Castle in the Sky: Dynamic Sky Replacement and Harmonization in Videos

Zhengxia Zou*, Rui Zhao, Tianyang Shi, Shuang Qiu, and Zhenwei Shi, *Member, IEEE*

***Abstract*—We propose a vision-based framework for dynamic sky replacement and harmonization in videos. Different from previous sky editing methods that either focus on static photos or require real-time pose signal from the camera's inertial measurement units, our method is purely vision-based, without any requirements on the capturing devices, and can be well applied to either online or offline processing scenarios. Our method runs in real-time and is free of manual interactions. We decompose the video sky replacement into several proxy tasks, including motion estimation, sky matting, and image blending. We derive the motion equation of an object at infinity on the image plane under the camera's motion, and propose "flow propagation", a novel method for robust motion estimation. We also propose a coarse-to-fine sky matting network to predict accurate sky matte and design image blending to improve the harmonization. Experiments are conducted on videos diversely captured in the wild and show high fidelity and good generalization capability of our framework in both visual quality and lighting/motion dynamics. We also introduce a new method for content-aware image augmentation and proved that this method is beneficial to visual perception in autonomous driving scenarios. Our code and animated results are available at https://github.com/jiupinjia/SkyAR.**

***Index Terms*—Augmented reality, video sky replacement, motion estimation, data augmentation.**

I. Introduction

SKY is a key component in outdoor photography. Photos and videos captured in the wild often suffer from an over-exposed or plain-looking sky under uncontrollable weather and lighting conditions. Thanks to the recent advances in computer vision, we see automatic sky editing/optimization became an emerged research topic recently [1–4]. Some recent photo editors like MeituPic and Photoshop have featured sky replacement toolboxes where users can easily swap out the sky in their photos with only a few clicks.

Despite the recent effort in image-based sky replacement, for video applications, automatic sky editing is still rarely studied but has huge potential in short video and film applications. In film production, manually replacing the sky in the video is laborious, time-consuming and even requires professional post-film skills. The editing typically involves frame-by-frame blue screen matting and background motion capturing. Users may spend hours after considerable practice, even with the help of professional software. Some recent mobile Apps featured sky augmented reality. For example, the stargazing App StarWalk2 can help users track stars, planets, constellations, and other celestial objects with interactive augmented reality. However, These applications are highly dependent on the hardware platform. To calibrate the real camera with the virtual one, the algorithms typically require real-time pose signals from the Inertial Measurement Units (IMU) available when capturing the video [5], and thus cannot be directly applied to offline application scenarios.

In this paper, we explore an interesting problem that whether the sky augmentation in videos can be realized in a purely vision-based manner, and propose a new solution for this task. Different from the previous methods, the proposed method does not rely on specific capturing devices and is suitable for both online augmented reality and offline video editing. There are three basic components in our method:

- **A motion estimator.** To render a virtual sky background in videos, the virtual camera needs to be synchronized with the motion of a real one. We, therefore, propose a "flow propagation" network for robust motion estimation of sky background regions. The training of the network does not require any manual annotation on the videos.
- **A sky matting network.** Different from previous methods that consider this process as a binary pixel classification (foreground v.s. sky) problem, we design a coarse-to-fine prediction pipeline that produces soft sky matte with a more accurate detection result and so a more visually pleasing blending effect.
- **A skybox.** Given the predicted sky matte and the motion parameters, the skybox aligns a 360 degree sky template with the foreground and blends them together. The skybox also applies relighting and recoloring to make the blending result realistic.

We evaluate the proposed framework on outdoor videos diverse captured by both dash cameras and handheld smartphones. As shown in Fig. 1, our method can generate high fidelity and visually pleasing sky effects with nice lighting/motion dynamics. Also, the proposed framework can be applied to weather and lighting synthesis and shows better fidelity than previous methods based on generative adversarial networks.

The contribution of this paper is summarized as follows:

The work was supported by the National Natural Science Foundation of China under the Grant 62125102, and the Fundamental Research Funds for the Central Universities (Corresponding author: Zhengxia Zou (zhengxiazou@buaa.edu.cn).)

Zhengxia Zou is with the Department of Guidance, Navigation and Control, School of Astronautics, Beihang University, Beijing 100191, China.

Rui Zhao and Tianyang Shi are with the Fuxi AI Lab, Netease, Hangzhou, Zhejiang, 310052, China.

Shuang Qiu is with the Booth School of Business, University of Chicago, Chicago, IL 60637, USA.

