Motion Prompting: Controlling Video Generation with Motion Trajectories

Supplementary Material

A. Implementation Details

A.1. Architecture and Training

We train our model for 70,000 steps using Adafactor [60] with a learning rate of 1×10^{-4}. We do not use any learning rate decay. For the ControlNet, we copy Lumiere's encoder stack, add in zero convolutions as in [87], and replace the first convolutional layer with a new layer that accepts a $T \times H \times W \times C$ conditioning signal. From the constraints of the base architecture, we have $T = 80$ and $H = W = 128$. We set $C = 64$. During training, we sample the number of input tracks uniformly from 1000 to 2000 inclusive. For each track we randomly assign a sinusoidal positional encoding [70], of 64 dimensions, by sampling integers without replacement from 0 to 16384 – the maximum number of tracks for a 128×128 image, and using the corresponding positional embedding for that integer. Note that the encoding is completely randomly assigned. In particular, its spatial location has no bearing on its embedding.

All sampled videos are passed through Lumiere's spatial super resolution (SSR) model, resulting in a 1024×1024, 80 frame video at 16 frame per second, for a total of 5 seconds. We use Lumiere's SSR model as is, without finetuning it for motion conditioning, as we find that the 128×128 output of the base model already contains all of the motion conditioned dynamics.

Data. We train on an internal dataset of 2.2 million videos. We precompute trajectories on this dataset by center cropping each video to a square, resizing it to 256×256, and then running BootsTAP [13] with a dense gird of query points, resulting in 16,384 tracks per video. During training, a video is sampled, and then tracks are randomly sampled from this dataset. During the Lumiere fine-tuning, videos are resized to match Lumiere's 128×128 input and output size.

A.2. Qualitative Results

We provide additional details for qualitative results in Tab. A1, including text prompt and licensing information. All images and videos are used with permission and under open and free licenses. In addition, as can be seen we construct text prompts to describe the scene but not the motion, in order to limit the influence of text conditioning on the motion as much as possible.

A.3. Mouse GUI

We record mouse motions through a simple HTML GUI, which is shown in Fig. A1. It consists of a `canvas` element which displays the first frame conditioning, labels that indicate the position of the mouse in the canvas, and whether or not it is currently being clicked, a button to start the recording, a countdown timer which gives three seconds before recording starts, and a second countdown timer which shows when the recording will end. We record 80 frames of mouse input to match the five seconds of video that our model outputs at 16 frames per second. For each frame we record the mouse (x, y) position, and a flag indicating whether the mouse is being clicked.

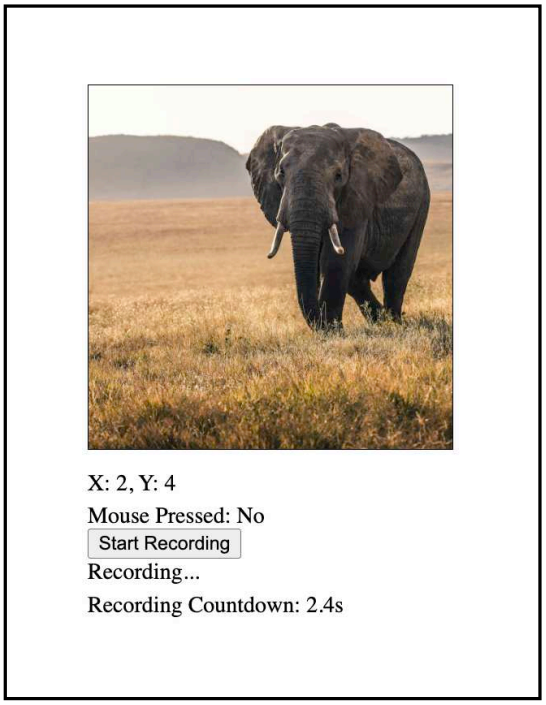

Figure A1. **Mouse Motion GUI.** We show a screenshot of the GUI that we use to record mouse motions. For more information please see Appendix A.3.

