Controllable Animation of Fluid Elements in Still Images

Aniruddha Mahapatra
Adobe Research India
anmahapa@adobe.com

Kuldeep Kulkarni
Adobe Research India
kulkulka@adobe.com

Figure 1. Our approach takes in input image along with the user-provided motion hints (red arrows in (a) and the user-provided mask (white in (b)) indicating the regions of fluid elements to be animated and outputs the sequence of frames of the animated videos.

Abstract

We propose a method to interactively control the animation of fluid elements in still images to generate cinemagraphs. Specifically, we focus on the animation of fluid elements like water, smoke, fire, which have the properties of repeating textures and continuous fluid motion. Taking inspiration from prior works, we represent the motion of such fluid elements in the image in the form of a constant 2D optical flow map. To this end, we allow the user to provide any number of arrow directions and their associated speeds along with a mask of the regions the user wants to animate. The user-provided input arrow directions, their corresponding speed values, and the mask are then converted into a dense flow map representing a constant optical flow map (F_D). We observe that F_D, obtained using simple exponential operations can closely approximate the plausible motion of elements in the image. We further refine computed dense optical flow map F_D using a generative-adversarial network (GAN) to obtain a more realistic flow map. We devise a novel UNet based architecture to autoregressively generate future frames using the refined optical flow map by forward-warping the input image features at different resolutions. We conduct extensive experiments on a publicly available dataset and show that our method is superior to the baselines in terms of qualitative and quantitative met-

rics. In addition, we show the qualitative animations of the objects in directions that did not exist in the training set and provide a way to synthesize videos that otherwise would not exist in the real world. Project url: https://controllable-cinemagraphs.github.io/

1. Introduction

It is widely perceived that animations capture human imagination more than still images. The effect of this can be seen in the proliferation of video content that is being uploaded on social media. Studies show that video-based ads and explainers are far more likely to gain trust and engagement than those based on other modalities, leading to a significant boost in sales. However, the required animations or videos are less easily available for the users to leverage than still images that exist in abundance in one's collection. Hence, it is desirable to empower the practitioners with controllable tools to convert the still images to videos of the required kind. This motivates us to consider the problem of animating images with user control to generate output videos that are generally called 'cinemapgraphs' in literature. Similar to [11] we focus on the images that contain fluid elements like water, smoke, fire that have repeating textures and continuous fluid motion.

There has been a rich body of work [1,3,5,7,9–11,16,20,25]

that has focused on generating animations from still images. While [3, 7, 11, 16, 20, 25] focus on uncontrollable image-to-video synthesis, attempts [1, 2, 5, 9, 10] have been made for controllable image-to-video synthesis with the user-provided direction of the motion of the objects in the images. While these methods provide some control to the user, they suffer from certain drawbacks. Specifically, [1, 2, 5] either allow the user to poke at just a single pixel location or provide a single user direction. Halperin et. al [9] obtain a displacement field by exploiting self-similarity that exists in images of repeating structures like buildings, staircases. However, such a method is unsuitable for animating fluid objects that we are considering as such objects do not have specific structures that can manifest in self-similarity, leading to an erroneous displacement field. Hao et. al [10] proposed a method in which the user can provide sparse trajectories as input, defined by the direction of the motion at different locations. A dense optical flow map is estimated in an unsupervised manner and is warped with the input image to obtain the future frames. As shown later, the dense optical flow estimated thus is brittle and is prone to produce unrealistic video synthesis results. Motivated by this, we consider the problem of animating images given i) a single still image ii) a user-provided mask specifying the region to be animated and iii) a set of movement directions, called flow hints at different locations in the masked region.

To circumvent the problems associated with directly obtaining a flow map from user inputs, we propose a two-step approach to estimate the flow map from a sparse set of arrow directions and their associated speeds. Firstly, we approximate the dense optical flow using simple exponential operations on the movement directions and speeds input by the user. Next, the thus estimated approximate flow map is further refined using a GAN-based network [8] to obtain the final estimate of the flow map representing the constant 2D flow map of the desired movement. The estimated flow map along with the input image is fed into a GAN-based image generator, similar to Holynski et al. [11] to obtain the future frames. The contributions of our paper are as follows.

- We propose a two-stage approach to interactively control the animation of fluid elements from a still image.

- We propose a novel approach to approximate the constant flow map governing the motion using simple exponential operations on the user-provided inputs in the form of speed and directions.

- Through qualitative and quantitative experiments, we show that our method beats all previous and other proposed baselines on a publicly available dataset of images of fluid motion.

- We prove the generalizability of our method to any arbitrary set of user directions by showing the qualitative

animations of fluid objects in directions that did not exist in the training set.

2. Related Work

Video synthesis works occur in a myriad of ways. A good number of works have focused on video synthesis in an unsupervised and stochastic manner [3, 7, 16, 20, 25, 26, 29]. There is a body of research that deals with video generation from intermediate representations like semantic label map [18, 27, 28]. Of relevance to this work are the works on single-image-to-video synthesis [1, 2, 4–6, 9–11, 14, 15, 29–31]. Chuang et al. [4] animate pictures by allowing the users to decompose the images into several layers, each one of which being needed to be animated in a different fashion. Xiong et al. [30] propose a two-stage approach to synthesize a video from a single image, wherein in the first stage a sequence of frames is generated using a 3D-GAN and in the second stage, the sequence of frames are further refined using another GAN. Li et al. [14] first predict a sequence of optical flow maps for future frames from the input image and then use them to obtain the future RGB frames. Logacheva et al. [15] propose a radically different approach by modeling the sequence of landscape frames in a video in the StyleGAN [12] latent space while enforcing the temporal consistency. Similar to [14], Holynski et al. [11] first estimates the optical flow for future frames, except that the work assumes a constant 2D flow map across the video. These methods [4, 11, 14, 14, 15, 30] generate a video from a single still image automatically and thus do not allow the user interaction to control the animation.

