Animating Cloud Images With Flow Style Transfer

KAZUMA KURISAKI¹ AND KAZUHIKO KAWAMOTO², (Member, IEEE)

¹Graduate School of Science and Engineering, Chiba University, Chiba 263-8522, Japan
²Graduate School of Engineering, Chiba University, Chiba 263-8522, Japan

Corresponding author: Kazuhiko Kawamoto (kawa@faculty.chiba-u.jp)

This work was supported by Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP19K12039.

ABSTRACT We propose a method for animating static images using a generative adversarial network (GAN). Given a source image depicting a cloud image and a driving video sequence depicting a moving cloud image, our framework generates a video in which the source image is animated according to the driving sequence. By inputting the source image and optical flow of the driving video into the generator, a video is generated that is conditioned by the optical flow. The optical flow enables the application of the captured motion of clouds in the source image. Further, we experimentally show that the proposed method is more effective than the existing methods for animating a keypoint-less video (in which the keypoints cannot be explicitly determined) such as a moving cloud image. Furthermore, we show an improvement in the quality of the generated video due to the use of optical flow in the video reconstruction.

INDEX TERMS Image animation, video generation, generative adversarial networks, optical flow.

I. INTRODUCTION Generating a high-quality video from a single image is a challenging task and has garnered considerable research attention in recent years. Accordingly, several methods have been proposed to generate videos with structural conditioning such as skeleton tracking, use of object keypoints, utilization of depth video together with color video, semantic segmentation, and optical flow [1]–[5].

Most of the existing studies on video generation are based on deep learning methods such as generative adversarial networks (GAN) [6] and variational autoencoder (VAE) [7]. These methods extract video representations from large data, which are used to generate videos. In the case of image generation, the generated image is indistinguishable from the real one. V-GAN [4] is the first GAN for video generation. Recently, Clark et al. [8] proposed dual video discriminator-GAN (DVD-GAN) to generate real videos without any conditioning. When video generation methods are used for real problems, the control of the generated video is often desirable. Furthermore, it is convenient if an animation can be generated from a single image. Thus, in this study, we propose a method for animating a single arbitrary image of cloud. However, motion conditioning is complicated, and it is difficult to represent the motion as a simple label, word, or formula. To solve this issue, we use a driving video as a reference input to drive the motion of the source image. In other words, the single image is conditioned using the motion of the driving video. By using a driving video as motion representation, we can animate a static cloud picture according to the CG-generated motion derived from the driving video. Models such as X2face [9] and Monkey-Net [1] also utilize driving video to animate static images. In these models, the skeleton or keypoints extracted from a video are used to generate a video, which is animated according to the sequence of keypoints. Nevertheless, these methods require keypoints in the video. Therefore, they are not suitable for cloud videos in which the key points cannot be explicitly determined. To this end, the proposed method utilizes a cloud image (the cloud has no joints and is deformed during motion) as a keypoint-less video. We use dense optical flow to obtain motion representation from the driving video and generate a video by conditioning on the acquired optical flow. Consequently, the generated video is animated according to the driving video sequence. Some examples of video generation based on the proposed method are shown in Fig. 1.

Overall, the key contributions of this study are as follows:

- We propose a method for animating a single image using a driving video depicting cloud image video as keypoint-less video.
video by generating and color-converting depth videos. Flow-and-texture GAN (FTGAN) [5] generates a video by generating and color-converting optical flows. These studies have improved the accuracy of video generation by training the models on the physical structure. Our method also uses GAN with optical flow as the physical structure.

B. IMAGE ANIMATION
Image animation refers to the task of generating a video from a single image rather than from a latent code along with the ability to control the characteristics (e.g., motion, color, etc.) of the generated video.

