of complex scenes. *Siam journal on imaging sciences*, 7(4):1993–2019, 2014.

[43] Makoto Okabe, Ken Anjyo, Takeo Igarashi, and Hans-Peter Seidel. Animating pictures of fluid using video examples. In *Computer Graphics Forum*, volume 28, pages 677–686. Wiley Online Library, 2009.

[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle,

A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[45] Alex Rav-Acha, Yael Pritch, and Shmuel Peleg. Making a long video short: Dynamic video synopsis. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 435–441. IEEE, 2006.

[46] Jian Ren, Menglei Chai, Sergey Tulyakov, Chen Fang, Xiaohui Shen, and Jianchao Yang. Human motion transfer from poses in the wild. In *European Conference on Computer Vision*, pages 262–279. Springer, 2020.

[47] Michael Rubinstein, Ariel Shamir, and Shai Avidan. Improved seam carving for video retargeting. *ACM transactions on graphics (TOG)*, 27(3):1–9, 2008.

[48] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *Proceedings of the IEEE international conference on computer vision*, pages 2830–2839, 2017.

[49] Syuhei Sato, Yoshinori Dobashi, Theodore Kim, and Tomoyuki Nishita. Example-based turbulence style transfer. *ACM Transactions on Graphics (TOG)*, 37(4):1–9, 2018.

[50] Syuhei Sato, Yoshinori Dobashi, and Tomoyuki Nishita. Editing fluid animation using flow interpolation. *ACM Transactions on Graphics (TOG)*, 37(5):1–12, 2018.

[51] Arno Schödl, Richard Szeliski, David H Salesin, and Irfan Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498, 2000.

[52] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4570–4580, 2019.

[53] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2377–2386, 2019.

[54] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in Neural Information Processing Systems*, 32:7137–7147, 2019.

[55] Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. Summarizing visual data using bidirectional similarity. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[56] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[57] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[58] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

[59] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*, 2017.

[60] Ruben Villegas, Dumitru Erhan, Honglak Lee, et al. Hierarchical long-term video prediction without supervision. In *International Conference on Machine Learning*, pages 6038–6046. PMLR, 2018.

[61] Ruben Villegas, Arkanath Pathak, Harini Kannan, Dumitru Erhan, Quoc V Le, and Honglak Lee. High fidelity video prediction with large stochastic recurrent neural networks. *arXiv preprint arXiv:1911.01655*, 2019.

[62] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *arXiv preprint arXiv:1609.02612*, 2016.

[63] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018.

[64] Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. Few-shot video-to-video synthesis. *arXiv preprint arXiv:1910.12713*, 2019.

[65] Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. Imaginator: Conditional spatio-temporal gan for video generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1160–1169, 2020.

[66] Yaohui Wang, Francois Bremond, and Antitza Dantcheva. Inmodegan: Interpretable motion decomposition generative adversarial network for video generation. *arXiv preprint arXiv:2101.03049*, 2021.

[67] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488, 2000.

[68] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*, pages 93–117. Eurographics Association, 2009.

[69] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time video completion. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004.

[70] Lior Wolf, Moshe Guttmann, and Daniel Cohen-Or. Non-homogeneous content-driven video-retargeting. In *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV)*, 2007.

[71] Zhuoqian Yang, Wentao Zhu, Wayne Wu, Chen Qian, Qiang Zhou, Bolei Zhou, and Chen Change Loy. Transmomo: Invariance-driven unsupervised video motion retargeting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5306–5315, 2020.