Z. Shi is with the Image Processing Center, School of Astronautics, Beihang University, Beijing 100191, China, and with the Beijing Key Laboratory of Digital Media, Beihang University, Beijing 100191, China, and also with the State Key Laboratory of Virtual Reality Technology and Systems, School of Astronautics, Beihang University, Beijing 100191, China.

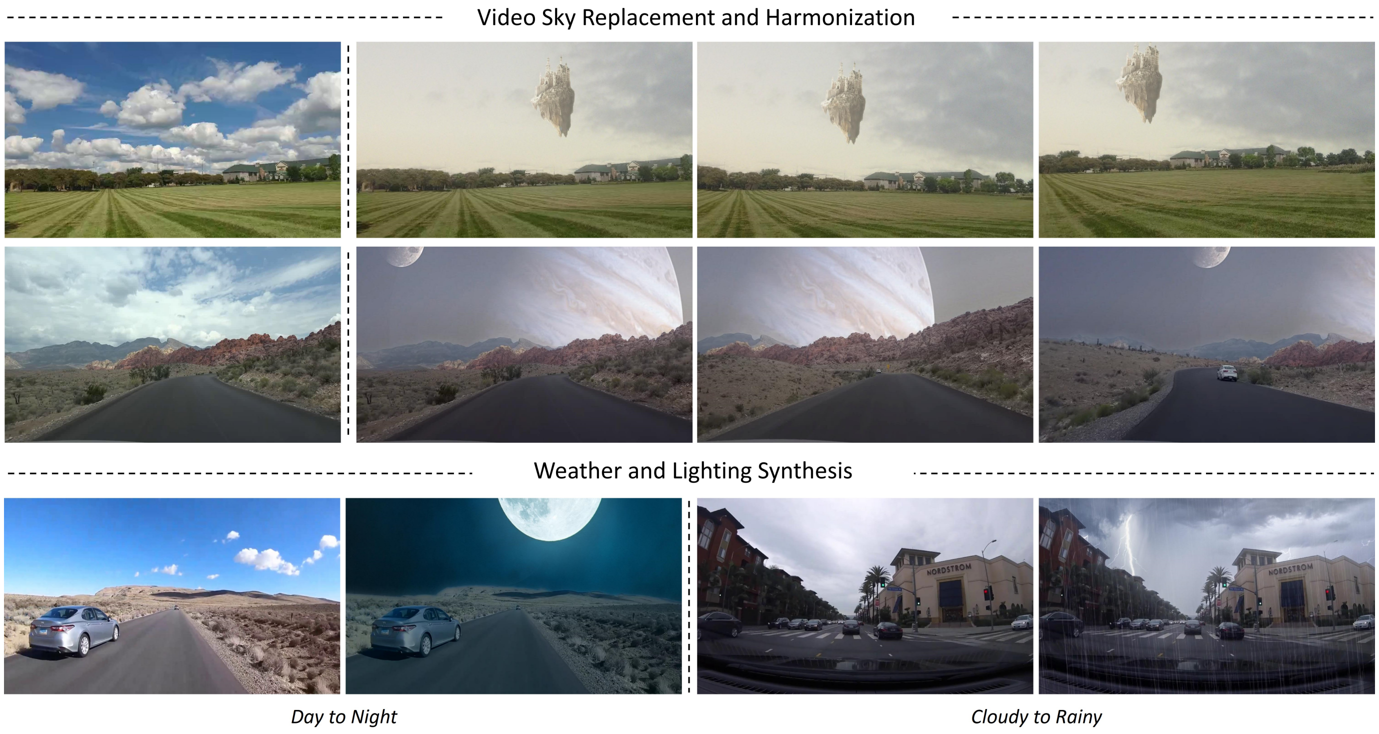
Video Sky Replacement and Harmonization
Weather and Lighting Synthesis
NORDSTROM
Day to Night
Cloudy to Rainy

Figure 1. We propose a vision-based method for generating videos with controllable sky backgrounds and realistic weather/lighting conditions. First and second row: rendered sky videos - flying castle and Jupiter sky (leftmost shows a frame from input video and the rest are the output frames). Third row: weather and lighting synthesis (day to night, cloudy to rainy). We suggest the reviewers/readers visit our project page and view our animated results..