Different from the above set of works, [1, 2, 5, 6, 9, 10] allow the users to interact and control the movement in the animation to varying degrees, and hence are more closely related to the current work. Dorkenwald et al. [5] propose a one-to-one mapping between image and video using a residual representation, that allows the user to provide a single direction of motion for video generation. [1] and [2] propose methods that govern the animation of different parts in the image with a single poke at a particular location defined by the start and end location of the motion. However, these methods [1, 2, 5] are unsuitable for our problem that necessitates the use of a sparse set of input directions and speeds at arbitrary locations. The closest approach to our work is [10]. This approach allows for user interaction through sparse trajectories for every frame to be predicted. Given the sparse trajectory for a particular frame, a single network is used to obtain a dense optical flow map and a hallucinated image. The dense optical map is bilinearly warped with the input image to obtain an estimate for the frame that is further added to the hallucinated image to obtain the final predicted frame. We differ from this method in two ways, i) instead of obtaining the dense optical flow directly from sparse trajectories, we first obtain its approximation by ap-

plying simple exponential functions on the user inputs, and then refining it using a network. and ii) instead of obtaining the final image by simple bilinear interpolation on the input image, we adapt the method from [11] and use a separate network that takes in the input image and flow map corresponding to a particular frame, with symmetric splatting of intermediate features to obtain the RGB frame.

3. Methodology

Given a single input image $I \in \mathbb{R}^{H \times W \times 3}$, mask of the regions in the image the user wants to animate $M \in \mathbb{R}^{H \times W \times 1}$, K arrow directions $A^{1..,K}$ and corresponding speed values $S^{1..,K}$, our goal is to generate a realistic animated video comprising of N frames ($I^{1..,N}$). Our method consists of first converting the arrow directions and speed values to sparse input flow maps, $F_S \in \mathbb{R}^{H \times W \times 2}$ (Section 3.2), where flow at position i (at the location of arrows in F_S) defines the position where a pixel at that location will move to in all future frames. We propose to convert F_S into a dense flow map F_D (Section 3.3) using simple exponential functions. Using F_D and input image I as guidance, we use a flow-refinement network to generate a dense refined flow map F_D^R (Section 3.4). Finally, to generate output video frames, we use a UNet [21] based frame generator to generate video frames by warping the input image with F_D^R at different resolutions of feature maps (Section 3.5).

3.1. Preliminary: Eulerian Flow Fields

Generating video frames by warping a single image using optical flow requires very accurate optical flow to obtain realistic video. In the case of real-world videos, the optical flow between each pair of frames in a video, in most cases, is time-varying. Following this principle, Endo et al. [6] predicts optical flow autoregressively using the previously generated frame. Although theoretically, this seems feasible, in the long-term it leads to large distortion due to error propagation. In contrast, Holynski et al. [11] hypothesize that a constant and time-invariant optical flow field M_F, termed as Eulerian flow, that describes the motion of pixel locations between consecutive frames in a video, can accurately approximate the complex motion of fluid elements (like water, smoke, fire, etc.) in realistic videos. More specifically, for a given pixel location i, the optical flow $F_{t \to t+1}$ between consecutive frames in a video at any time t is given by,

$$F_{t \to t+1}(i) = M_F(i) \tag{1}$$

Correspondingly, the optical flow between the first frame and frame at any time t, can be obtained by Euler-integration of M_F, t times as given by,

$$F_{0 \to t}(i) = F_{0 \to t-1}(i) + M_F(i + F_{0 \to t-1}(i)) \tag{2}$$

where $F_{0 \to 1} = M_F$.

Since we also operate in the domain of fluid elements, we adopt the principle proposed in [11] to use a constant optical flow field to model the motion of elements in the generated frames.

3.2. Baseline: Sparse Input Flow $F_S \to$ Sparse Refined Flow F_S^R

We convert the arrow directions $A^{1..,K}$ and corresponding speed $S^{1..,K}$ into sparse optical flow map F_S. Each arrow A^n (where $n \in [0, K]$) at location i in the image, is given by the start (i) and end (j) location of the arrow $(x^i, y^i) \to (x^j, y^j)$. The sparse optical flow F_S is formulated as,

$$F_S(x^i, y^i) = \begin{cases} (x^j, y^j) * S^n & \text{if } A^n \text{ starts at } (x^i, y^i) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Following Hao et al. [10], who generate time varying dense flow maps from input sparse trajectories, we transform the sparse flow map F_S to a dense optical flow map F_S^R. However, unlike their method that rely on generation of time-varying flow maps, we generate a constant (Eulerian) dense optical flow F_S^R. For this we use a SPADE [19] based flow-refinement network G_F by using F_S, input image and mask as cues, in the SPADE normalization layers. We call this baseline to compute dense flow map as 'Hao et al. modified'.

