# Controllable Video Generation with Sparse Trajectories

Zekun Hao Xun Huang Serge Belongie
Department of Computer Science & Cornell Tech, Cornell University
{hz472,xh258,sjb344}@cornell.edu

## Abstract

*Video generation and manipulation is an important yet challenging task in computer vision. Existing methods usually lack ways to explicitly control the synthesized motion. In this work, we present a conditional video generation model that allows detailed control over the motion of the generated video. Given the first frame and sparse motion trajectories specified by users, our model can synthesize a video with corresponding appearance and motion. We propose to combine the advantage of copying pixels from the given frame and hallucinating the lightness difference from scratch which help generate sharp video while keeping the model robust to occlusion and lightness change. We also propose a training paradigm that calculate trajectories from video clips, which eliminated the need of annotated training data. Experiments on several standard benchmarks demonstrate that our approach can generate realistic videos comparable to state-of-the-art video generation and video prediction methods while the motion of the generated videos can correspond well with user input.*

## 1. Introduction

The ability to synthesize realistic videos is a hallmark of motion understanding and has a wide range of applications (*e.g.*, video editing, augmented reality, movie and game production). State-of-the-art computer graphics engines are able to synthesize photo-realistic videos, yet they require heavy manual labor of experts and are usually constrained to specific domains. There has been a significant body of recent work that explores alternative solutions based on deep generative neural networks [30, 8, 22, 26, 38, 25, 14, 2, 31, 28], but the quality of the results is still far from satisfactory.

Here, we emphasize that video generation models should support high-level control over the motion to be synthesized in order for it to be useful. In many scenarios, it is important for users to be able to control how objects should move in the generated video. Most existing methods either deterministically generate a single predicted future from some given frames [25, 14, 2, 31, 28], or randomly sample from a distribution of futures without any high-level control over which future to be generated [32, 38, 1].

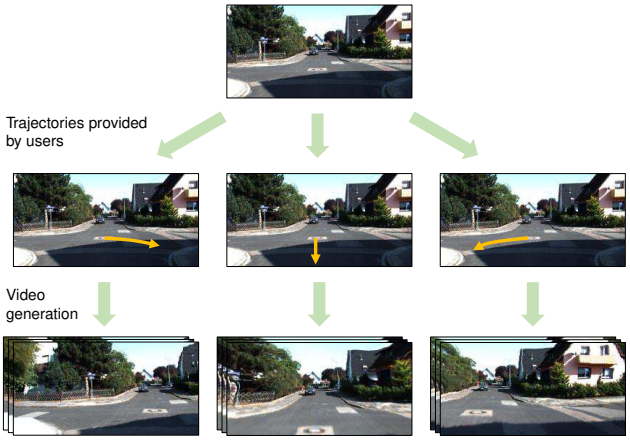



Figure 1. **Video generation conditioned on sparse trajectories**. Given an input image (top row) and three different trajectories provided by human users (middle row). Our model can generate different pixel-level video predictions that exhibit motion corresponding to the input trajectories (bottom row).

In this paper, we aim at improving both the *controllability* and *quality* of video generation. Our model synthesizes a video clip conditioned on a single initial frame that provides object and scene appearance, and some sparse trajectories that encode the desired high-level motion. For example, given the initial image in the top row of Figure 1, users can control which direction the car should move in the video by drawing trajectories (middle row). Our model can then generate videos that exhibit motion corresponding to the input trajectories. Different trajectories lead to different generated videos. The sparse trajectories not only provide a convenient way for human users to guide the synthesized motion, but also improve the generation quality by providing the model with more information, as shown in our following experiments. The trajectory-based manipulation is also general enough to be applied to any types of videos (*e.g.* human actions, robotic arm movements).