Xiong et al. proposed multi-discriminator-GAN (MD-GAN) [12] to generate a video from a single image. MD-GAN includes two stages, and the generator and discriminator are composed of 3D convolutions. In the first stage, a video is roughly generated, and this video is refined in the second stage. This architecture is inspired by stack-GAN [13]. In the second stage, a loss function is used so that the generated video is closer to the real video. Li et al. proposed flow-grounded VAE method [14] to generate an optical flow from a latent code, which was used along with a previous frame to generate the next frame. The use of optical flows facilitates a smooth and continuous generation of the video. Endo et al. proposed a method for animating high-resolution and long-sequence landscape images [15]. They generated the video by recursively generating the next frame from the previous frame. Their model included motion prediction and appearance prediction phases. In the motion prediction phase, optical flow was generated, and the frame was warped with this optical flow to generate the video. In the appearance prediction phase, the color of the video was varied over time. This method can roughly control the motion and appearance (direction of cloud flow and sky color) of the generated video by adjusting the latent code. Cheng et al. proposed a method of time-lapse video image animation [16]. This is similar to our method, but with a distinct difference. Their method focuses on changing the color of the still image along with the driving video and does not consider motion. In contrast, we focus on motion. Siarohin et al. proposed Monkey-Net [1], which generates the video from a single image conditioned on a driving video. Monkey-Net is a self from video and estimates the optical flow from the extracted keypoints. The input of this model (source image and driving video) is similar to that used in our approach.

III. PROPOSED APPROACH
Our objective is to generate a video in which the source image is animated according to a specific driving video. The flow of cloud is overly complex as it contains various motion elements (e.g., direction, intensity, and deformation). To accurately represent these elements, we use dense optical flow, and condition the video with this optical flow. Our model learns a pair of inputs consisting of the source image and optical flow of the driving video. Consequently, the network can generate a video, which is animated according to the

- We show that optical flow conditioning improves the quality of the generated video in the video reconstruction task.

II. RELATED WORK
In this section, we focus our discussions on recent learning-based video generation. In addition, we discuss the related topic on image animation.

A. DEEP VIDEO GENERATION
In recent years, deep neural networks that learn spatio-temporal feature have been widely used in video generation. Vondrick et al. proposed VGAN [4] in which the generator consists of three-dimensional (3D) convolutions to learn spatio-temporal features. V-GAN includes two streams of networks: one is used for generating static background and the other for generating dynamic foreground. Saito et al. proposed TGAN [10], which generates a video frame by frame. However, the quality of the generated video by these methods is quite low. Tulyakov et al. proposed MoCoGAN [11], which is based on a recurrent architecture. It divides latent space into content and motion, thereby improving the visual and motion quality. Clark et al. proposed DVD-GAN [8], which facilitates high-quality video generation by training a large dataset. They used two discriminators to reduce computational complexity. Overall, GAN method is primarily used for video generation. Although the quality of the generated animation has improved, conventional animation generates unnatural movements of the object. To address this issue, several studies have focused on generating moving images by conditioning on the physical structure. Depth conditional video GAN (DCVGAN) [3] generates the output

![FIGURE 1. Examples of video generated based on the proposed method. Given a source image and driving video, a video is generated by animating the source image according to the driving video. The generated video consists of 32 frames of 128 × 128 image. This figure shows the 1st, 8th, 15th, 22nd, and 29th frames.](image-url)
FIGURE 2. Proposed training architecture. The generator uses source image and driving video as inputs for generating the final video.

optical flow of the driving video. This architecture is inspired by pix2pix [17].

During training, the optical flow is extracted from a video. Subsequently, the optical flow and the first frame of the video is input to the generator. The generator outputs a video conditioned by the extracted optical flow. The discriminator receives the optical flow and ground truth video or the optical flow and generated video. This network is trained by optimizing the adversarial and reconstruction losses.

A. OPTICAL FLOW EXTRACTION

Gunnar-Farneback method [18], which is an algorithm to estimate the optical flow, is used to extract the optical flow in our method.

We also tried FlowNet2 [19] as an alternative deep learning method. However, the quality of the generated video was worse than the one obtained by using the Farneback method. We generated the video using these optical flow and evaluated it during the reconstruction task. The results are shown in Tab. 1. The videos generated by using the Farneback method optical flow and FlowNet2 optical flow are shown in Fig. 3. The video generated by using the optical flow of the Farneback method is better at reproducing details. This may be due to the fact that the optical flow of FlowNet2 is generally smooth and does not represent details.