3.3. Sparse Input Flow $F_S \to$ Dense Flow F_D

From fig. 3, it can be observed that the refined dense optical flows F_S^R generated using F_S are very different from ground-truth Eulerian flows. We hypothesize that F_S does not provide adequate information to G_F to produce realistic flows. Instead of directly using F_S to generate dense flow using G_F, we propose to create an intermediate dense optical F_D from F_S using simple functions of the distances between every pixel location and the arrow positions. We calculate the exponential $L2$ Euclidean distance (D_{exp}) between each pixel location in input image and the starting coordinate of all the K arrows. The exponential of the Euclidean distance between location (x^i, y^i) of input image and starting position of arrow A^j, (x^j, y^j) is given by,

$$D_{L2}^{i,j} = \|(x^i, y^i) - (x^j, y^j)\|_2 \tag{4}$$

$$D_{exp}^{i,j} = e^{-\left(D_{L2}^{i,j}/\sigma\right)^2} \tag{5}$$

where $D_{L2}^{i,j}$ is the Euclidean distance between location (x^i, y^i) of input image and starting position of arrow A^j, (x^j, y^j) and σ is a constant. The dense optical flow F_D for a particular pixel location i in in the image is defined as weighted average of flow magnitude at each non-zero location in input mask M, where the weights are taken from
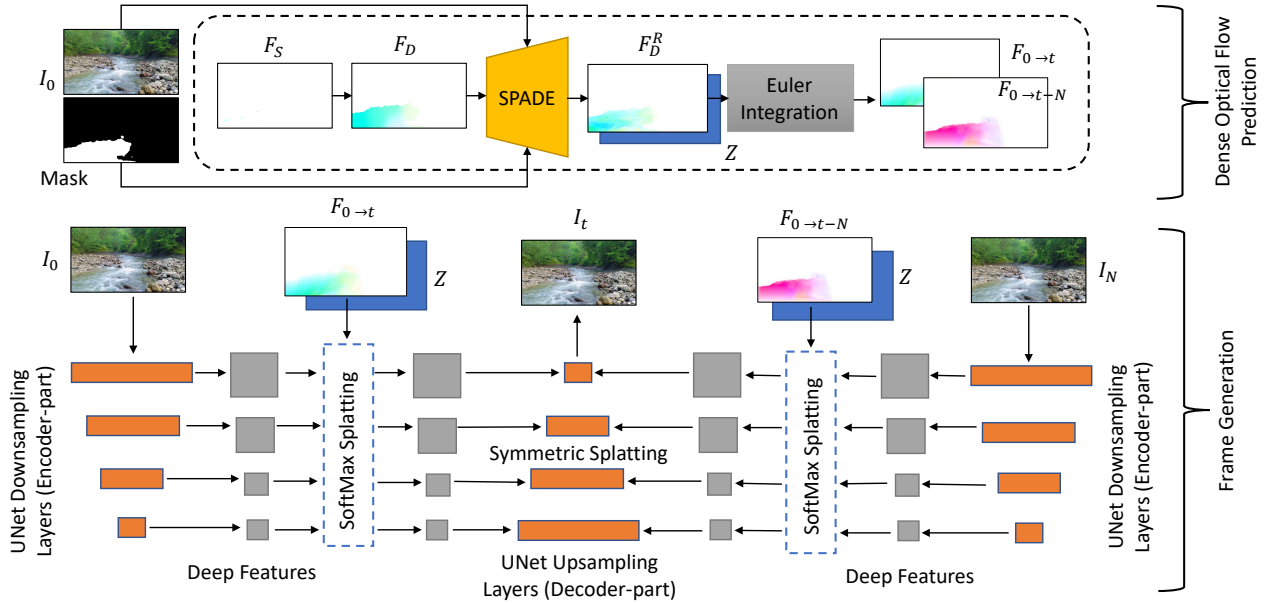
Figure 2. The figure shows our full pipeline. The inputs to our system are the input image, the user-provided mask indicating the region to be animated, and motion hints, F_S. The motion hint is converted into a dense flow map F_D using simple exponential operations on F_S, which is further refined using a SPADE network, G_F to obtain F_D^R. During test time, instead of using both I_0 and I_N, we obtain the tth frame, I_t as the output of the UNet into which we feed the input image in place of both I_0 and I_N and the Euler integrated flow maps corresponding to that frame, in both forward and backward directions that are used to perform symmetric splatting in deep feature space.

$D_{exp}^{i,j}$ and is given by,

$$F_D(i) = \begin{cases} \sum_{j=1}^{K} \frac{D_{exp}^{i,j} * F_S(i)}{\sum_{j=1}^{N} D_{exp}^{i,j}} & \text{if } i \in M \\ 0 & \text{if } i \notin M \end{cases} \quad (6)$$

3.4. Dense Flow $F_D \rightarrow$ Dense Refined Flow F_D^R

Although F_D can suitably describe the motion of fluid regions moving approximately in the same direction, and also having smooth transition at the boundary of two different flow regions, there are limitations of directly using F_D to generate video frames by warping. In the figures 3 and 5, we consider examples and show the approximated dense flow and the refined dense flow. While the dense flow is a good approximation, the refined dense flow is closer to the ground-truth flow than the approximated flow. From figure 5 we see that the dense flow map generated is assigned the same horizontal flow to the majority of the waterfall and below the lake, whereas, realistically the waterfall would have been moving vertically downward. This is due to the fact the F_D is generated purely based on the closeness of a particular pixel location to hint points with simple exponential operations. Hence, unlike F_D^R, it cannot distinguish the object boundaries (in this case the dotted demarcation like between waterfall and lake in figure 5) and naively mixes

the flow values in both regions. Due to this same drawback, it assigns very different flow values to different regions of the same waterfall, which again is realistically inappropriate. Hence, using our flow-refinement network G_F, with the input image, mask, and F_D as cues, we generate a dense refined optical flow F_D^R. Using F_D^R as Eulerian flow fields, we generate video frames by the method discussed in Section 3.5.

