3D Cinemagraphy from a Single Image

Xingyi Li\textsuperscript{1,3} Zhiguo Cao\textsuperscript{1} Huiqiang Sun\textsuperscript{1} Jianming Zhang\textsuperscript{2} Ke Xian\textsuperscript{3}* Guosheng Lin\textsuperscript{3}

\textsuperscript{1}Key Laboratory of Image Processing and Intelligent Control, Ministry of Education
School of Artificial Intelligence and Automation, Huazhong University of Science and Technology
\textsuperscript{2}Adobe Research \textsuperscript{3}S-Lab, Nanyang Technological University

\{xingyi<li,zgcao,shq1031}@hust.edu.cn, jianmzha@adobe.com, \{ke.xian,gslin\}@ntu.edu.sg

https://xingyi-li.github.io/3d-cinemagraphy

\textbf{Abstract}

We present 3D Cinemagraphy, a new technique that marries 2D image animation with 3D photography. Given a single still image as input, our goal is to generate a video that contains both visual content animation and camera motion. We empirically find that naively combining existing 2D image animation and 3D photography methods leads to obvious artifacts or inconsistent animation. Our key insight is that representing and animating the scene in 3D space offers a natural solution to this task. To this end, we first convert the input image into feature-based layered depth images using predicted depth values, followed by unprojecting them to a feature point cloud. To animate the scene, we perform motion estimation and lift the 2D motion into the 3D scene flow. Finally, to resolve the problem of hole emergence as points move forward, we propose to bidirectionally displace the point cloud as per the scene flow and synthesize novel views by separately projecting them into target image planes and blending the results. Extensive experiments demonstrate the effectiveness of our method. A user study is also conducted to validate the compelling rendering results of our method.

\*Corresponding author.

\textbf{1. Introduction}

Nowadays, since people can easily take images using smartphone cameras, the number of online photos has increased drastically. However, with the rise of online video-sharing platforms such as YouTube and TikTok, people are no longer content with static images as they have grown accustomed to watching videos. It would be great if we could animate those still images and synthesize videos for a better experience. These living images, termed cinemagraphs, have already been created and gained rapid popularity online \cite{1,71}. Although cinemagraphs may engage people with the content for longer than a regular photo, they usually fail to deliver an immersive sense of 3D to audiences. This is because cinemagraphs are usually based on a static camera and fail to produce parallax effects. We are therefore motivated to explore ways of animating the photos and moving around the cameras at the same time. As shown in Fig. \textbf{1}, this will bring many still images to life and provide a drastically vivid experience.

In this paper, we are interested in making the first step towards 3D cinemagraphy that allows both realistic animation of the scene and camera motions with compelling parallax effects from a single image. There are plenty of attempts to tackle either of the two problems. Single-image animation methods \cite{12,19,35} manage to produce a real-
istic animated video from a single image, but they usually operate in 2D space, and therefore they cannot create camera movement effects. Classic novel view synthesis methods [5, 6, 9, 14, 25] and recent implicit neural representations [37, 40, 58] entail densely captured views as input to render unseen camera perspectives. Single-shot novel view synthesis approaches [21, 39, 52, 66] exhibit the potential for generating novel camera trajectories of the scene from a single image. Nonetheless, these methods usually hypothesize that the observed scene is static without moving elements. Directly combining existing state-of-the-art solutions of single-image animation and novel view synthesis yields visual artifacts or inconsistent animation.

To address the above challenges, we present a novel framework that solves the joint task of image animation and novel view synthesis. This framework can be trained to create 3D cinemagraphs from a single still image. Our key intuition is that handling this new task in 3D space would naturally enable both animation and moving cameras simultaneously. With this in mind, we first represent the scene as feature-based layered depth images (LDIs) [50] and unproject the feature LDIs into a feature point cloud. To animate the scene, we perform motion estimation and lift the 2D motion to 3D scene flow using depth values predicted by DPT [45]. Next, we animate the point cloud according to the scene flow. To resolve the problem of hole emergence as points move forward, we are inspired by prior works [3, 19, 38] and propose a 3D symmetric animation technique to bidirectionally displace point clouds, which can effectively fill in those unknown regions. Finally, we synthesize novel views at time $t$ by rendering point clouds into target image planes and blending the results. In this manner, our proposed method can automatically create 3D cinemagraphs from a single image. Moreover, our framework is highly extensible, e.g., we can augment our motion estimator with user-defined masks and flow hints for accurate flow estimation and controllable animation.