- We propose a new framework for dynamic sky replacement and harmonization in videos. Previous approaches on this topic either focus on static images or require real-time camera pose signals from IMU. As a comparison, our method is purely vision-based and thus can be applied to both online and offline application scenarios.
- We propose "flow propagation", a new method for robust background motion estimation, which propagates the motion flow from foreground objects to the background with deep neural networks. The networks can be effectively trained on videos without using external data annotations.
- We derive the motion equation of an infinite-far object on the image plane under the camera's motion and prove the motion follows a rigid affine transformation. The conclusion provides theoretical support for the proposed motion estimation method.
- Our method can be also used as a new way of data augmentation for visual perception. We refer to our method as "content-aware" augmentation. We show that our method is more effective than conventional content-agnostic data augmentation methods like random flip and random color jittering.

II. Related Work

Sky replacement and editing are common techniques in photo processing and film post-production. For static photos, some automatic methods for sky segmentation and replacement have been proposed recently [1–3].

SkyFinder [1] proposed by Tao *et al*. is one of the pioneer methods that study sky replacement. This method first trains a random-forest-based sky pixel detector and then applies graph cut segmentation [6] to produce refined binary sky masks. The authors further conducted attribute-based sky image retrieval and controllable sky replacement. Tsai *et al*. [2] later focus on improving sky segmentation by using deep CNNs and conditional random field. Apart from the sky segmentation, Rawat *et al*. [3] studied the matching problem between the foreground query and background images using handcrafted color features and CNN features. Mihail *et al*. [7] contribute a dataset for sky segmentation which consists of 100k images from 53 stationary surveillance cameras. The images in this dataset have their binary ground truth masks for sky regions. This dataset provides a benchmark for subsequent studies on sky segmentation [5, 8]. Very recently, Liba *et al*. [4] propose a method for sky optimization in low-light photography, which also involves sky segmentation. They also introduce a refined sky image dataset built based on a subset of the AED20k dataset [9], which contains 10,072 images diversely captured in the wild with their high-quality soft sky mattes.

For video sky enhancement, Tran *et al*. proposed a method named "Fakeye" recently which renders virtual sky in real-time on smartphones [5]. However, their method requires pose signals from IMU, and thus cannot directly be applied to offline applications. "Clear Skies Ahead" *et al*. [10] proposed by Halperin *et al*. is another recent method that focuses on video sky augmentation. Note that although this method is also a purely vision-based approach, our method differs from theirs

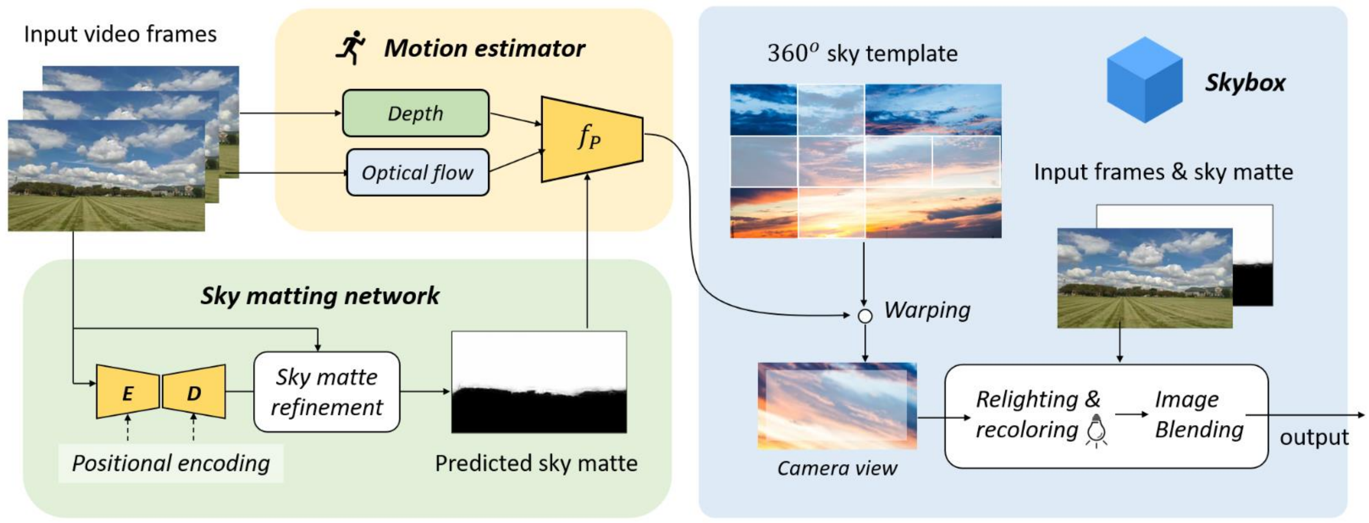

Figure 2. An overview of our method. Our method consists of a motion estimator for background motion estimation, a sky matting network for sky matte prediction, and a skybox for image blending.

in two aspects. First, we consider the sky pixel segmentation as a soft image matting problem that produces more natural blending results. As a comparison, their method considers the sky segmentation as binary pixel classification which may cause blending artifacts at the segmentation junctions. Second, Clear Skies Ahead perform camera calibration based on tracked points between frames and further assumes the motion undergoes by the camera is only rotational. In our method, instead of estimating the motion of the camera, we propose flow propagation and estimate the background motion on the image plane directly, which can deal with both rotation and translation of the camera in a unified framework.