3.5. Video Frame Generation

From the refined dense optical flow field F_D^R, we estimate the flow fields $F_{0 \rightarrow t}$ from input image to all the future frames for $t \in [0, N]$ using equation 2. Instead of the backward optical flow field (as used in [6]), we use forward flow to warp input frame to generate future frames as [11] observed that forward flow produces more reliable flow estimates and sharper values at object boundaries. However, forward warping (also known as splatting) has its challenges *(i)* it can map multiple source pixels to the same destination resulting in loss of information and aliasing, *(ii)* it may not map any source pixel to a particular target location leading to blank region. To mitigate these artifacts, we use softmax-splatting (proposed in [17]). It resolves the first challenge associated with splatting by using softmax to scale the con-

Input Image   Ground-Truth Flow

Sparse Input Flow   Refined Sparse Flow

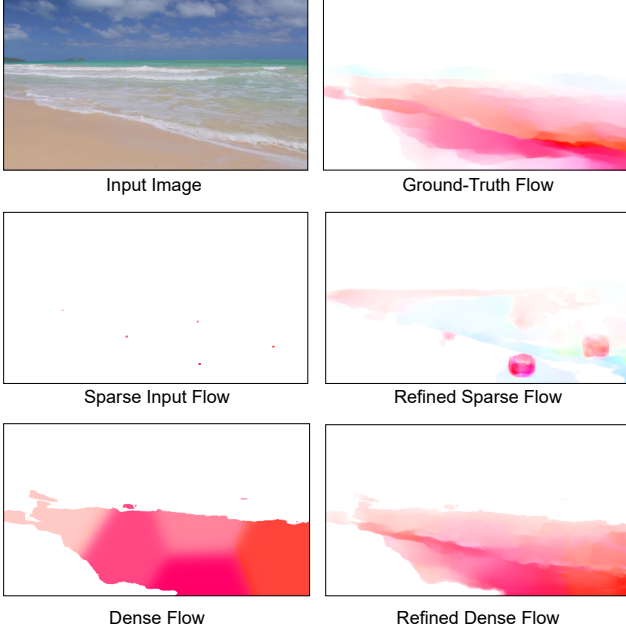Dense Flow   Refined Dense Flow

Figure 3. The figure shows the input image, the sparse input flow in the form of flow hints and the various flows that can be obtained. It is clear that that dense flow computed using simple functions closely approximates the ground-truth flow, and the refined dense flow best resembles the ground-truth flow among all. The refined sparse flow obtained is quite poor.

tribution of source pixels mapping to the same destination location, based on importance metric $Z \in \mathbb{R}^{H \times W \times 1}$. In our method, we predict Z as an additional channel in the output of G_F during fine-tuning (Section 3.5.3).

3.5.1   Multi-scale Deep Feature Warping

Instead of directly splatting the input image with $F_{0 \rightarrow t}$ in RGB space to generate frames, that might otherwise produce holes in generated frames, we perform splatting on the deep features of the image, similar to [11]. However, unlike their method, we perform splatting at different resolutions of input image features. Using a UNet based image generator G_I, we extract the image features at different resolutions from the UNet encoder. We use softmax-splatting to warp the features at different scales and generate an image using the decoder of the UNet. Note that all the splatted features from the encoder part, except at the bottleneck layer of UNet, are connected to the decoder via skip connections. For a feature map corresponding to input image I at resolution D_0^r, the softmax-splatting output D_t^r at pixel location i using $F_{0 \rightarrow t}$ is given by,

$$D_t^r(i) = \frac{\sum_{j \in \mathcal{X}} D_0^r(i) e^{Z(j)}}{\sum_{j \in \mathcal{X}} e^{Z(j)}} \qquad (7)$$

where $\mathcal{X}$ consists of all the pixels that map to the same target location i after splatting.

3.5.2   Symmetric Splatting

Even with softmax-splatting on multi-scale image feature space, we observe the presence increasingly large void regions, in the generated frames (similar to what was observed in [11]) in places of significant motion where pixels are warped away from the region and there are no pixels to replenish them. We hypothesize that G_I might not be able to generate appropriate pixel values for these regions to fill in the gaps. To resolve this artifact we use the method of symmetric-splatting proposed in [11]. In this method, similar to producing flow fields in the forward direction $F_{0 \rightarrow t}$, we also generate flow fields in the backward direction $F_{0 \rightarrow -t}$ by Euler-integration of $-M_F$. Thus, instead of just using softmax-splatting on deep features D_0^r obtained from first frame I_0 with $F_{0 \rightarrow t}$ to generate D_t^r, we use a combination of the deep features D_0^r from I_0 and D_N^r obtained from last frame I_N, splatted with $F_{0 \rightarrow t}$ and $F_{0 \rightarrow t-N}$ respectively. Specifically, any given pixel location i in the combined deep feature $\hat{D}_t^r$ is given by,