In summary, our main contributions are:

- We propose a new task of creating 3D cinemagraphs from single images. To this end, we propose a novel framework that jointly learns to solve the task of image animation and novel view synthesis in 3D space.
- We design a 3D symmetric animation technique to address the hole problem as points move forward.
- Our framework is flexible and customized. We can achieve controllable animation by augmenting our motion estimator with user-defined masks and flow hints.

2. Related Work

Single-image animation. Different kinds of methods have been explored to animate still images. Some works [8, 22] focus on animating certain objects via physical simulation but may not be easily applied to more general cases of in-the-wild photos. Given driving videos as guidance, there are plenty of methods that attempt to perform motion transfer on static objects with either a priori knowledge of moving objects [7, 11, 33, 46, 55] or in an unsupervised manner [53, 54, 56]. They entail reference videos to drive the motion of static objects, and thus do not suit our task. Recent advances in generative models have attracted much attention and motivated the community to develop realistic image and video synthesis methods. Many works [31, 32, 34, 51, 69] are based on generative adversarial networks (GANs) and operate transformations in latent space to generate plausible appearance changes and movements. Nonetheless, it is non-trial to allow for explicit control over those latent codes and to animate input imagery in a disentangled manner. As diffusion models [17, 59] improve by leaps and bounds, several diffusion-based works [16, 18, 57] attempt to generate realistic videos from text or images. However, these methods are time-consuming and expensive in terms of computation. Here we focus on methods that utilize learned motion priors to convert a still image into an animated video texture [12, 13, 19, 29, 35]. In particular, Holynski et al. [19] first synthesize the optical flow of the input image via a motion estimation network, then obtain future frames using the estimated flow field. This method renders plausible animation of fluid elements in the input image but suffers from producing camera motions with parallax.

Novel view synthesis from a single image. Novel view synthesis allows for rendering unseen camera perspectives from 2D images and their corresponding camera poses. Recent impressive synthesis results may credit to implicit neural representations [37, 40, 58]. Nevertheless, these methods usually assume dense views as input, which is not always available in most cases. Moreover, they focus on the task of interpolation given multiple views rather than extrapolation. As such, we instead turn to methods aiming at handling single input. Among them, a number of works [15, 26, 28, 62, 63, 70, 72] infer the 3D structure of scenes by learning to predict a scene representation from a single image. These methods are usually trained end-to-end but suffer from generalizing to in-the-wild photos. Most relevant to our work are those approaches [39, 52, 66] that apply depth estimation [45, 65, 67, 68] followed by inpainting occluded regions. For example, 3D Photo [52] estimates monocular depth maps and uses the representation of layered depth images (LDIs) [43, 50], in which context-aware color and depth inpainting are performed. To enable fine-grained detail modeling, SLIDE [21] decomposes the scene into foreground and background via a soft-layering scheme. However, unlike our approach, these methods usually assume the scene is static by default, which largely lessens the sense of reality, especially when some elements such as
3. Method

3.1. Overview

Given a single still image, our goal is to synthesize plausible animation of the scene and simultaneously enable camera motion. The output of our method is a realistic cinemagraph with compelling parallax effects. Fig. 2 schematically illustrates our pipeline. Our method starts by estimating a motion field and a depth map from the input image. We then separate the RGBD input into several layers as per depth discontinuities and inpaint occluded regions, followed by extracting 2D feature maps for each layer, resulting in feature LDIs $L$. Subsequently, we lift feature LDIs into 3D space using corresponding depth values to obtain a feature point cloud $P$. To animate the scene, we estimate a 2D motion field from the input image and apply Euler integration to generate forward and backward displacement fields $F_{0 \rightarrow t}$ and $F_{0 \rightarrow t-N}$. We then augment displacement fields with estimated depth values to obtain 3D scene flow fields.

Next, we bidirectionally displace the feature point cloud $P$ as per the scene flow and separately project them into target image planes to obtain $F_f$ and $F_b$. Finally, we blend them together and pass the result through our image decoder to synthesize a novel view at time $t$.

4597
where $F_{t \rightarrow t+1}(\cdot)$ represents the optical flow map from frame $t$ to frame $t+1$. This defines how each pixel in the current frame will move in the future. Specifically, we can obtain the next frame via Euler integration:

$$x_{t+1} = x_t + M(x_t),$$

where $x_t$ represents the coordinates of a pixel at time $t$.