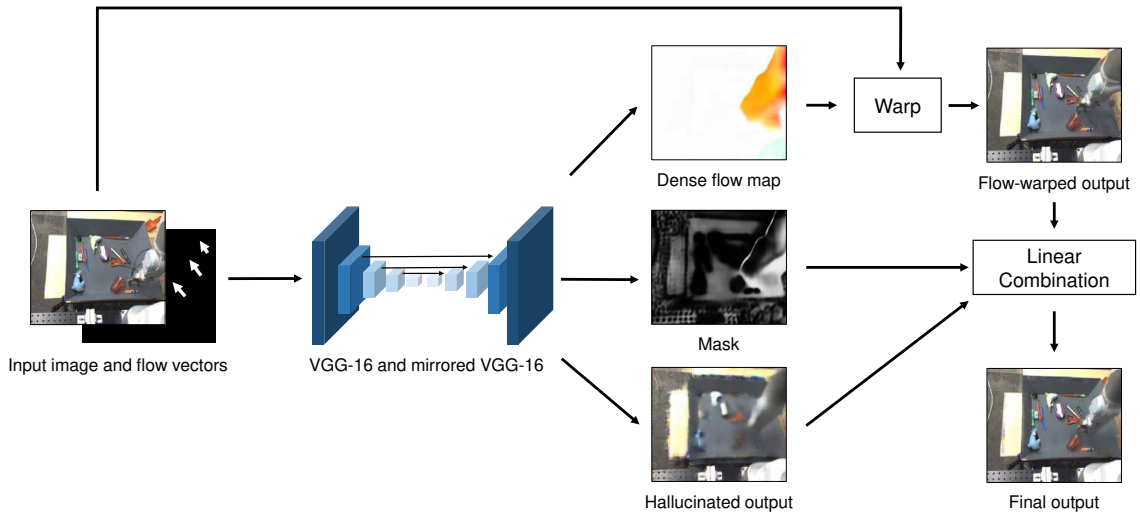



Figure 2. **Overview of our image generation model.** Our model receives a single frame and a set of flow vectors as inputs, and produces an output frame that has moved according to the flow vectors. The neural network in our pipeline serves as two purposes: sparse-to-dense flow completion and hallucination. For flow completion, the network generate the dense flow map from the sparse flow vectors and produces the output frame by warping the input frame with the predicted flow map. For hallucination, the network attempts to directly generate pixels together with a weight map for linearly combining the hallucinated pixels with warped pixels. While this figure only illustrates the process of generating a single frame, we describe how to generate videos with this model in Figure 3.

Still, we find that synthesizing realistic videos is challenging even with the trajectories provided. We observe that most pixels in the output frames can be directly copied from the input frame with only minor difference in RGB value, whereas a few pixels are occluded in the input frame and need to be generated from scratch. Inspired by this observation, we propose a two-stage architecture, in which the first step warps the input frame into an intermediate output with a predicted dense flow, while second step hallucinates pixels that are missing from the input frame and also compensate for color change. Through experiments, we demonstrate that the two steps are complementary to each other, resulting in higher generation quality than using either part alone. We perform extensive experiments on several standard video generation benchmarks to demonstrate the effectiveness of the proposed approach.

## 2. Related work

There has been a vast amount of work on video generation and prediction. Some earlier methods focus on predicting semantic information in the future, such as action categories [12], pedestrian trajectories [11], car trajectories [33], deep CNN features [29], or optical flows [34, 20]. Inspired by advances in generative image modeling [10, 6], recent works start to tackle the problem of raw pixel generation. Some of them focus on unconditional generation of short video clips [30, 8, 22, 26]. For example, Vondrick et al. [30] leverage a GAN [6] framework that separately generates the static background and the foreground motion. Kalchbrenner et al. [8] apply PixelCNN [19, 27] to video generation. Others aim at extrapolating or interpolating videos from a few given frames [38, 25, 14, 2, 31, 28]. However, previous methods are either uncontrollable at all, or only allow domain-specific controls such as robot arm actions [4], player actions in Atari games [18], or human keypoints [28, 35]. There are some concurrent works that propose to guide video generation with natural language descriptions [13, 16]. However, language can be very ambiguous and does not allow exact manipulation. Our work provides a general, precise, and convenient way for human users to control how the video would develop.

Many of the previous video prediction models adopt an encoder-decoder paradigm, in which an encoder encodes the known frames and a decoder predicts the future frames from the encoded features [15, 17, 30, 22, 21]. These approaches usually lead to blurry predictions due to the large ambiguity and high output dimensionality. To alleviate this problem, some recent works propose to directly warp pixels from previous frames [14, 2, 31]. However, this method will struggle to generate pixels that do not appear in the given frames. We propose to combine the strengths of both approaches, which leads to significantly better results.

## 3. Methods

### 3.1. Overview

Figure 2 shows an overview of our approach. The model receives a single image $\mathbf{I} \in \mathbb{R}^{W \times H \times 3}$ and a

sparse flow map $\mathbf{S} \in \mathbb{R}^{W \times H \times 6}$ as inputs, where $W, H$ are the width and height respectively. The sparse flow map $\mathbf{S}$ encodes $N$ flow vectors that represent the desired motion. For each flow vector with start point $(x^i, y^i)$ and displacement $(\Delta x^i, \Delta y^i), i = 1, 2, ..., N$, we fill $\mathbf{S}_{x^i+\lfloor \Delta x^i \rceil, y^i+\lfloor \Delta y^i \rceil, j}, j = 1, 2, 3$ with negative displacement and an indicator: $(-\Delta x^i, -\Delta y^i, 1)$, so it corresponds to the sparse flow from the target frame to the input frame. Since the input trajectories are sparse, the 1 serves as an indicator of the presence of a flow vector at that position. Similarly, we fill $\mathbf{S}_{x^i, y^i, j}, j = 4, 5, 6$ with positive displacement and an indicator: $(\Delta x^i, \Delta y^i, 1)$. All the other values of $\mathbf{S}$ are filled with zeros. Our network learns to produce an output image that has the same content as the given frame, but has moved in accordance with the provided sparse flow. The pipeline consists of two steps: sparse-to-dense flow completion and pixel hallucination, which are detailed in the following subsections.