III. Methodology

Fig. 2 shows an overview of the proposed framework. Our framework consists of a motion estimator for background motion estimation, a sky matting network for sky matte prediction, and a skybox for image blending. In below, we will first analyze the projected motion of infinite-far objects and then introduce each module in detail.

A. Projected Motion Equation

In this subsection, we explain that the motion of the sky background pixels under goes with the camera motion can be approximated by a rigid affine transformation (translation and rotation only) in the image plane. To prove this, we assume the camera follows a standard pinhole model with zero skew and a single focal length. We then project 3D points from the world coordinate to the image according to the camera intrinsic and extrinsic parameters. By assuming the sky pixels are located at infinity far away, we derive an approximated equation of the projected motion and finally prove that the pixel motion between different time steps can be described by an rigid affine model.

Proposition. *Suppose* $\mathbf{p}_w = [x_w, y_w, z_w]^\top \in \mathbb{R}^3$ *represents a 3D point in the world coordinate;* $\mathbf{R}^{(i)} \in \mathbb{R}^{3\times3}$ *and* $\mathbf{t}^{(i)} \in \mathbb{R}^3$ *are the rotation and translation parameters of the camera at time step* i*. Suppose the camera follows a pinhole model with zero skew and a single focal length. If* $\mathbf{p}_w$ *is located at infinity far away from the camera and always has a small angle of incoming ray, then the motion of the projected points on the image plane between time step* 0 *and time step* i *can be described by an rigid affine transformation* $\boldsymbol{p}^{(i)} = \mathbf{M}^{(i)}\boldsymbol{p}^{(0)}$*, where* $\boldsymbol{p}^{(0)} \in \mathbb{R}^3$ *and* $\boldsymbol{p}^{(i)} \in \mathbb{R}^3$ *are the homogeneous coordinate of the points on the image plane,* $\mathbf{M}^{(i)} \in \mathbb{R}^{3\times3}$ *is an rigid affine transformation matrix (rotation and translation only).*

Proof. Suppose at time step 0, the world coordinate and the camera coordinate have the same original point and the same rotation. By transforming the point $\mathbf{p}_w = [x_w, y_w, z_w]^\top$ from world coordinate to the camera coordinate, the point $\mathbf{p}_w$ in the camera coordinate at time step i can be represented as

$$\begin{bmatrix} x_c^{(i)} \\ y_c^{(i)} \\ z_c^{(i)} \end{bmatrix} = \mathbf{R}^{(i)} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \mathbf{t}^{(i)}. \tag{1}$$

According to the standard pinhole camera model, the projected point in the image coordinate can be represented as

$$\begin{bmatrix} x_p^{(i)} \\ y_p^{(i)} \\ z_p^{(i)} \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_c^{(i)} \\ y_c^{(i)} \\ z_c^{(i)} \end{bmatrix} = \mathbf{K}\mathbf{R}^{(i)} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \mathbf{K}\mathbf{t}^{(i)}, \tag{2}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the intrinsic parameter matrix of the camera and $[x_p^{(i)}, y_p^{(i)}, z_p^{(i)}]^\top$ is the homogeneous coordinate representation of the point on the image plane. The normalized coordinate can be represented as

$$[u^{(i)}, v^{(i)}]^\top = [x_p^{(i)}/z_p^{(i)},\ \ y_p^{(i)}/z_p^{(i)}]^\top, \tag{3}$$

where $u^{(i)}$ and $v^{(i)}$ are the pixel row and column index of the image at time step i.

Since the camera has zero skew factor and a single focal length, the intrinsic matrix can be written as:

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

where f is the pixel focal length; c_x and c_y are the pixel offsets of the principal point.