Since the optical flow between consecutive frames is identical, we can easily deduce the displacement field by recursively applying:

$$F_{0 \rightarrow t}(x_0) = F_{0 \rightarrow t-1}(x_0) + M(x_0 + F_{0 \rightarrow t-1}(x_0)), \quad (3)$$

where $F_{0 \rightarrow t}(\cdot)$ denotes the displacement field from time 0 to time $t$, which describes the course of each pixel in the input image across future frames. To estimate the Eulerian flow field, we adopt an image-to-image translation network as our motion estimator, which is able to map an RGB image to the optical flow.

### 3.3. 3D Scene Representation

One common disadvantage of previous single-image animation methods [12, 19, 29] is that they usually operate in 2D space via a deep image warping technique, which prevents them from creating parallax effects. Instead, to enable camera motion, we propose to lift our workspace into 3D and thus resort to 3D scene representation.

We start by estimating the underlying geometry of the scene using the state-of-the-art monocular depth estimator DPT [45], which can predict reasonable dense depth maps for in-the-wild photos. Following Wang et al. [64], we then convert the RGBD input into an LDI representation [50] by separating it into several layers as per depth discontinuities and inpainting occluded regions. Specifically, we first divide the depth range of the source depth map into multiple intervals using agglomerative clustering [36], followed by creating layered depth images $L = \{C_l, D_l\}_{l=1}^L$. Next, we inpaint occluded regions of each color and depth layer by applying the pretrained inpainting model from 3D Photo [52]. To improve rendering quality and reduce artifacts, we also introduce a 2D feature extraction network to encode 2D feature maps for each inpainted LDI color layer, resulting in feature LDIs $F = \{F_l, D_l\}_{l=1}^L$. Finally, in order to enable animation in 3D space, we unproject feature LDIs into 3D via their corresponding inpainted depth layers, yielding a feature point cloud $P = \{(X_i, f_i)\}$, where $X_i$ and $f_i$ are 3D coordinates and the feature vector for each 3D point respectively.

### 3.4. Point Cloud Animation and Rendering

We now have the estimated displacement fields $F_{0 \rightarrow t}$ and the feature point cloud $P$. Our next step is to animate this point cloud over time. To bridge the gap between 2D displacement fields and 3D scene representation, we first augment the displacement fields with estimated depth values to lift them into 3D scene flow. In other words, we now have a function of time $t$ and the coordinates of a 3D point that returns a corresponding 3D translation vector that can shift this 3D point accordingly. Thus, for time $t$, we then move each 3D point by computing its destination as its original position plus a corresponding 3D translation vector, i.e., $P(t) = \{(X_i(t), f_i)\}$. Intuitively, this process indeed animates the point cloud from one time to another. However, we empirically find that as points move forward, increasingly large holes emerge. This frequently happens when points leave their original locations without any points filling in those unknown regions.

#### 3D symmetric animation

To resolve this, inspired by prior works [3, 19, 38], we propose a 3D symmetric animation technique that leverages bidirectionally displaced point clouds to complement each other. With 3D symmetric animation, we can borrow textural information from point clouds that move in the opposite direction and integrate both of the animated point clouds to feasibly fill in the missing regions (the red and blue regions).

To address the hole issue, we borrow textural information from the point cloud that moves in the opposite direction and integrate both of the animated point clouds to feasibly fill in the missing regions (the red and blue regions).
lar, given camera poses and intrinsics, we use a differentiable point-based renderer [66] to splat feature point clouds $\mathcal{P}_f(t) = \{ (X_f^t(t), f_i) \}$ and $\mathcal{P}_b(t) = \{ (X_b^t(t), f_i) \}$ separately into the target image plane. This process yields 2D feature maps $F_f$ and $F_b$ along with depth maps $D_f$, $D_b$ and alpha maps $\alpha_f$, $\alpha_b$. Next, we wish to fuse $F_f$ and $F_b$ into one feature map $F_t$. Inspired by prior work [64], our intuition is three-fold: 1) to enable endless and seamless looping, we should assign the weight of the two feature maps based on time so as to guarantee that the first and last frame of the synthesized video are identical; 2) the weight map should favor pixel locations with smaller depth values, in the sense that it is impossible to see objects behind those objects closer to the eye; 3) to avoid missing regions as much as possible, we should greatly increase the contribution of those pixel locations that can fill in holes. With this in mind, we formulate the weight maps as follows:

$$W_t = \frac{(1 - \frac{t}{N}) \cdot \alpha_f \cdot e^{-D_f} + \frac{t}{N} \cdot \alpha_b \cdot e^{-D_b}}{(1 - \frac{t}{N}) \cdot \alpha_f \cdot e^{-D_f} + \frac{t}{N} \cdot \alpha_b \cdot e^{-D_b}},$$