### 3.2. Sparse-to-dense flow completion

The first branch of our network completes a dense flow map $\mathbf{D} = f(\mathbf{I}, \mathbf{S})$ from sparse flow vectors $\mathbf{S}$ and image $\mathbf{I}$. We then use a differentiable warp operator to transform $\mathbf{I}$ into an intermediate output prediction $\mathbf{O}^f$ according to $\mathbf{D}$. The output pixel at location $(x, y)$ is warped from the input pixel at location $(x + \Delta x, y + \Delta y)$, where $(\Delta x, \Delta y) = \mathbf{D}_{x,y}$. Since $\Delta x$ and $\Delta y$ are fractional in general, we employ bilinear interpolation to estimate the warped value from the neighboring integer locations [40, 14, 39]

$$\mathbf{O}^f_{x,y} = \sum_{x',y'} (1 - |x + \Delta x - x'|)(1 - |y + \Delta y - y'|)\mathbf{I}_{x',y'}$$

where $(x', y')$ are from the 4-pixel neighbors (top-left, top-right, bottom-left, bottom-right) of $(x + \Delta x, y + \Delta y)$. The bilinear sampling operator is locally differentiable, making our system end-to-end trainable.

Since the output frame is usually very similar to the input frame, most output pixels can be directly copied from input pixels. Also, thanks to the spatial consistency of most movements in the real world, the dense flow don't contain much high-frequency component, It is therefore much easier to predict a flow than to generate all the low-level details from scratch, as also observed by previous works [31, 14].

### 3.3. Pixel hallucination

However, the flow-based branch alone can only copy existing pixels and is unable to generate new pixels or handle color change. We introduce a second branch $h$ to "hallucinate" the pixels missing from the given frame and compensate for color change of existing pixels, which complements the flow-based approach. The output $\mathbf{O}^h = h(\mathbf{I}, \mathbf{S})$ is then merged with $\mathbf{O}^{\mathrm{f}}$ via a mask $\mathbf{M} = m(\mathbf{I}, \mathbf{S})$ predicted by another network branch $m$.

$$\mathbf{O} = \mathbf{M} \odot \mathbf{O}^f + (1 - \mathbf{M}) \odot \mathbf{O}^h$$

where $\odot$ denotes element-wise multiplication. Through experiments, we find $\mathbf{M}$ automatically learns to prioritize the flow completion output $\mathbf{O}^{\mathbf{f}}$ wherever possible (*e.g.*, static and not occluded background, moving objects that appear in the input frame), and prioritize $\mathbf{O}^{\mathbf{h}}$ when there is no corresponding pixel in the input image (*e.g.*, background occluded in the input frame, novel objects).

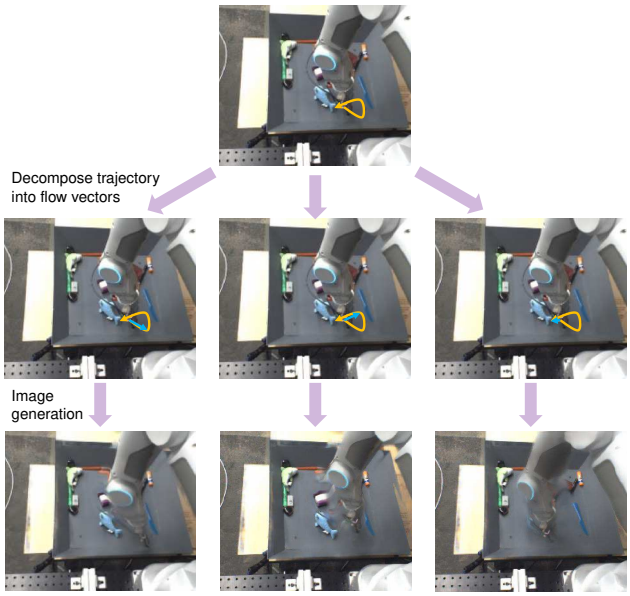



Figure 3. **Generating videos with the image generation model.** Given an input image and a set of trajectories (we only show a single yellow one here for simplicity), we move the end points of the flow vectors (blue vectors) along each trajectory. We then use our image generation model to produce a sequence of frames from the same input image and different flow vectors.

### 3.4. Video generation

The model described above can generate an image from an input image and a set of sparse flow vectors. To generate videos from trajectories, we simply move the end point of the flow vectors along the trajectory (as shown in Figure 3) and generate a sequence of images. Although our goal is to control the model with human input trajectories, training it with human trajectories would require time-consuming annotation effort. We therefore train our model with trajectories automatically extracted from video clips and show that it generalizes to human-provided trajectories at test time.

| | Full model | W/o hallucination | W/o flow completion |
|---|---|---|---|
| Robot Pushing | **87.2 (273)** | 10.9 (34) | 1.9 (6) |
| KITTI | **86.2 (288)** | 11.4 (38) | 2.4 (8) |
| UCF-101 | **79.0 (271)** | 17.8 (61) | 3.2 (11) |

Table 1. **Human evaluation results**. The numbers denote the percentage a model is judged as the most realistic among three models (numbers in parentheses are raw data). The data was collected from 10 volunteers, with each volunteer making 50 to 200 choices on randomly sampled images from randomly sampled datasets. For all three datasets, most users preferred videos generated from our full model. The hallucination part does not produce good results by itself, yet is complementary to the flow completion part.