B. VIDEO GENERATOR

The architecture of the video generator is illustrated in Fig. 2. Our generator is nearly similar to the stage I generator of MDCGAN [12]. The difference is in the number of channels in the first layer. The architecture of the generator is similar to that of U-net [20], which consists of 3D convolutions. Each layer has skip-connections, which prevent the vanishing gradient.

The training procedure is explained as follows. Firstly, we obtain a single RGB image \( x \in \mathbb{R}^{3 \times H \times W} \) from the first frame of the video \( X \in \mathbb{R}^{T \times 3 \times H \times W} \) and generate \( X' \in \mathbb{R}^{T \times 3 \times H \times W} \) by duplicating \( x \in \mathbb{R}^{3 \times H \times W} \) \( T \) times. \( X' \) represents a video composed of the duplicated first frame of the video \( X \). Secondly, we extract the optical flow \( X_f \in \mathbb{R}^{(T-1) \times 2 \times H \times W} \) from \( X \). Finally, we concatenate \( X' \) and \( X_f \) along the channel axis to obtain \( X_{\text{input}} \in \mathbb{R}^{T \times 5 \times H \times W} \) as the input of the generator. The generator encodes and decodes \( X_{\text{input}} \) by 3D convolutions. The generator outputs the video \( Y \in \mathbb{R}^{T \times 3 \times H \times W} \).

C. DISCRIMINATOR

The discriminator has a decoder architecture, which consists of 3D convolution, leaky Relu, batch normalization, and

| Optical flow Extractor | PSNR ↑ | SSIM ↑ | Perceptual dissimilarity ↓ |
|------------------------|--------|--------|---------------------------|
| Farneback              | 22.5   | 0.690  | 0.135                     |
| FlowNet2               | 22.2   | 0.691  | 0.169                     |

VOLUME 9, 2021
The discriminator architecture is shown in Fig. 2. Two types of data are used as the input for the discriminator. The first one is a pair of $X \in \mathbb{R}^{T \times 3 \times H \times W}$ and $X_f \in \mathbb{R}^{(T-1) \times 2 \times H \times W}$, and the second one is a pair of $Y \in \mathbb{R}^{T \times 3 \times H \times W}$ and $X_f \in \mathbb{R}^{(T-1) \times 2 \times H \times W}$. These pairs are concatenated along the channel axis, and the discriminator outputs the probability of the input data being real for adversarial loss calculation.

**D. NETWORK TRAINING**

While training our model, we optimize the adversarial loss as

$$\mathcal{L}_{adv} = \mathbb{E}[\log D(X, X_f)] + \mathbb{E}[\log(1 - D(G(X', X_f), X_f))]$$  \hspace{1cm} (1)

Previous research on GAN has shown that the use of $L_1$ distance as content loss improves the quality of the result and provides sharper outputs than that obtained using $L_2$ distance [17]. These losses facilitate regularization during the training. Accordingly, we also use $L_1$ distance as

$$\mathcal{L}_{con} = ||X - G(X', X_f)||_1$$  \hspace{1cm} (2)

Thus, the total loss for model training is given by

$$\mathcal{L}_{total} = \mathcal{L}_{adv} + \mathcal{L}_{con}$$  \hspace{1cm} (3)

**E. GENERATION PROCEDURE**

During the testing, we input the source image $x_s \in \mathbb{R}^{3 \times H \times W}$ and optical flow of the driving video $X_f \in \mathbb{R}^{(T-1) \times 3 \times H \times W}$ to the generator. First, $X_s$ is generated by duplicating $x_s$ $T$ times. Second, the optical flow $X_f \in \mathbb{R}^{(T-1) \times 3 \times H \times W}$ is extracted from driving video $X_d \in \mathbb{R}^{T \times 3 \times H \times W}$. These tensors are used as the inputs for the generator for generating a video $Y$, which is the animation of the source image conditioned by the driving video.

$$Y = G(X_s, X_f)$$  \hspace{1cm} (4)

**F. GENERATION RESULT**

The generated videos are shown in Fig. 1. The generated videos not only reflect the direction of the cloud flow but also the intensity and the deformation of the cloud formation, which are directly derived from the driving video. However, the color information of the driving video is not reflected in the generated video because the generator only receives the optical flow of the driving video.