(4)

where $N$ is the number of frames. Therefore, we can integrate $F_f$ and $F_b$ via:

$$F_t = W_t \cdot F_f + (1 - W_t) \cdot F_b.$$  

(5)

We also obtain the merged depth map $D_t$:

$$D_t = W_t \cdot D_f + (1 - W_t) \cdot D_b.$$  

(6)

Finally, we employ an image decoder network to map the 2D feature map $F_t$ and depth map $D_t$ to a novel view at time $t$. Repeating this method, we are able to synthesize a realistic cinemagraph with compelling parallax effects.

### 3.5. Training

This section describes our training scheme. In general, we train our image-to-image translation network, 2D feature extraction network, and image decoder network in a two-stage manner.

**Training dataset.** We use the training set from Holynski et al. [19] as our training dataset. This dataset comprises short video clips of fluid motion that are extracted from longer stock-footage videos. We use the first frames of each video clip and the corresponding ground truth motion fields estimated by a pretrained optical flow network [60] as motion estimation pairs to train our motion estimation network. To develop animation ability, we randomly sample training data from fluid motion video clips. For novel view synthesis training, we require multi-view supervision of the same scene, which is not available in the training set. Instead, we use 3D Photo [52] to generate pseudo ground truth novel views for training.

**Two-stage training.** Our model is trained in a two-stage manner. Specifically, we first train our motion estimation network using motion estimation pairs. To train the motion estimation network, we minimize GAN loss, GAN feature matching loss [49], and endpoint error as follows:

$$L_{Motion} = L_{GAN} + 10L_{FM} + L_{EPE}.$$  

(7)

In the second stage, we freeze the motion estimation network and train the feature extraction network and image decoder network. Our model simultaneously learns to render novel views and animate scenes. For novel view synthesis, we set $t = 0$ and use pseudo ground truth novel views to supervise our model. We randomly sample target viewpoints of scenes and require the model to synthesize them. For animation, we train our model on training triplets (start frame, middle frame, end frame) sampled from fluid motion video clips. In particular, we render the middle frame from both directions using $F_{0\rightarrow t}$ and $F_{0\rightarrow t-N}$ without changing the camera poses and intrinsics. Besides GAN loss and GAN feature matching loss [49], we also enforce VGG perceptual loss [23, 73] and $l_1$ loss between synthesized and ground truth images. The overall loss is as follows:

$$L_{Animation} = L_{GAN} + 10L_{FM} + L_{l_1} + L_{VGG}.$$  

(8)

### 4. Experiments

#### 4.1. Implementation Details

Our motion estimator is a U-Net [48] based generator with 16 convolutional layers, and we replace Batch Normalization with SPADE [42]. For the feature extraction network and image decoder network, we follow the network architectures from Wang et al. [64]. We adopt the multi-scale discriminator used in SPADE [42] during training.

Our model is trained using the Adam optimizer [24]. We conduct all experiments on a single NVIDIA GeForce RTX 3090 GPU. We train the motion estimation network for around 120k iterations with a batch size of 16. We set the generator learning rate to $5 \times 10^{-4}$ and the discriminator learning rate to $2 \times 10^{-3}$. For the animation training stage, we train the feature extraction network and image decoder network for around 250k iterations with a learning rate starting at $1 \times 10^{-4}$ and then decaying exponentially.

#### 4.2. Baselines

In principle, to evaluate our method, we are required to compare it against current state-of-the-art models. However, to our knowledge, we are the first to tackle the novel task of synthesizing a realistic cinemagraph with compelling parallax effects from a single image. As a result, we cannot directly compare to previous works. Instead, we consider forming the following baselines to verify the superiority of our method:

**2D animation $\rightarrow$ novel view synthesis.** One might consider 2D image animation $\rightarrow$ single-shot novel view synthesis: first employing a 2D image animation method, then
Experimental setup. For evaluation, we render novel views of the ground truth videos in Naive point cloud animation. Intuitively, we may also result in unique scenes with the validation set consisting of the validation set from Holynski et al. [19] to evaluate our method and baselines. The validation set consists of a single image for each scene in the test set, we use Photo [52] and then use the image animation method [19] to animate each viewpoint. Note that motion estimation should be performed for each frame as viewpoints have changed. However, we empirically find that this usually results in varying motion fields across the video. To mitigate this, we further propose using the moving average technique to smooth estimated motions for each frame. This results in varying motion fields across the video and thus inconsistent animation. Although Naive PC Anim. and Naive PC Anim. + 3DSA are prone to produce noticeable holes inevitably. One reason for this is that they do not perform inpainting. Note that some artifacts are difficult to observe when only scanning static figures.