# 4. Experiments

## 4.1. Datasets

We perform experiments on three datasets commonly used to evaluate video generative models.

**KITTI** [5] is a dataset of videos captured by roof-mounted camera on a car traversing German streets. Following [14, 39], we use the odometry subset that has 22 long video sequences. But instead of doing view synthesis, we perform the more challenging video generation task on this dataset.

**Robotic Pushing** [4] consists of $59,000$ video clips of robotic arm pushing objects on a table. We do not use the provided annotations of robotic arm actions since it's not the focus of our paper.

**UCF-101** [24] contains $13,320$ videos from 101 categories of human actions (*e.g.*, skiing, surfing, boxing). It is the most challenging dataset among the three due to the wide variety of motion, scenes, characters, and background clutter it contains.

## 4.2. Implementation details

**Network architecture.** The down-sample part of the network resembles the convolution layers of VGG-16 network [23]. The network for up-sample part is simply mirrored from the down-sample part with pooling layers replaced by nearest-neighbor upsampling. We also adopted batch normalization [7] and added cross-connection between down-sample and up-sample parts, similar to [14].

**Training.** We randomly sample video clips of length 9 frames as our training data. Our model receives the first frame as input and tries to generate the next 8 frames. We first estimate dense trajectories using the algorithm by Wang *et al*. [36]. To automatically extract sparse yet informative trajectories, we then filter out trajectories that have total displacement smaller than a threshold. After that, we randomly choose $N \in \{1, 2, ..., 5\}$ trajectories from the remaining trajectories. This is to mimic the test scenario where users can input 1 to 5 trajectories. We then use the procedure described in Figure 3 to get 8 sets of sparse flow vectors corresponding 8 future frames as our training samples. We do not perform any video stabilization. We train our model using Adam [9] optimizer with batch size of 32, learning rate of $0.0005$, $\beta_1 = 0.9$, $\beta_2 = 0.99$. We resize all frames such that the largest dimension is 256.

While some recent video generation models employ adversarial training [31, 17, 30, 2], we observe good and stable performance with a simple L1 loss in the pixel space. This could be due to the less uncertainty our model needs to capture given the provided trajectories. Our flow-based generation may also reduce blurriness, as suggested by previous works [14].

## 4.3. Human evaluation on controlled generation

Previous works on video generation are usually evaluated by computing similarity metrics with ground truth frames as references. Metrics such as Peak Signal to Noise Ratio (PSNR) and Structral Similarity (SSIM [37]) are widely used. However, we do not have ground truth frames available for human-input trajectories, so we have to resort to subjective judgment.

We conduct a user study to evaluate the quality and controllability of different variants of our model. We show volunteers a single image and ask them to draw trajectories that represent the motion they would like to see. Volunteers are told to draw 1 to 5 trajectories and that the trajectories have to correspond to realistic motion. We then show users videos generated by three variants of our model: 1) flow completion only, 2) hallucination only, and 3) the proposed model. We train model 1) and 2) with the same architecture and hyper-parameters as our proposed model. The generated videos are also randomly shuffled so that users do not know which video is generated by which model. Finally, we ask user for preferences over the three generated videos, *i.e.*, which video is the most realistic given the input trajectories and image?

Table 1 summarizes the results of our user study on three datasets. The numbers in the table represent the percentage a model output is judged as the most realistic (the num-