A.4. Interacting with and Drag Editing Images

In order to feed mouse motions to our model, we create a grid of tracks that is centered on the mouse whenever it is being dragged. The user may choose the stride of these tracks, and the size of the grid. We use a square grid of tracks for simplicity. In addition, a user may choose to have the tracks "persist," in that before and after the mouse drag the tracks remain. This is useful in cases where objects should stay in place after a drag. A user may also place down a grid of tracks to "pin" the background in place. Note

Table A1. **Figure Details.** We provide details about qualitative samples shown in our figures, including text prompts fed to the model and licensing information. In general, these are sorted by the order that they appear in the paper, moving from left to right, top to bottom.

Description	Figure	Text Prompt	Source	URL	License	License URL
two elephants	Fig. 1, Fig. 5	`elephants`	Unsplash	link	Unsplash	license
owl	Fig. 1	`a close up of a great horned owl`	Unsplash	link	Unsplash	license
brown bear	Fig. 1	`a brown bear`	Unsplash	link	Unsplash	license
squirrel	Fig. 1	`a squirrel sitting on the ground in the woods`	Unsplash	link	Unsplash	license
golden retriever	Fig. 1, Fig. 7	`a golden retriever laying in the grass`	Unsplash	link	Unsplash	license
man (motion source)	Fig. 1, Fig. 8	–	private correspondence	–	permission granted	–
macaque	Fig. 1, Fig. 8	`a macaque monkey`	Unsplash	link	Unsplash	license
sand	Fig. 1, Fig. 3	`sand`	Unsplash	link	Unsplash	license
woman	Fig. 1, Fig. 3	`a woman`	private correspondence	–	permission granted	–
parrot	Fig. 3	`a close up of a green parrot`	Unsplash	link	Unsplash	license
cow	Fig. 3, Fig. 4, Fig. 9	`a highland cow standing in a grassy scottish wilderness`	Unsplash	link	Unsplash	license
skull	Fig. 4	`a white skull on a black background`	Unsplash	link	Unsplash	license
stool	Fig. 4	`a living room`	Unsplash	link	Unsplash	license
hot air balloons	Fig. 4	`a serene scene of multiple hot air balloons floating over Cappadocia, Turkey, during sunset`	Unsplash	link	Unsplash	license
arches	Fig. 5	`arches in arches national park, with shrubbery in the foreground and a bright blue sky`	Unsplash	link	Unsplash	license
roses	Fig. 5	`a red rose`	Unsplash	link	Unsplash	license
cat	Fig. 6	`a cat`	Unsplash	link	Unsplash	license
frog	Fig. 6	`a close up of a frog`	Unsplash	link	Unsplash	license
horse	Fig. 7	`a horse`	Unsplash	link	Unsplash	license
Earth (motion source)	Fig. 8	–	Pexels	link	Pexels	license
panda	Fig. 8	`a panda`	Unsplash	link	Unsplash	license
monkey (motion source)	Fig. 8	–	Pexels	link	Pexels	license
trees	Fig. 8	`birds eye view of trees`	Unsplash	link	Unsplash	license
chess	Fig. 9	`close-up of a chessboard with strong depth of field. The white king piece is in focus, surrounded by black pawns`	Unsplash	link	Unsplash	license

that this setup is identical to how we obtain the "drag-based image editing" results.

A.5. Geometric Primitives

To make spherical tracks we take points on a sphere and follow them as the sphere is spun. This gives us a trajectory of 3D points, which when orthographically projected gives us 2D tracks. The density of the points, the radius of the sphere, and the location of the sphere are determined by the user. Mouse motions are converted to sphere spins by rotating the sphere through a single axis such that the initial mouse location matches with the current mouse location at each frame. This uniquely defines a rotation and ensures that the sphere tracks the mouse.