$$\hat{D}_t^r(i) = \frac{\sum_{j \in \mathcal{X}} \alpha D_t^r(i) e^{Z(j)} + \sum_{j \in \mathcal{X}} \hat{\alpha} D_{t-N}^r(i) e^{Z(j)}}{\sum_{j \in \hat{\mathcal{X}}} \alpha e^{Z(j)} + \sum_{j \in \hat{\mathcal{X}}} \hat{\alpha} e^{Z(j)}} \qquad (8)$$

where α and $\hat{\alpha}$ equals $\frac{t}{N}$ and $(1 - \frac{t}{N})$, $D_t^r(i)$ and $D_{t-N}^r(i)$ are feature map obtained by softmax-splatting D_0^r, D_N^r with $F_{0 \rightarrow t}$ and $F_{0 \rightarrow t-N}$ respectively, $\mathcal{X}$ and $\hat{\mathcal{X}}$ consists of all the pixels that map to the same target location i after splatting for D_t^r and D_{t-N}^r respectively. The intuition behind this is the void regions appearing in the frames generated at time t by splatting the first image with $F_{0 \rightarrow t}$ is complementary to the void regions appearing in the frames generated by splatting the last image with $F_{0 \rightarrow t-N}$.

3.5.3   Training and Inference

As proposed by [17], for stable training, we first train the 2 components of dense optical flow refinement and frame generator separately. While training G_F, we use the standard GAN loss and Discriminator Feature-matching loss [19]. In this stage, we compute losses based on generated dense refined optical flow (and not the generated Z). During the training of G_I, we use the standard GAN loss, VGG loss [22], L1 loss, and Discriminator Feature-matching loss. Prior to end-to-end fine-tuning, we freeze the refined dense optical flow maps and only train G_F to generate Z. In addition, we only use the discriminator for the frame generator part. Contrary to training where we use both the first and the last frames for symmetric-splatting used in G_I, at test time, since we only have a single input static image, use the same image as both the first and the last frames in G_I.

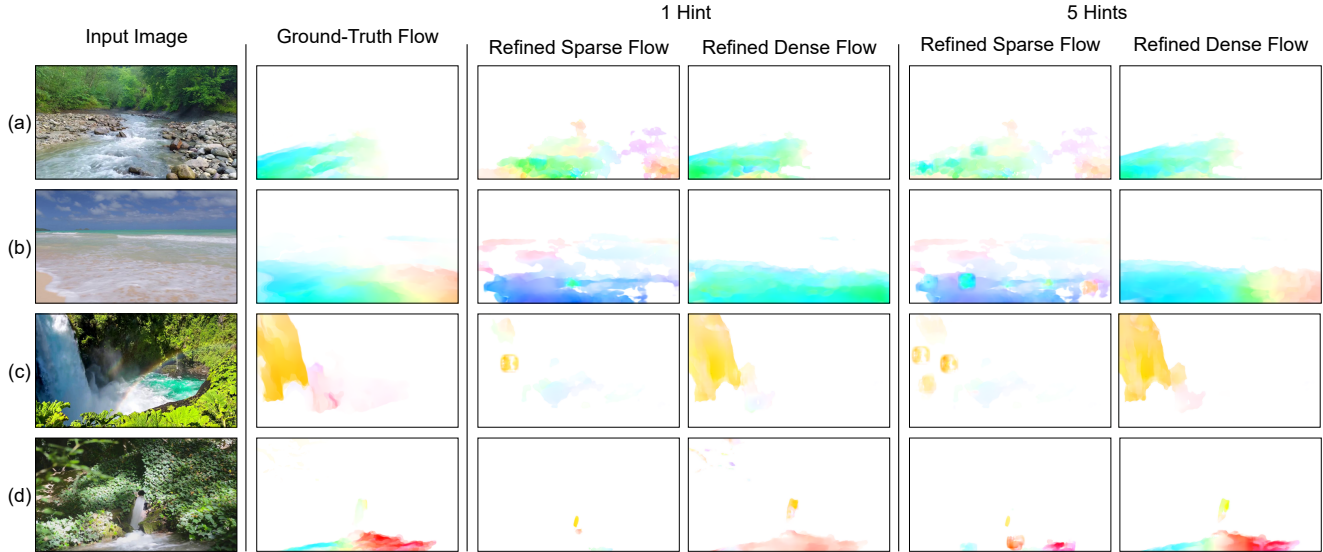|	Input Image	Ground-Truth Flow	1 Hint		5 Hints	
			Refined Sparse Flow	Refined Dense Flow	Refined Sparse Flow	Refined Dense Flow

Figure 4. The figure shows the comparison of the refined sparse flow and the refined dense flow for one hint and five hints. It is quite evident that across diverse set of hints, the refined dense flow resembles the ground-truth flow far more closely that refined sparse flow.

4. Experiments

4.1. Dataset preparation

Due to the unavailability of any existing human-annotated controllable video generation datasets with masks, input flow hints/arrows in the domain of natural scenes, we curate our dataset from the uncontrollable video generation dataset provided by [11]. This dataset already contains the ground-truth videos, starting frame, and the average optical flow for all videos. The number of frames across all videos is 60, each frame having a resolution of 720x1280. For our purpose of training and testing, in addition to the input image, and average flow, we also require a mask of regions the user wants to annotate along with arrows and corresponding speed values. We generate a substitute of the user annotated mask and arrows with the mask and flow hint points generated heuristically from ground-truth average optical flow.