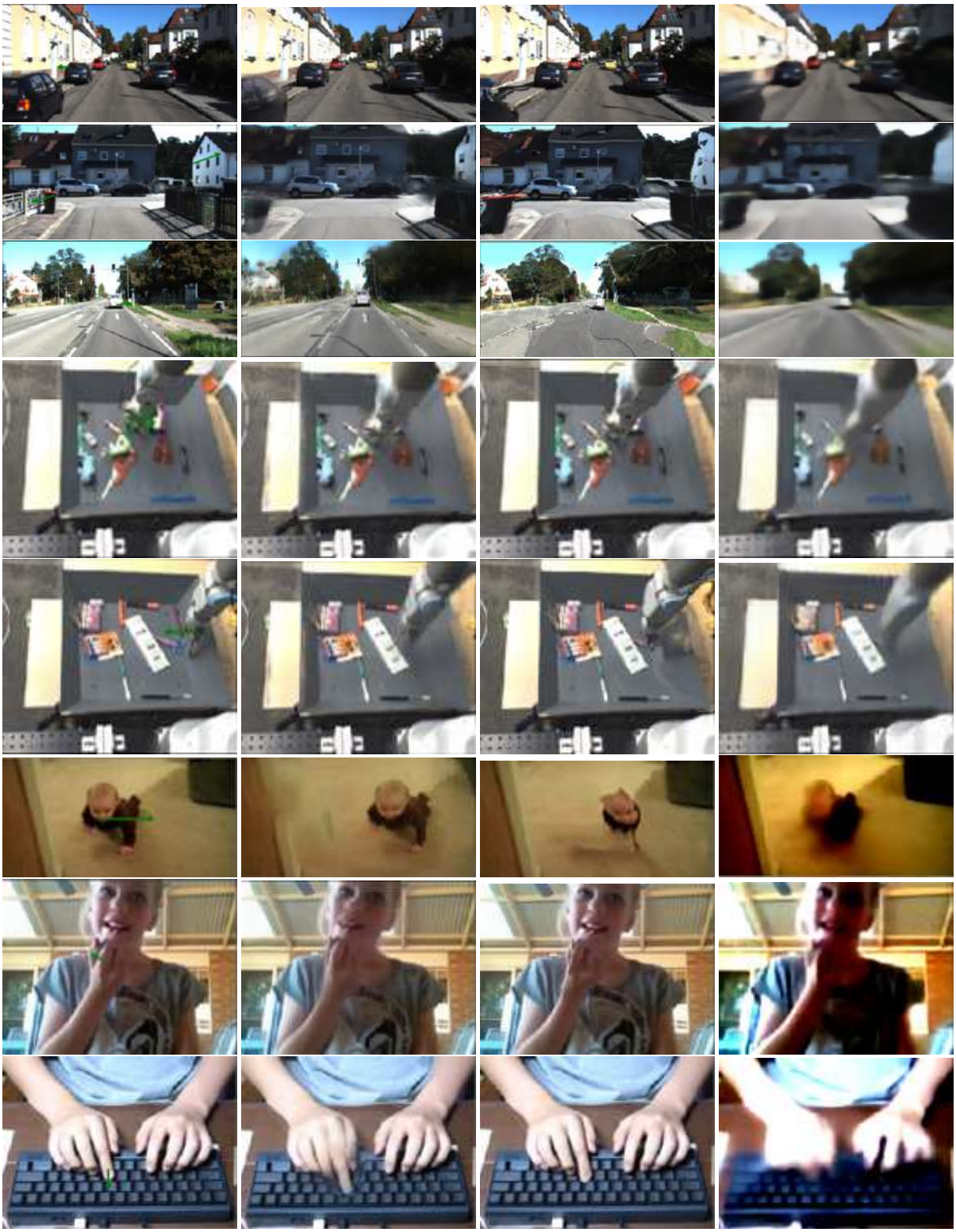



Figure 4. **Example results in user study**. Here we show example results generated by variants of our models during user study. Our full model generates more realistic results in most cases. We only show the last generated frame with corresponding flow vectors due to space constraints.