A.6. Camera Control

In order to obtain camera control, we run a monocular depth estimator on the first frame input to the model. This gives us camera intrinsics as well as depths, allowing us to unproject into a point cloud. We then project this point cloud onto a sequence of camera poses forming the desired camera trajectory, resulting in 2D point tracks. In addition, we run z-buffering to determine occlusions, where only the closest point that has been projected to some neighborhood is visible while all other points in that neighborhood are occluded—unless that point is sufficiently close to the visible point. This requires choosing a radius for the neighborhood size, and a threshold for a point to remain visible if it is close enough to the visible point. Both are set manually to constant values that we find to work well for all examples.

We also discuss translating mouse motion to camera motion. This is done by having the camera move in such a way that the mouse is always above the same point. Because this is underdetermined, we also add the constraint that the camera should stay fixed in the vertical plane. Note that this is not the only constraint possible. Other constraints may restrict the camera to the surface of a sphere around the scene for example.

A.7. Track Sparsity

For camera control and motion transfer motion prompts, we obtain a dense set of tracks. Empirically, we find that it is helpful to randomly subsample these tracks, as using too many tracks suppresses the video model's learned priors from working, while using too few affords too little control. Somewhere in the middle is a sweet spot. For exam-

ple, for the majority of the depth-based motion prompts, we use 1024 tracks, which we find offers a good balance between control and emergent video prior effects. In other cases, such as transferring the motion of the person's face in Fig. 8, we find that fewer tracks is helpful in dealing with misalignments between the source video and the input first frame. Finally, for out-of-domain motion transfer as in the *monkey chewing* example in Fig. 8, we find that very dense tracks help. We use 1500 tracks, as we need a lot of control to apply such an unnatural motion to the first frames.

A.8. Davis Eval

We conduct a quantitative evaluation of first frame, text, and track conditioned video generation using the DAVIS validation dataset, with a subset of results in Tab. 1 and full results in Tab. A2. The validation dataset contains 30 videos from a wide range of scenes, involving subjects from sports to humans to animals to cars. In order to create inputs for the models, we extract tracks using BootsTAP [13]. First frame inputs are square crops of the first frames of the videos, and text prompts are the titles of the videos given by the dataset and typically consist of a word or two. For each evaluation for a given number of tracks, we randomly sample that number of tracks for conditioning.

To ensure a fair comparison, we make the following accommodations for baselines. In addition to a first frame, tracks, and a text prompt, DragAnything requires segmentation masks for objects that the tracks move. To get this, we use the provided ground truth segmentations in the DAVIS dataset. Image Conductor is a finetuned version of AnimateDiff and is trained on videos of resolution 384×256. We initially gave the model 256×256 images, and found that we got reasonable results. However, we experimented with reflection padding the input frame to 384×256 and cropping the output, which gave slightly better results which we report.

A.9. Human Studies

To perform the human studies, we handcraft 30 inputs with diverse image subjects and input motions. Motions consist of a single uninterrupted trajectory. Text prompts are designed to describe the image, but not the desired motion, to factor out the influence of text as much as possible. DragAnything requires masks, which we obtain by running SAM [38] on the first frame with the initial location of the tracks as query points. For our method, we turn the trajectory into a grid of tracks as described above. We then feed these inputs to the models and take a single random sample. We follow the same protocol as above for Image Conductor. This results in 30 samples for each model and 90 samples in total.

We run a two alternative forced choice (2AFC) test between our model and the baselines. We display a sample