Mask Generation: For every average optical flow map F_{avg} in the dataset, we calculate the mean-squared flow value for F_{avg}. Then we mask out all the regions having per-pixel squared-flow less than m times the mean-squared flow value. Following observation on the visual quality of generated mask at different m values, we set the value of m to 10.

Flow Hint Generation: Using the mask M generated in the previous step, we calculate masked average flow F_{avg}^M for each video as $M * F_{avg}$, where F_{avg} is the ground-truth average flow map for that video. We perform k-means clustering for 100 iterations on the individual F_{avg}^M, to find the cluster centers based on the number of desired hint points

for our input sparse optical flow map F_S. F_S consists of values equal to the ground-truth average flow maps at the pixel location of cluster centers and zero elsewhere. In our experiments, the number of hints points are chosen to be either 1, 3, or 5. In Section 3.2, we discuss the procedure to convert user-provided arrows and speed values to F_S, which is required in real-world interactive testing.

4.2. Experimental Setup

For flow refinement network G_F, we use SPADE [19]. We also use multi-scale discriminator D_F from [19] during training. We train the flow refinement part of our method separately for 200 epochs with both generator and discriminator learning rates of 2×10^{-3} with TTUR method of updating learning rate proposed in [19]. We train on a triplet of (first frame, F_S, ground-truth average flow), where F_S is randomly selected to have 1, 3, or 5 flow hints. For frame generation network G_I, we use a modification of UNet (shown in fig. 2) which incorporates symmetric-splatting. We use the same multi-scale discriminator D_I from [19] during training. We train the frame generation part separately for 200 epochs on training tuples of (start frame, ground-truth average flow, middle frame, last frame), where the middle frame is selected randomly from time $[1, 59]$. Both generator and discriminator learning rates are set to 2×10^{-3} with the TTUR method of updating the learning rate. During fine-tuning we only use G_F, G_I and D_I. Additionally, we fix the value to G_F that is responsible for flow generation and only keep Z trainable. Both generator and discriminator learning rates are reduced to 1×10^{-3}. We train for 40 epochs. Prior to training, we resize all the average flows and frames to 288x512 (maintaining the $\frac{16}{9}$
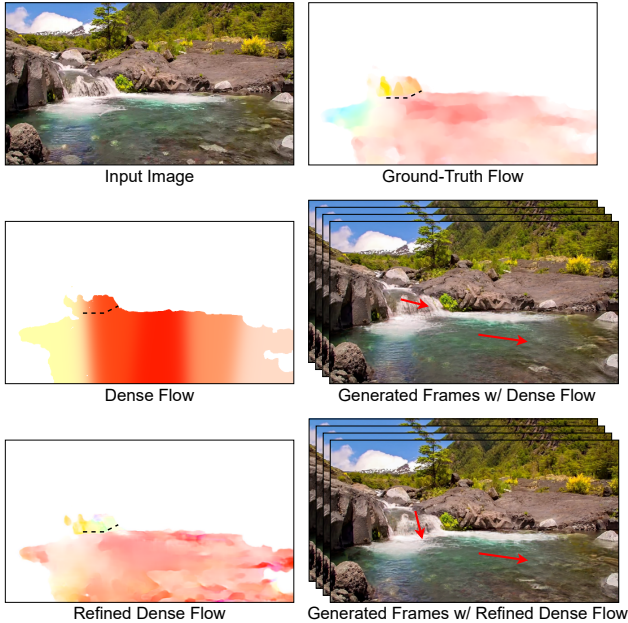
Figure 5. The figure shows the dense flow and the refined dense flow for a particular example and the corresponding generated frames of the video. While the dense flow is a good approximation, the direction of motion is more appropriate in the video corresponding to the refined dense flow that closely resembles the ground-truth flow. In the dense flow video, the waterfalls tend to shift sideways to the right, while the refined dense flow video is very similar to the ground-truth video. The videos are shown in the supplemental. The dotted line (- - -) denotes the region where the waterfall ends and the lake starts.

aspect ratio of original frames). At inference, we generate 60 frames.

4.3. Baselines

We compare our final method with five different baselines, one of our own, Endo et al. [6], Hao et al. [10], and the vanilla Eulerian method [11]. Our own baseline is computing the dense flow map using simple exponential functions followed by an image generator. Endo et al. provide an optimization procedure to compute the directions and speed during test time. For Hao et al., we repurpose the method they have provided in their paper by making the following modifications. Instead of obtaining sparse trajectories, we use the sparse hints that were obtained using the procedure outlined earlier in the section. Instead of having a single network to convert the sparse user inputs, we first compute dense flow map directly from the sparse hints by training the same procedure, i.e the GAN-based network. Once the dense flow map is obtained, instead of bilinearly warping the input image (as done in Hao et al.), we use the same image generator that is used for our method to generate the