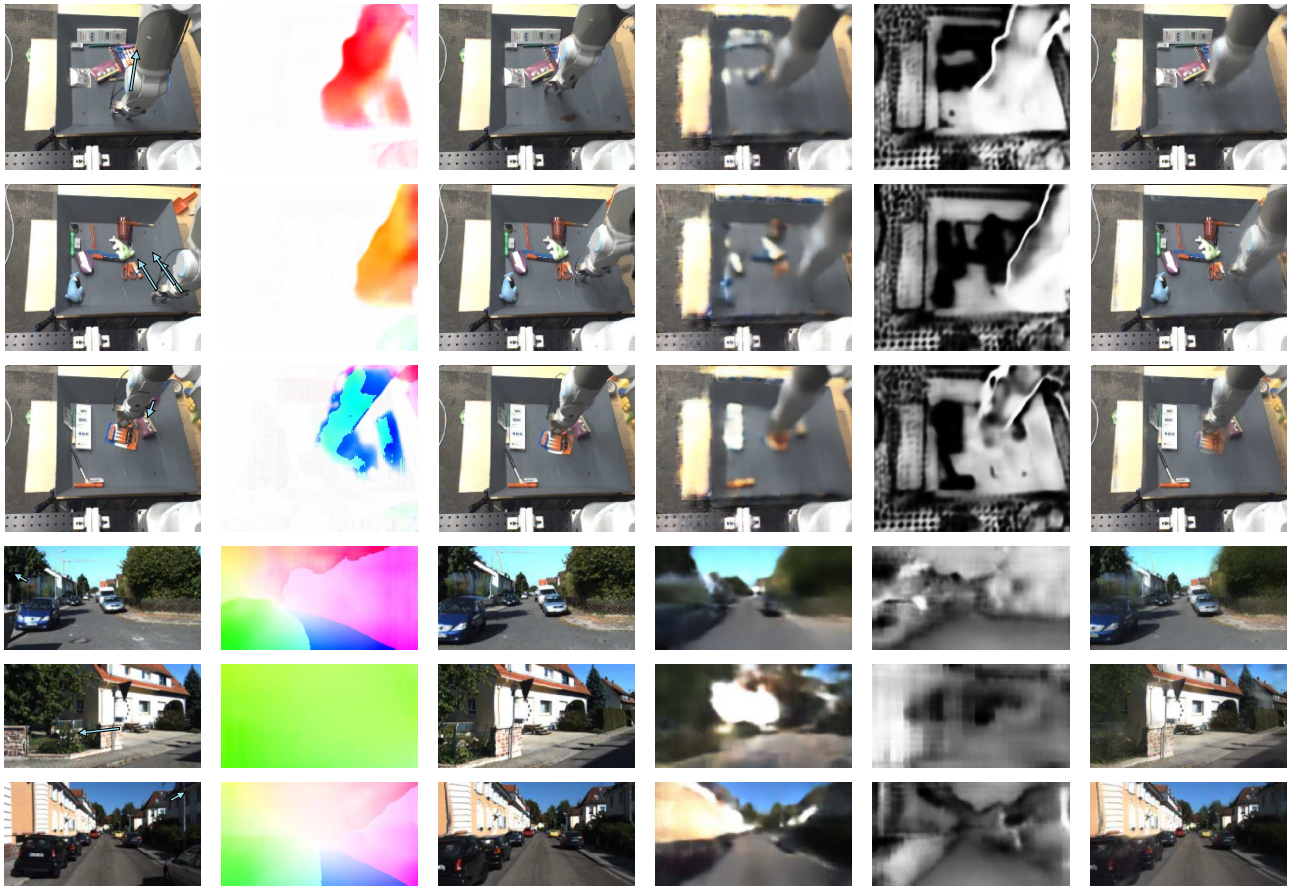



Figure 5. **Example intermediate results**. From left to right: input image and flow vectors, dense flow map predicted by flow completion branch, intermediate output generated by warping the input image with the dense flow map, intermediate output generated by hallucination branch, mask to combine both outputs (inverted for better contrast), final output.

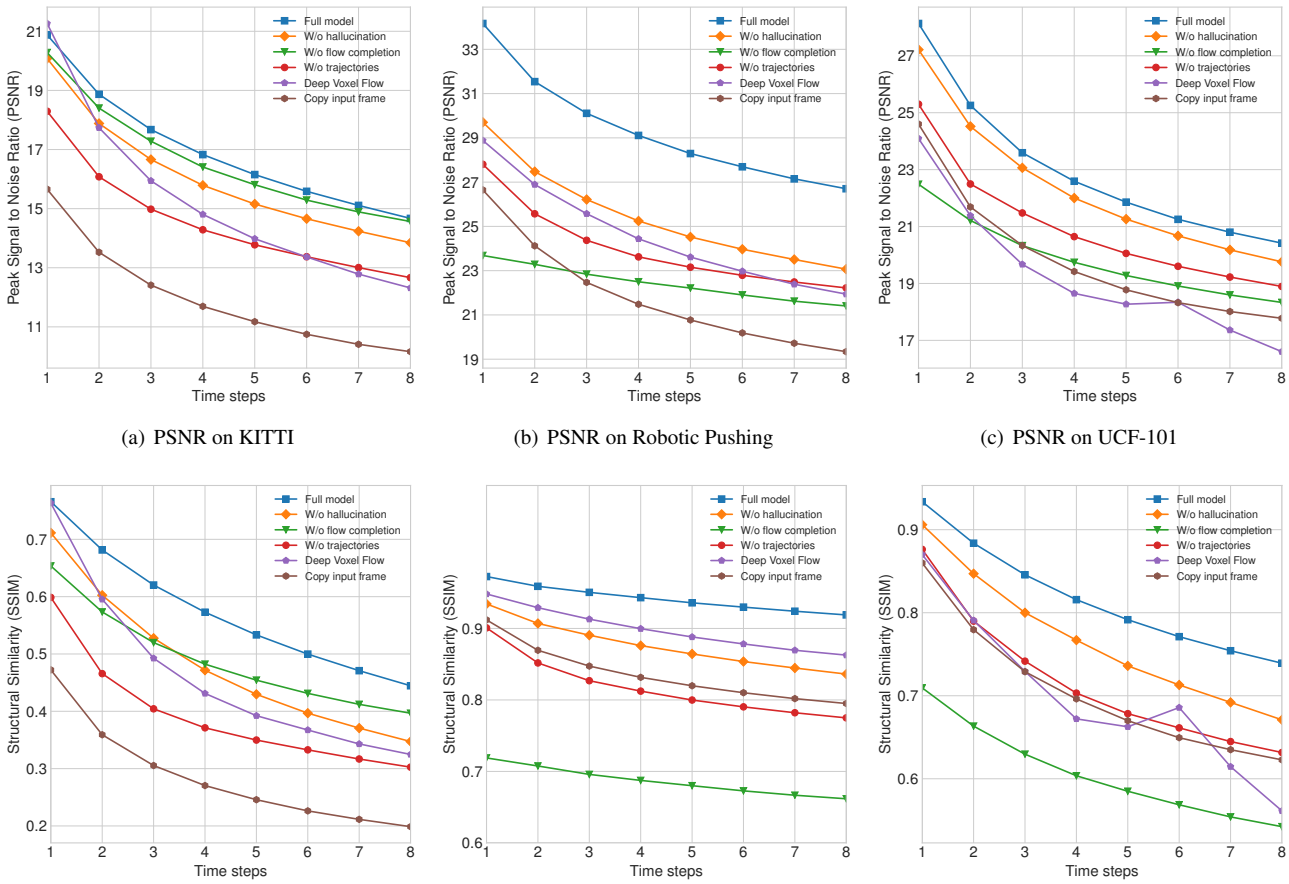
bers in parentheses are raw data). For all three datasets, most users prefer our full model to the baselines. The hallucination-only model generates very blurry videos and is almost never preferred. However, when paired with flow completion, it significantly improves the performance, especially on the challenging UCF-101 dataset. In Figure 5, we show examples of flow vectors drawn by users and corresponding generation results of different variants of models. Our full model produces more realistic results in most cases.