**IV. EXPERIMENTS**

In this section, we compare our method with previous work. In addition, we also show the results of applying it to a painting. Our source code is publicly available and you can see a demo of the generated video at GitHub\(^1\).

**A. DATASET**

We used the time-lapse video dataset created by Xiong *et al.* [12] for the experiments. This dataset consists

\(^1\)https://github.com/kult0922/Animating-Cloud-Images-with-Flow-Style-Transfer
FIGURE 4. Comparison between the videos generated using the proposed method and Monkey-Net [1]. The 1st row shows the optical flow extracted using the Farneback method. The videos in the 2nd and 3rd rows are based on the proposed method and Monkey-Net, respectively. This figure shows the 1st, 5th, 9th, 13th, 17th, 21st, 25th, and 29th frames.

of landscape videos containing sky, which are recorded with a fixed-point camera. This dataset includes various the sky patterns (sunny, rainy, cloudy, sunset, and night sky). There are over 5,000 time-lapse videos on YouTube. A video consists of RGB images and its dimension is $3 \times 360 \times 640$. This video data is decomposed into several short video clips by sequentially combining continuous 32 frames as a clip. There are 35,392 training video clips and 2,815 testing video clips in the dataset.

B. IMPLEMENTATION DETAIL

The size of the dataset video is $3 \times 360 \times 640$. Our model only accepts square videos, which is realized by resizing the videos to a size of $128 \times 128$. The generated video frame $T$ is set as 32. Adam is used as the optimizer with $\beta = 0.5$ and the momentum $= 0.9$. The learning rate is 0.0002, and it remains constant throughout the training procedure. The content loss is calculated by obtaining the average loss per pixel. We have used calcOpticalFlowFarneback function of openCV, whose parameters are (pyrScale = 0.5, levels = 3, winsize = 15, iterations = 5, polyN = 5, polySigma = 1.2).

The video and optical flow are normalized in the range of $[-1, 1]$. The optical flow is normalized by dividing it with the size of the frame. The number of frames in the optical flow $X_f \in \mathbb{R}^{(T-1)\times2\times H\times W}$ and replicated video $X' \in \mathbb{R}^{T\times3\times H\times W}$ are different because a frame in the optical flow is calculated by using the difference between two consecutive frames of the driving video. This issue, i.e., the difference in the number of frames must be resolved when the replicated video and optical flow are being concatenated. To this end, we append an empty frame $X_0 \in \mathbb{R}^{2\times H\times W}$ in which all the elements are 0 to optical flow.
C. COMPARISON WITH PREVIOUS WORKS

In this section, we compare the proposed method with previously reported methods based on two tasks: image animation and video reconstruction.

1) IMAGE ANIMATION

The source images and driving videos are selected from the test data of the time-lapse video dataset. These data are used in image animation for generating videos based on the proposed method and Monkey-Net [1]. The examples of generated videos are shown in Fig. 4. Our method generates a natural video, where the source images are animated according to the driving videos. However, the videos generated by Monkey-Net are unnatural. The cloud shape is broken, and the motion pattern is very different from the driving video pattern. For evaluation of image animation, we conduct a user study using Amazon Mechanical Turk, encompassing 40 subjects. The driving video and generated video sampled from our method and Monkey-Net are shown to the subjects with unlimited watching time. Then the subjects are asked, “Which video is closer to the driving video in terms of motion?” We sample 50 videos for user study from the test set. The ratios of votes each method acquired are given in Tab 2. Monkey-Net extracts keypoints from videos. As the cloud image does not contain keypoints, this framework fails to generate natural videos. This confirms that our method is more effective than Monkey-Net in the case of keypoint-less videos.