	Method	FVD ↓	PSNR ↑
1 Hint	Endo et al.	561.33	23.59
	Hao et al. modified + frame generator	419.015	25.12
	Our (F_D + frame generator)	419.49	**25.2**
	Our (F_D^R + frame generator)	**380.475**	25.07
3 Hints	Endo et al.	526.55	23.35
	Hao et al. modified + frame generator	375.98	25.11
	Our (F_D + frame generator)	331.8	**25.22**
	Our (F_D^R + frame generator)	**318.39**	25.09
5 Hints	Endo et al.	519.18	23.21
	Hao et al. modified + frame generator	344.55	25.11
	Our (F_D + frame generator)	335.4	**25.24**
	Our (F_D^R + frame generator)	**315.31**	25.1
	Vanilla Eulerian	419.74	25.2

Table 1. The table shows the FVD and PSNR values for various methods that use different number of hints as well as the Vanilla Eulerian method for the original speed. It clearly shows our method that uses refined flow and frame generator performs the best in terms of FVD, thus showing the high-fidelity animations generated by our method.

frames from the computed dense optical flow and the input image. We dub this baseline as 'Hao et al. modified + frame generator'. In addition, we also compare our results with the vanilla Eulerian method which is fully automatic and does not require any user inputs.

4.4. Metrics

In order to evaluate our method against the various baselines, we use the following metrics.
Frechet Video Distance (FVD) [24]: It is a standard metric used to quantify the fidelity of the generated videos and provides a measure of the distance between the generated videos and the real videos. Prior to obtaining the features, we resize all videos to 224×224 and use 60 frames. To obtain the features from the videos, we use the pre-trained I3D [23] model that was trained on Kinematics dataset [13].
PSNR: While FVD assesses the perceptual quality of the generated videos, we assess the mean pixel accuracy using PSNR. Given that it is based on mean square error, PSNR tends to favor those methods that produce somewhat blurry results.

4.5. Results

Evaluating the flow map generation: In order to show the efficacy of our flow map generation method, we compute the mean PSNR between the generated flow maps and the ground-truth flow maps. The results for three different numbers of hints are shown in table 2. It is clear that our method that involves flow refinement network on dense flow is able to generate better flow maps than just sparse hints with refinement network, in terms of PSNR. This trend is consistent across a different number of hints. It is interesting that for the vanilla Eulerian method, the PSNR is lower than both methods. This is expected given that the Eulerian
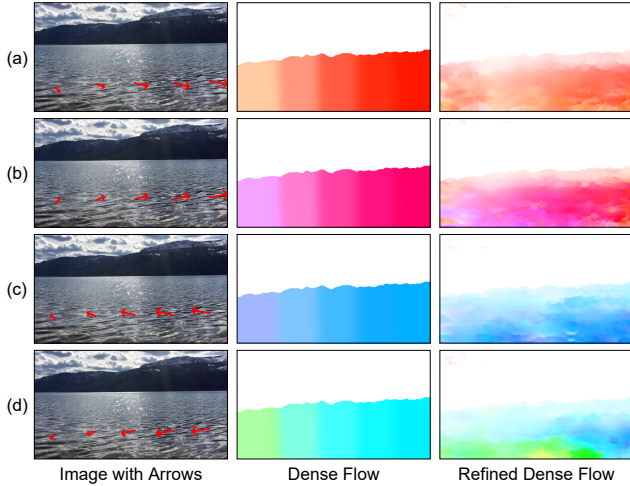
Figure 6. The figure shows dense flows and corresponding refined dense flow maps for four different arrow directions for the same input image. This shows the robustness of our flow generation method to arbitrary input directions for flow hints.

	Method	PSNR $\uparrow$
1 Hint	Hao et al. modified	20.48
	Our (F_D^R)	**24.15**
3 Hint	Hao et al. modified	21.06
	Our (F_D^R)	**25.53**
5 Hint	Hao et al. modified	21.4
	Our (F_D^R)	**25.82**
	Vanilla Eulerian	18.28

Table 2. The table shows the average PSNR between the ground-truth flows and the two refined flows. The refined dense flow is significantly better than the refined sparse flow, thus corroborating the need for the exponential operations before refinement.

method is automatic, unlike the other two methods. Similarly, we show in figure 4, for various examples and the different number of hints, that the flow maps computed by our method perform significantly better than 'Hao et al. modified' and more closely resemble the ground-truth flows.

Evaluating the video generation: Table 1 shows the comparison of the various methods for two different metrics for three different numbers of hints. It is clear that our methods beat all the baselines in terms of FVD, while the PSNR is comparable across the different methods except Endo et al is the least of all. We also observe that our method (Our (F_D + frame generator)) is better than 'Hao et al. modified', thus showing the need to approximate the dense flow map. In addition, our final method (Our (F_D^R + frame generator)) is better than just (Our (F_D + frame generator), thus buttressing the need for refining the approximated dense flow. It is also observed that the FVD scores for all methods get progressively better with an increasing number of hints.

Qualitative Results: From figures 3 and 4, it is clearly visible that dense refined flow F_D^R resembles the average ground-truth optical flow much more accurately than the sparse refined flow F_S^R. The generated videos using our methods and baseline are provided in supplementary. In the figure 5, we consider a particular example that shows the approximated dense flow, the refined dense flow, and a frame of the generated video (video provided in supplementary). While the dense flow is a good approximation in many situations, the refined dense flow corrects some of the regions representing inappropriate flows in dense flow map (see figure 5). This is directly reflected in the quality of the anima-

tion that is generated. The video generated using the dense flow has artifacts wherein the waterfalls tend to shift progressively towards the right (due to dense flow having the same flow values for waterfall and lake), whereas the video generated using the refined flow is very realistic and resembles the actual downward motion of waterfall observed in real-world videos.