In Figure 6 we show some failure cases of warp-only model. Generating new images by warping pixels from other images can almost always guarantee image sharpness. However, it is extremely sensitive to color and lightness difference between source and destination images that are commonly introduced by camera exposure change and lighting condition change. This is exactly the case for KITTI dataset. We observed that when the model don't have other means to compensate for these changes (e.g. hallucination branch), the model can be trained to produce erroneously warped output images, as shown in Figure 6, which is not observed in full model.

Figure 6. **Failure cases of using warping alone**. We observed that when the model don't have other means to compensate for these changes (e.g. hallucination branch), the model can produce erroneously warped output images.

### 4.4. Quantitative comparison on video prediction

To conduct quantitative comparison with other video estimation algorithms using objective metrics, we feed our model with trajectories automatically extracted from test




(a) PSNR on KITTI (b) PSNR on Robotic Pushing (c) PSNR on UCF-101

(d) SSIM on KITTI (e) SSIM on Robotic Pushing (f) SSIM on UCF-101

Figure 7. **Quantitative comparison**. We compare our models with baselines on three datasets (KITTI [5], Robotic Pushing [4], and UCF-101 [24]) using two commonly used metrics (PSNR and SSIM [37]). Given a single input frame, the models generate up to 8 frames in the future. Our full model obtains the highest score under almost every condition.

videos. Since the ground truth frames are available, we can evaluate our model using similarity metrics such as PSNR and SSIM. Our test setting is the same as the training setting, where we generate 8 future frames. This setting is similar to previous works on video prediction (*e.g.*, [17]), except that we use sparse trajectories and a single frame as input, rather than multiple input frames. While our model is not designed for this task, this setting can more or less illustrate the potential of our model for user-guided video generation. We increased the maximum number of trajectories to 10 to reduce their arbitrarity. We still note that these scores can only provided a rough guidance on the video quality. A video that is very different from the ground truth will receive a low score, even though it might be perfectly realistic and follows the trajectories well.

Figure 7 shows results from different models on KITTI, Robotic Pushing, and UCF-101. Our full two-step model achieves the highest score except for SSIM on Robotic Pushing. We hypothesize that this is because Robotic Pushing is a relatively stationary dataset with little camera motion and background motion so warping existing pixels is usually good enough. We also test a simple baseline that always copies the input frame, as well as a baseline model that has the full architecture but does not receive trajectories as input (denoted as w/o trajectories). The model w/o trajectories performs much worse than our full model that employs trajectory information, which demonstrates that trajectory-conditioning can significantly improve video generation quality. This phenomenon is also evident in the comparison between our model and Deep Voxel Flow [1] [14]. While Deep Voxel Flow is the state-of-the-art for short-term video prediction, its performance drops significantly with increased output video length.

We also compare our method with Beyond MSE [17], a state-of-the-art approach for frame-conditional video generation. It uses a multi-scale adversarial training pipeline

[1]We used our own implementation of Deep Voxel Flow (based on the simplified code released by the author of Deep Voxel Flow) for benchmark.

| Datasets | Beyond MSE PSNR | Beyond MSE SSIM | Ours PSNR | Ours SSIM |
|---|---|---|---|---|
| KITTI | 10.86 | 0.452 | 20.88 | 0.766 |
| Robot Pushing | 23.30 | 0.886 | 34.17 | 0.973 |
| UCF-101 | 18.64 | 0.830 | 28.13 | 0.934 |

Table 2. **Comparison between our model and Beyond MSE (Non-standard setting) [17].** In this experiment we use both methods to generate the next one frame. Our model receives the input frame and sparse flow vectors, while Beyond MSE [17] only uses the single input frame.

similar to LAPGAN [3]. In their paper, they adopt the setting of taking 4 frames as input and predicting the next 1 frame. To make a direct comparison with our model, we run their open-sourced code using a single frame as input to predict the next frame[2]. As shown in Table 2, our method outperforms Beyond MSE significantly over both PSNR and SSIM on all three datasets. The improvement comes from both our two-step architecture and the trajectory information we use.