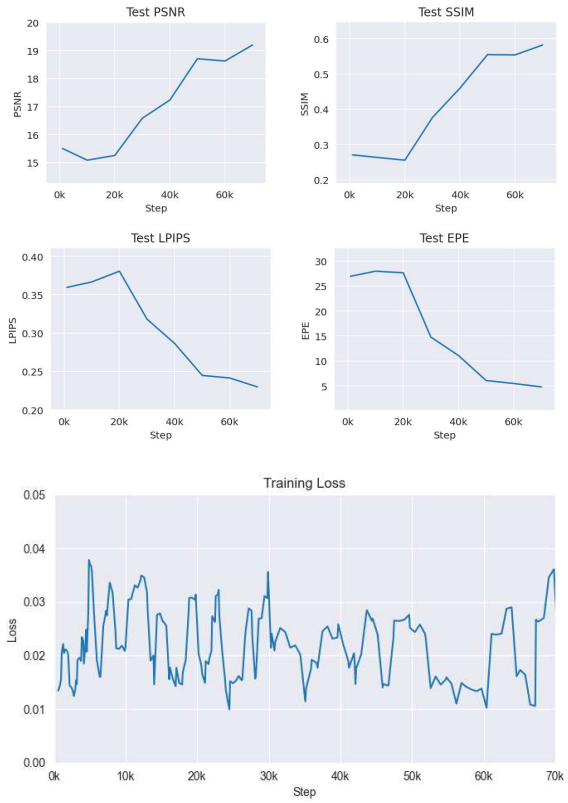

Figure A2. **Test and Train Metrics.** Here we plot out training loss, along with PSNR, SSIM, LPIPS, and EPE on our DAVIS test set. Note that there is no correlation between the training loss and the test metrics, and that the test metrics show no signs of improvement until step 20,000 at which point the network learns quite rapidly.

from our method and a sample from the baseline in a random order with a video of the corresponding motion conditioning in the middle, visualized as a moving red dot. Participants are then asked three questions. Verbatim, we ask (1) *Which video better follows the motion of the red dot?* (2) *Which video has the more realistic motion?* (3) *Which video is of higher visual quality?* These questions are designed to measure the adherence of the motion to the conditioning, the quality of the motion, and the overall visual quality of video, respectively. This results in a total of 180 questions.

We recruit participants for our study through Amazon Mechanical Turk (MTurk). To ensure responses are of high quality, we add three "vigilance" questions with clearly correct answers. We discard all responses that fail any of these three questions. Each question is conducted as a separate study, and the resulting number of participants are $N = 103$, $N = 103$, and $N = 115$ for each question respectively. This results in a total of 19,260 answers.

B. Training Observations

In training, we observe similar behavior as noted in ControlNet [87] and ControlNext [50]: 1) training loss does not directly correlate with model performance, and 2) "sudden convergence" where in a few epochs the model goes from not adhering to control signal to full adherence. ControlNext identifies both of these behaviors as coming from the zero initialization and offers cross normalization as a potential solution. We believe this and other future control scheme is a promising avenue for future work in track conditioned video generation. We show training loss and test metrics in Fig. A2. As can be seen, the training loss is fairly inscrutable, while the test losses do not begin to decrease until step 20,000.

C. Full Quantitative Results

In Sec. 5 we present DAVIS evaluation results for $N = \{1, 16, 512, 2048\}$. In Tab. A2 we present results for $N = 4$ and $N = 64$ as well, which we omit from the main paper for brevity.

Table A2. **Quantitative Evaluations.** We evaluate the appearance (PSNR, SSIM, LPIPS, FVD) and motion (EPE) of generated videos using the validation set of the DAVIS dataset. Please note that each method is trained from a different base model.

# Tracks	Method	PSNR ↑	SSIM ↑	LPIPS ↓	FVD ↓	EPE ↓
N = 1	Image Conductor	11.468	0.145	0.529	1919.8	19.224
	DragAnything	14.589	0.241	0.420	1544.9	**9.135**
	Ours	**15.431**	**0.266**	**0.368**	**1445.2**	14.619
N = 4	Image Conductor	12.017	0.176	0.499	1735.0	18.921
	DragAnything	15.040	0.272	0.397	1497.2	**8.946**
	Ours	**15.820**	**0.319**	**0.353**	**1207.7**	12.985
N = 16	Image Conductor	12.184	0.175	0.502	1838.9	24.263
	DragAnything	15.119	0.305	0.378	**1282.8**	9.800
	Ours	**16.618**	**0.405**	**0.319**	1322.0	**8.319**
N = 64	Image Conductor	12.513	0.180	0.503	1947.7	26.316
	DragAnything	14.499	0.274	0.393	1342.0	10.642
	Ours	**18.000**	**0.513**	**0.265**	**951.4**	**4.127**
N = 512	Image Conductor	11.902	0.132	0.524	1966.3	30.734
	DragAnything	15.055	0.289	0.381	1379.8	10.948
	Ours	**18.968**	**0.583**	**0.229**	**688.7**	**4.055**
N = 2048	Image Conductor	11.609	0.120	0.538	1890.7	33.561
	DragAnything	14.845	0.286	0.397	1468.4	12.485
	Ours	**19.327**	**0.608**	**0.227**	**655.9**	**3.887**