| Ours | Monkey-Net |
|------|-------------|
| 82.9% | 17.1%       |

2) VIDEO RECONSTRUCTION

For video reconstruction, the first frame of the driving video is set as the source image. The generated videos are shown in Fig. 5. The generated video is evaluated by the peak signal-to-noise ratio (PSNR), structural similarity (SSIM), and perceptual dissimilarity between the generated video and the ground truth. Perceptual dissimilarity [21] is an evaluation metric with a sensitivity that is close to that of human perception. It is the distance between the high-level feature representations. These features are extracted from a well-trained convolutional neural network (CNN) model. Our goal is to animate a static image according to driving video. Monkey-Net is the only work with the same problem set. Therefore, we compare with Monkey-Net. Also, we compare with DTV-Net [22] and MD-GAN [12]. DTV-Net uses optical flow to extract a motion vector. Here, we compared our method with these methods using the time-lapse video dataset [12]. The results are shown in Tab 3. It may be noted MD-GAN cannot utilize the driving video, therefore we omitted it in our experiment. Our method surpasses other methods in all the evaluation metrics. The quality of videos generated by our method is much higher than that generated by other...
methods. And, our model embeds optical flow information into the generated video. This shows that conditioning on the optical flow is effective for video reconstruction. Furthermore, these results validate the efficacy of our method for the reconstruction of keypoint-less videos.

3) OPTICAL FLOW EVALUATION
To see if the motion is transferred to a still image, we evaluate the optical flow of input video and output video. The optical flow is evaluated by the metrics Average Endpoint Error (AEE) and Average Angle Error (AAE). The results are shown in Tab 4. Each optical flow was extracted using the Farneback method, except for DTV-Net. For DTV-Net, we use optical flow extracted by ARFlow [23], following the original method [22].

D. APPLICATION TO PAINTINGS
Here, we applied our method to paintings. A painting is set as the source image, and a test video of the dataset is set as the driving video. Note that these paintings are not included in the training data. The generated videos are shown in Fig. 6, which confirm that the paintings are animated according to the driving video. This shows that our method can be applied to painting animation. However, paintings in which the cloud appearance is vastly different than the real condition cannot be animated because the driving video shows a real cloud scene instead of a painted one.

E. DISCUSSION
The proposed image animation method generates a video according to a driving video and facilitates keypoint-less image animation. However, our method has some limitations.
First, our method fails to generate a video when the structures of the source image and driving video are very different. In particular, the cloud position in the source image and the driving video must be the same for generating a natural video. Some examples of this failure are shown in Fig. 7. Therefore, for using our model, one needs to pay extra attention to choosing the source image and driving video with similar structures.

![Source image](image1.png) ![Driving video](image2.png) ![Generated video](image3.png)

**FIGURE 7.** Failed examples of the proposed model. We sample a frame from the generated videos for each example. The red circle shows a distorted part of the image. If the structures of the source image and the driving video are very different (e.g., there is no cloud at the top right of the source image, but there is a cloud at the top right of the driving video), the generated video appears to be unnatural.

Second, the quality of videos generated by our method depends on the accuracy of the Farneback method for extracting optical flow. Although the accuracy of the Farneback method is high, it is not perfect. If this accuracy decreases, the quality of the generated video also decreases. However, as the process of extracting the optical flow is independent of the video generation process, we have the flexibility of replacing the optical-flow extraction method with more effective ones.

V. CONCLUSION

We proposed an effective image animation method based on the use of a driving video. By conditioning on the optical flow, the generated video could be animated according to a driving video. The dense optical flow contained various types of motion information (direction, intensity, and deformation). Consequently, the generated video reproduced the motion of the driving video. Particularly, we suggest the effectiveness of our method in the case of a cloud video without keypoints and that conditioning with optical flow enhances the video quality during the video reconstruction task. The proposed method requires driving videos to animate static images by implicitly learning fluid dynamics from the videos. The applications of such visual learning are not only image animation but also other potential applications such as video generation and forecasting of fluid dynamics in videos.

ACKNOWLEDGMENT

The authors are grateful to Calvin Janitra Halim for useful discussion and advice.

REFERENCES

[1] A. Siarohin, S. Lathuiliere, S. Tulyakov, E. Ricci, and N. Sebe, “Animating arbitrary objects via deep motion transfer,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 2372–2381.

[2] C. Chan, S. Ginosar, T. Zhou, and A. Efros, “Everybody dance now,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 5933–5942.