Controllable animation. Our method is able to create 3D cinematographs from a single image automatically. Further, we show that our framework is also highly extensible. For example, we can involve masks and flow hints as extra inputs. For a single-shot novel view synthesis method, specifically, we first adopt a state-of-the-art image animation method [19] to produce an animated looping video. We then apply DPT [45] to estimate geometry and utilize 3D Photo [52] to generate novel views for each frame.

Novel view synthesis $\rightarrow$ 2D animation. It also appears to be feasible that we first render novel views of scenes by 3D Photo [52] and then use the image animation method [19] to animate each viewpoint. This process does not involve inpainting, thus ground truth frames at the same time and viewpoint. For a fair comparison, all methods utilize the depth maps estimated by DPT [45]. Since we focus on comparing rendering quality, all methods use ground truth optical flows, except that NVS [52] $\rightarrow$ 2D Anim. [19] and NVS [52] $\rightarrow$ 2D Anim. [19] + MA have to estimate optical flows for each frame apart from the first frame. We adopt PSNR, SSIM, and LPIPS [73] as our evaluation metrics.

Quantitative comparisons. As shown in Table 1, our method outperforms all baselines across all metrics by a large margin. This result implies that our method achieves better perceptual quality and produces more realistic renderings, which demonstrates the superiority and effectiveness of our method.

Qualitative comparisons. We showcase the visual comparisons in Fig. 4. One can observe that our method presents photorealistic results while other comparative baselines produce more or less visual artifacts. 2D Anim. [19] $\rightarrow$ NVS [52] intends to generate stripped flickering artifacts. This is because 2D Anim. [19] $\rightarrow$ NVS [52] predicts the depth map for each animated frame, leading to frequent changes in the 3D structure of the scene and inconsistent inpainting. NVS [52] $\rightarrow$ 2D Anim. [19] and NVS [52] $\rightarrow$ 2D Anim. [19] + MA show jelly-like effects as optical flow should be estimated for each novel view. This results in varying motion fields across the video and thus inconsistent animation. Although Naive PC Anim. and Naive PC Anim. + 3DSA also lift the workspace into 3D, they are often prone to produce noticeable holes inevitably. One reason for this is that they do not perform inpainting. Note that some artifacts are difficult to observe when only scanning static figures.

4.3. Results

Evaluation dataset. Since Holynski et al. [19] only provide a single image for each scene in the test set, we use the validation set from Holynski et al. [19] to evaluate our method and baselines. The validation set consists of 31 unique scenes with 162 samples of ground truth video clips captured by static cameras.

Experimental setup. For evaluation, we render novel views of the ground truth videos in 4 different trajectories, resulting in 240 ground truth frames for each sample. This process does not involve inpainting, thus ground truth frames may contain holes. Only considering valid pixels when calculating metrics, we compare the predicted images with the ground truth frames at the same time and viewpoint. For a fair comparison, all methods utilize the depth maps estimated by DPT [45]. Since we focus on comparing rendering quality, all methods use ground truth optical flows, except that NVS [52] $\rightarrow$ 2D Anim. [19] and NVS [52] $\rightarrow$ 2D Anim. [19] + MA have to estimate optical flows for each frame apart from the first frame. We adopt PSNR, SSIM, and LPIPS [73] as our evaluation metrics.

Quantitative comparisons. As shown in Table 1, our method outperforms all baselines across all metrics by a large margin. This result implies that our method achieves better perceptual quality and produces more realistic renderings, which demonstrates the superiority and effectiveness of our method.