### 4.5. Slow motion generation

One of the useful and interesting propriety of our model is that the video length it can generate is not constrained by a fixed number of time steps. As long as the input trajectories are smooth, the output image sequences is also temporally smooth. This enables the model to produce as many frames as needed in a fixed time period by interpolating the input trajectories to have more steps. In other words, our model can convert the video temporal interpolation problem to a much easier trajectory interpolation problem that can be handled by a simple linear interpolation. For illustration, we experimentally interpolated a set of trajectories to have eight times more segments. A portion of frames generated from this set of trajectories are shown in Figure 8. Since the movements between time steps are small, the motion is not easily perceivable on a static image sequence.

## 5. Conclusion

In this paper, we propose a new method to generate videos conditioned on a static image and sparse motion trajectories. Our model learns to copy pixels from the input image wherever possible, and to hallucinate pixels that are missing from the input. It does not require extra human annotations during training time, and generates more realistic videos than state-of-the-art unconditional approaches. In the future, we plan to develop a module that predicts the sparse trajectories from the input image, so that our method can also be applied to video prediction without user guidance. We also observe occasional training difficulty since the warping operator can only get local gradients from 4-pixel neighbors. Using larger transformation kernels like [31] may alleviate this problem.

[2]We also tried to train it to predict up to 8 future frames. However we found the quality degrades quickly and the scores are not competitive. So we don't include it in this comparison.

Figure 8. **Slow motion generation**. We show 8 frames from a 240 FPS video generated by our model. Since the movements between time steps are small, the motion is not easily perceivable.

## 6. Acknowledgments

We would like to thank all the volunteers for participating in the user study as the drawing of trajectories is a very meticulous task. This work was supported in part by a Google Focused Research Award.

## References

[1] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017. 1

[2] B. Chen, W. Wang, J. Wang, X. Chen, and W. Li. Video imagination from a single image with transformation generation. *arXiv preprint arXiv:1706.04124*, 2017. 1, 2, 4

[3] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015. 8

[4] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, pages 64–72, 2016. 2, 4, 7

[5] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 4, 7

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015. 4

[8] N. Kalchbrenner, A. v. d. Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016. 1, 2

[9] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4

[10] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[11] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *European Conference on Computer Vision*, pages 201–214. Springer, 2012. 2

[12] T. Lan, T.-C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *European Conference on Computer Vision*, pages 689–704. Springer, 2014. 2

[13] Y. Li, M. R. Min, D. Shen, D. Carlson, and L. Carin. Video generation from text. *arXiv preprint arXiv:1710.00421*, 2017. 2

[14] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. *arXiv preprint arXiv:1702.02463*, 2017. 1, 2, 3, 4, 7

[15] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016. 2

[16] T. Marwah, G. Mittal, and V. N. Balasubramanian. Attentive semantic video generation using captions. *arXiv preprint arXiv:1708.05980*, 2017. 2

[17] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 2, 4, 7, 8

[18] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871, 2015. 2

[19] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016. 2

[20] S. L. Pintea, J. C. van Gemert, and A. W. Smeulders. Déja vu. In *European Conference on Computer Vision*, pages 172–187. Springer, 2014. 2

[21] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014. 2

[22] M. Saito and E. Matsumoto. Temporal generative adversarial nets. *arXiv preprint arXiv:1611.06624*, 2016. 1, 2

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4

[24] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 4, 7

[25] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*, pages 843–852, 2015. 1, 2

[26] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. *arXiv preprint arXiv:1707.04993*, 2017. 1, 2

[27] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016. 2

[28] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee. Learning to generate long-term future via hierarchical prediction. *arXiv preprint arXiv:1704.05831*, 2017. 1, 2

[29] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating the future by watching unlabeled video. *arXiv preprint arXiv:1504.08023*, 2015. 2

[30] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016. 1, 2, 4

[31] C. Vondrick and A. Torralba. Generating the future with adversarial transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 3, 4, 8

[32] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, pages 835–851. Springer, 2016. 1

[33] J. Walker, A. Gupta, and M. Hebert. Patch to the future: Unsupervised visual prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3302–3309, 2014. 2

[34] J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2443–2451, 2015. 2

[35] J. Walker, K. Marino, A. Gupta, and M. Hebert. The pose knows: Video forecasting by generating pose futures. *arXiv preprint arXiv:1705.00053*, 2017. 2

[36] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558, 2013. 4

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 4, 7

[38] T. Xue, J. Wu, K. Bouman, and B. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2016. 1, 2

[39] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European Conference on Computer Vision*, pages 286–301. Springer, 2016. 3, 4

[40] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. *arXiv preprint arXiv:1611.07715*, 2016. 3