D. Human Pose Control

We show results on using our method to control humans through human pose estimated keypoints in Fig. A3. To do this, we first estimate the pose with an off-the-shelf model, and then apply motions to desired keypoints and feed it to our model.

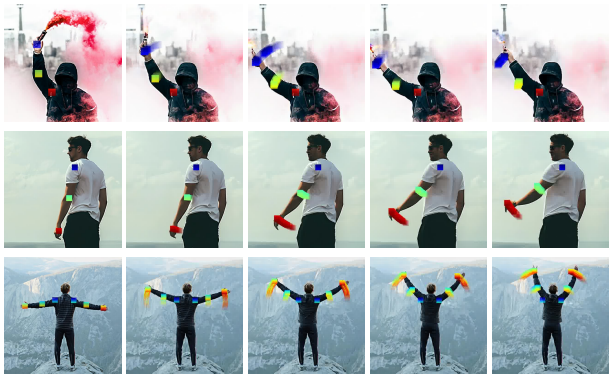

Figure A3. **Pose Conditioning.** We estimate human pose, animate it, translate it to tracks, and then feed it to our model. In each row, we show frames from generated videos with input tracks overlaid on top.

E. Motion Magnification

One additional application of our model is motion magnification [40, 47, 49, 71, 77]. This task involves taking a video with subtle motions and generating a new video in which these motions have been magnified, so that they are easier to see. We do this by running a tracking algorithm [12] on an input video, smoothing the tracks by applying a Gaussian blur over space and time, and then magnifying the resulting tracks. We then feed the first frame of the input video and the magnified tracks to our model. We show results, along with space-time slices, in Fig. A4. We found that smoothing was necessary to reduce noise in the estimated tracks. As a result the magnified tracks are not exactly at the specified magnification factor, but nonetheless are qualitatively useful in revealing subtle motions. We expect more accurate point tracking algorithms will remove the need for this smoothing step.

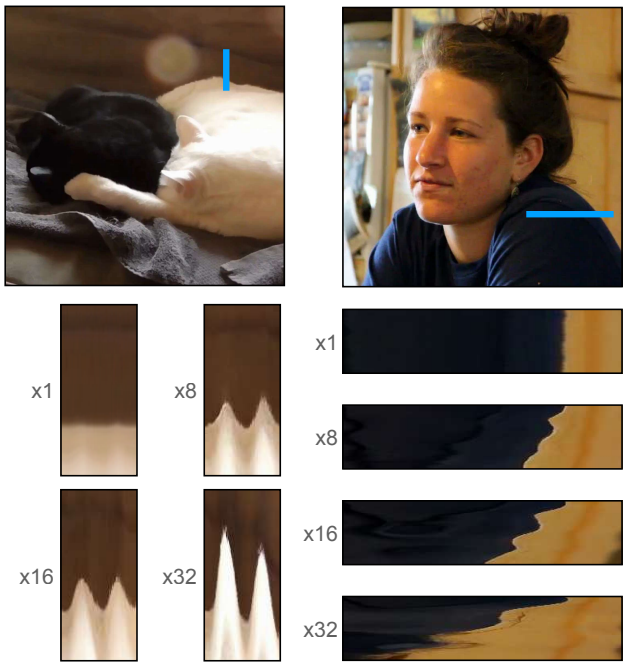

Figure A4. **Motion Magnification.** We show the result of using our model to perform motion magnification. We show the first frame of two videos, and space-time slices through the blue line at different magnification factors.