Qualitative comparisons. We showcase the visual comparisons in Fig. 4. One can observe that our method presents photorealistic results while other comparative baselines produce more or less visual artifacts. 2D Anim. [19] $\rightarrow$ NVS [52] intends to generate stripped flickering artifacts. This is because 2D Anim. [19] $\rightarrow$ NVS [52] predicts the depth map for each animated frame, leading to frequent changes in the 3D structure of the scene and inconsistent inpainting. NVS [52] $\rightarrow$ 2D Anim. [19] and NVS [52] $\rightarrow$ 2D Anim. [19] + MA show jelly-like effects as optical flow should be estimated for each novel view. This results in varying motion fields across the video and thus inconsistent animation. Although Naive PC Anim. and Naive PC Anim. + 3DSA also lift the workspace into 3D, they are often prone to produce noticeable holes inevitably. One reason for this is that they do not perform inpainting. Note that some artifacts are difficult to observe when only scanning static figures.

Controllable animation. Our method is able to create 3D cinematographs from a single image automatically. Further, we show that our framework is also highly extensible. For example, we can involve masks and flow hints as extra in-
Our method produces compelling results while other comparative alternatives suffer from visual artifacts. (a) 2D animation \[19\] → novel view synthesis \[52\], (b) novel view synthesis \[52\] → 2D animation \[19\], (c) novel view synthesis \[52\] → 2D animation \[19\] + moving average, (d) naive point cloud animation, (e) naive point cloud animation + 3D symmetric animation, (f) our method, and (g) pseudo ground truth.

inputs to augment our motion estimator. This brings two advantages: (1) more accurate flow estimation; (2) interactive and controllable animation. As shown in Fig. 5, we can control the animation of the scene by providing various masks and motion hints to obtain different motion fields.

Generalizing on in-the-wild photos. To further demonstrate the generalization of our method, we also test our method on in-the-wild photos. We first create cinemagraphs with camera motions on the test set from Holynski et al. \[19\], where, for each scene, only a single image is provided. We then select some online images at random to test our method. To accurately estimate motion fields, we provide masks and flow hints as extra inputs to our motion estimator. As shown in Fig. 6, our method produces reasonable results for in-the-wild inputs while other comparative alternatives yield visual artifacts or inconsistent animation.

4.4. User Study

We further conduct a user study to investigate how our method performs in the view of humans when compared with all baselines, 3D Photo \[52\], and Holynski et al. \[19\]. Specifically, we collect 50 photos from the test set of Holynski et al. \[19\] and the Internet. We use different approaches to generate videos with identical settings. During the study, we show each participant an input image and two animated videos generated by our method and a randomly selected approach in random order. 108 volunteers are invited to choose the method with better perceptual quality and realism, or none if it is hard to judge. We report the results in Table 2, which points out that our method surpasses alternative methods by a large margin in terms of the sense of reality and immersion.

4.5. Ablation Study

To validate the effect of each component, we conduct an ablation study on the validation set from Holynski et al. \[19\] and show the results in Table 3. One can observe: i) 3D symmetric animation technique matters because it allows us to leverage bidirectionally displaced point clouds to complement each other and feasibly fill in missing regions; ii)
introducing inpainting when constructing 3D geometry can improve the performance as this allows our model to produce plausible structures around depth discontinuities and fill in holes; iii) switching from directly using RGB colors to features in 3D scene representation significantly improves the rendering quality and reduces artifacts.

5. Conclusion

In this paper, we introduce a novel task of creating 3D cinemagraphs from single images. To this end, we present a simple yet effective method that makes a connection between image animation and novel view synthesis. We show that our method produces plausible animation of the scene while allowing camera movements. Our framework is flexible and customized. For accurate motion estimation and controllable animation, we can further include masks and flow hints as extra input for the motion estimator. Therefore, users can control how the scene is animated. Furthermore, our method generalizes well to in-the-wild photos, even like paintings or synthetic images generated by diffusion models. We conduct extensive experiments to verify the effectiveness and superiority of our method. A user study also demonstrates that our method generates realistic 3D cinemagraphs. We hope that our work can bring 3D cinemagraphy into the sight of a broader community and motivate further research.

Limitations and future work. Our method may not work well when the depth prediction module estimates erroneous geometry from the input image, e.g., thin structures. In addition, inappropriate motion fields will sometimes lead to undesirable results, e.g., some regions are mistakenly identified as frozen. As we take the first step towards 3D cinemagraphy, in this paper, we focus on handling common moving elements, i.e., fluids. In other words, our method may not apply to more complex motions, e.g., cyclic motion. We leave this for our future work.

Acknowledgements. This study is supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s). This work is also supported by Adobe Gift and the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (MOE-T2EP20220-0007) and Tier 1 (RG14/22).
References

[26] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. MINE: Towards continuous depth mpi with nerf for novel view synthesis. In Proceedings of
the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 12578–12588, 2021. 2