Animation in arbitrary directions: Our method is capable of generating flow maps from flow hints that correspond to any arbitrary directions that may not have existed in the training set. Figure 6 shows the refined dense flows generated using the same input image with different arrow directions, and thus different dense flows. We see that G_F produces results that respect the input arrow directions and are not just based on the input image, showing the robustness and the generalizability of our method. Please see the supplement material for corresponding generated videos.

5. Conclusions and Limitations

We propose a method to animate images that contain fluid elements like water, fire, smoke, given a user-provided mask and flow hints in the forms of speed and direction. We proposed a simple yet powerful method to approximate the constant flow field governing the motion with simple exponential operations on the user-provided flow hints, and further show that in order to obtain a better flow field we need to refine using a network the approximation for dense flow field rather than just the sparse hints. Through quantitative experiments, we show that our method performs better than all baselines for a various number of motion hints. One of the limitations of our method is that it is restricted to the movements of fluid elements in an image. The motion of rigid objects or even definite structures like designs in buildings cannot be modeled using a constant flow field, thus making our method not applicable in such scenarios. Another potential limitation of our method is its inability to model multiple flow streams that are adjacent to each other but may belong to different objects.

References

[1] Andreas Blattmann, Timo Milbich, Michael Dorkenwald, and Bjorn 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. 1, 2

[2] Andreas Blattmann, Timo Milbich, Michael Dorkenwald, and Bjorn Ommer. Understanding object dynamics for interactive image-to-video synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5171–5181, 2021. 2

[3] Lluis Castrejon, Nicolas Ballas, and Aaron Courville. Improved conditional vrnns for video prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7608–7617, 2019. 1, 2

[4] Yung-Yu Chuang, Dan B Goldman, Ke Colin Zheng, Brian Curless, David H Salesin, and Richard Szeliski. Animating pictures with stochastic motion textures. In *ACM SIGGRAPH 2005 Papers*, pages 853–860. 2005. 2

[5] Michael Dorkenwald, Timo Milbich, Andreas Blattmann, Robin Rombach, Konstantinos G Derpanis, and Bjorn Ommer. Stochastic image-to-video synthesis using cinns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3742–3753, 2021. 1, 2

[6] Yuki Endo, Yoshihiro Kanamori, and Shigeru Kuriyama. Animating landscape: Self-supervised learning of decoupled motion and appearance for single-image video synthesis. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019)*, 38(6):175:1–175:19, 2019. 2, 3, 4, 7

[7] 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. 1, 2

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2

[9] Tavi Halperin, Hanit Hakim, Orestis Vantzos, Gershon Hochman, Netai Benaim, Lior Sassy, Michael Kupchik, Ofir Bibi, and Ohad Fried. Endless loops: detecting and animating periodic patterns in still images. *ACM Transactions on Graphics (TOG)*, 40(4):1–12, 2021. 1, 2

[10] Zekun Hao, Xun Huang, and Serge Belongie. Controllable video generation with sparse trajectories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7854–7863, 2018. 1, 2, 3, 7

[11] Aleksander Holynski, Brian L. Curless, Steven M. Seitz, and Richard Szeliski. Animating pictures with eulerian motion fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5810–5819, June 2021. 1, 2, 3, 4, 5, 6, 7

[12] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. 2

[13] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 7

[14] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Flow-grounded spatial-temporal video prediction from still images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 600–615, 2018. 2

[15] Elizaveta Logacheva, Roman Suvorov, Oleg Khomenko, Anton Mashikhin, and Victor Lempitsky. Deeplandscape: Adversarial modeling of landscape videos. In *European Conference on Computer Vision*, pages 256–272. Springer, 2020. 2

[16] Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos. *arXiv preprint arXiv:1906.07889*, 2019. 1, 2

[17] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 4, 5

[18] Junting Pan, Chengyu Wang, Xu Jia, Jing Shao, Lu Sheng, Junjie Yan, and Xiaogang Wang. Video generation from single semantic label map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2019. 2

[19] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019. 3, 5, 6

[20] Fitsum A Reda, Guilin Liu, Kevin J Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao, and Bryan Catanzaro. Sdc-net: Video prediction using spatially-displaced convolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 718–733, 2018. 1, 2

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[23] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 7

[24] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018. 7

[25] Ruben Villegas, Arkanath Pathak, Harini Kannan, Dumitru Erhan, Quoc V Le, and Honglak Lee. High fidelity video prediction with large stochastic recurrent neural networks.

Advances in Neural Information Processing Systems, 32:81–91, 2019. 1, 2

[26] 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. 2

[27] 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. 2

[28] 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. 2

[29] Yue Wu, Rongrong Gao, Jaesik Park, and Qifeng Chen. Future video synthesis with object motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5539–5548, 2020. 2

[30] Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, and Jiebo Luo. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2364–2373, 2018. 2

[31] Jiangning Zhang, Chao Xu, Liang Liu, Mengmeng Wang, Xia Wu, Yong Liu, and Yunliang Jiang. Dtvnet: Dynamic time-lapse video generation via single still image. In *European Conference on Computer Vision*, pages 300–315. Springer, 2020. 2