[3] Y. Nakahira and K. Kawamoto, “DCVGAN: Depth conditional video generation,” in Proc. IEEE Int. Conf. Image Process. (ICIP), Sep. 2019, pp. 749–753.

[4] C. Vondrlic, H. Pirsivash, and A. Torralba, “Generating videos with scene dynamics,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 613–621.

[5] K. Ohnishi, S. Yamamoto, Y. Ushiku, and T. Harada, “Hierarchical video generation from orthogonal information: Optical flow and texture,” in Proc. Assoc. Adv. Artif. Intell. (AAAI), 2018.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, Courville, and Y. Bengio, “Generative adversarial nets,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 2672–2680.

[7] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” 2013, arXiv:1312.6114. [Online]. Available: http://arxiv.org/abs/1312.6114

[8] A. Clark, J. Donahue, and K. Simonyan, “Adversarial video generation on complex datasets,” 2019, arXiv:1907.06571. [Online]. Available: http://arxiv.org/abs/1907.06571

[9] O. Wiles, A. Sophia Keepke, and A. Zisserman, “X2face: A network for controlling face generation using images, audio, and pose codes,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 670–686.

[10] M. Saito, E. Matsumoto, and S. Saito, “Temporal generative adversarial nets with singular value clipping,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2830–2839.

[11] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, “MoCoGAN: Decomposing motion and content for video generation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 1526–1535.

[12] W. Xiong, W. Luo, L. Ma, W. Liu, and J. Luo, “Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 2364–2373.

[13] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, “StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 5907–5915.

[14] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, “Flow-grounded spatial-temporal video prediction from still images,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 600–615.

[15] Y. Endo, Y. Kanamori, and S. Kuriyama, “Animating landscape: Self-supervised learning of decoupled motion and appearance for single-image video synthesis,” ACM Trans. Graph., vol. 38, no. 6, pp. 175:1–175:19, 2019.

[16] C.-C. Cheng, H.-Y. Chen, and W.-C. Chiu, “Time flies: Animating a still image with time-lapse video as reference,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 5641–5650.

[17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 1125–1134.

[18] G. Farnebäck, “Two-frame motion estimation based on polynomial expansion,” in Proc. 13th Scand. Conf. Image Anal., in Lecture Notes in Computer Science, vol. 2749, Gothenburg, Sweden, Jun./Jul. 2003, pp. 363–370.

[19] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “FlowNet 2.0: Evolution of optical flow estimation with deep networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 2462–2470.
[20] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in Proc. Int. Conf. Med. Image Comput. Assist. Intervent. Cham, Switzerland: Springer, 2015, pp. 234–241.

[21] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 586–595.

[22] J. Zhang, C. Xu, L. Liu, M. Wang, X. Wu, Y. Liu, and Y. Jiang, “DTVNet: Dynamic time-lapse video generation via single still image,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2020, pp. 300–315.

[23] L. Liu, J. Zhang, R. He, Y. Liu, Y. Wang, Y. Tai, D. Luo, C. Wang, J. Li, and F. Huang, “Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 6489–6498.

KAZUMA KURISAKI was born in Haruno, Shizuoka, Japan, in 1996. He received the B.E. degree in information engineering from Chiba University, in 2019, where he is currently pursuing the M.E. degree in information science. His research interests include deep learning and computer vision.

KAZUHIKO KAWAMOTO (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in information science from Chiba University, Japan, in 1997, 1999, and 2002, respectively. From 2002 to 2005, he was an Assistant Professor with the Tokyo Institute of Technology, Japan. From 2005 to 2009, he was an Associate Professor with the Kyushu Institute of Technology, Japan. He is currently a Professor with the Graduate School of Engineering, Chiba University. His research interests include computer vision, pattern recognition, and statistical signal processing.

KAZUHIKO KAWAMOTO (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in information science from Chiba University, Japan, in 1997, 1999, and 2002, respectively. From 2002 to 2005, he was an Assistant Professor with the Tokyo Institute of Technology, Japan. From 2005 to 2009, he was an Associate Professor with the Kyushu Institute of Technology, Japan. He is currently a Professor with the Graduate School of Engineering, Chiba University. His research interests include computer vision, pattern recognition, and statistical signal processing.