At time step i, given the the camera's pitch angle α, yaw angle β and roll angle γ, the rotation matrix of the camera can be written as follows:

$$\mathbf{R}^{(i)} = \mathbf{R}_x(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma), \tag{5}$$

where

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}, \tag{6}$$

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, \tag{7}$$

$$\mathbf{R}_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

Since we assume the object at infinite-far always has a small angle of the incoming ray, the rotation matrix $\mathbf{R}^{(i)}$ can be approximated with two small components on $\alpha \approx 0$ and $\beta \approx 0$. We do not apply any constraints on the roll angle γ. The matrix $\mathbf{R}^{(i)}$ thus can be approximated as follows:

$$\mathbf{R}^{(i)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -\alpha \\ 0 & \alpha & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \beta \\ 0 & 1 & 0 \\ -\beta & 0 & 1 \end{bmatrix} \begin{bmatrix} c\gamma & -s\gamma & 0 \\ s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{9}$$

where we denote $\cos\gamma$ and $\sin\gamma$ as $c\gamma$ and $s\gamma$ for abbreviation. Moreover, $\sin\alpha$, $\sin\beta$, $\cos\alpha$, and $\cos\beta$ are approximated by their first-order Talyor expansion such that $\sin\alpha \approx \alpha$, $\sin\beta \approx \beta$, and $\cos\alpha \approx \cos\beta \approx 1$.

By substituting Eq. 4 and Eq. 9 into Eq. 3, the projected image coordinate of the point $\mathbf{p}_w$ at time i can be calculated as:

$$\begin{aligned} \boldsymbol{p}^{(i)} = \begin{bmatrix} u^{(i)} \\ v^{(i)} \\ 1 \end{bmatrix} &= \begin{bmatrix} x_p^{(i)}/z_p^{(i)} \\ y_p^{(i)}/z_p^{(i)} \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_c^{(i)}/z_c^{(i)} \\ y_c^{(i)}/z_c^{(i)} \\ 1 \end{bmatrix} \\ &= \mathbf{K}\mathbf{R}^{(i)} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} /z_c^{(i)} + \mathbf{K}\mathbf{t}^{(i)}/z_c^{(i)}. \end{aligned} \tag{10}$$

Since the point $\mathbf{p}_w$ are located at infinite far way from the camera, we have $z_c^{(i)} \to \infty$. By further expressing $\mathbf{p}_w$ as

$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = l \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix}, \quad l = \|\boldsymbol{p}_w\|_2, \tag{11}$$

where $\boldsymbol{\theta} = [\theta_x, \theta_y, \theta_z]^\top$ has a unit norm $\|\boldsymbol{\theta}\|_2 = 1$, Eq. 10 can be approximated as

$$\begin{aligned} \boldsymbol{p}^{(i)} &\approx \mathbf{K}\mathbf{R}^{(i)} \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} l/z_c^{(i)} \\ &\approx \begin{bmatrix} (\theta_x/\theta_z \cos\gamma - \theta_y/\theta_z \sin\gamma + \beta)f + c_x \\ (\theta_x/\theta_z \sin\gamma + \theta_y/\theta_z \cos\gamma - \alpha)f + c_y \\ 1 \end{bmatrix}. \end{aligned} \tag{12}$$

Since at time 0 we have

$$\boldsymbol{p}^{(0)} = \begin{bmatrix} u^{(0)} \\ v^{(0)} \\ 1 \end{bmatrix} \approx \begin{bmatrix} f\theta_x/\theta_z + c_x \\ f\theta_y/\theta_z + c_y \\ 1 \end{bmatrix}, \tag{13}$$

by comparing Eq. 13 and Eq. 10, we have

$$\mathbf{T}_c\boldsymbol{p}^{(i)} = \mathbf{T}_{uv}^{(i)}\mathbf{R}_{uv}^{(i)}\mathbf{T}_c\boldsymbol{p}^{(0)}, \tag{14}$$

where

$$\mathbf{R}_{uv}^{(i)} = \begin{bmatrix} c\gamma & -s\gamma & 0 \\ s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

is a rotation matrix and

$$\mathbf{T}_{uv}^{(i)} = \begin{bmatrix} 1 & 0 & \beta f \\ 0 & 1 & -\alpha f \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{T}_c = \begin{bmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{16}$$

are two translation matrices.

From Eq. 14 we have

$$\boldsymbol{p}^{(i)} = \mathbf{T}_c^{-1}\mathbf{T}_{uv}^{(i)}\mathbf{R}_{uv}^{(i)}\mathbf{T}_c\boldsymbol{p}^{(0)}. \tag{17}$$

Let $\mathbf{M}^{(i)} = \mathbf{T}_c^{-1}\mathbf{T}_{uv}^{(i)}\mathbf{R}_{uv}^{(i)}\mathbf{T}_c$, then we have

$$\boldsymbol{p}^{(i)} = \mathbf{M}^{(i)}\boldsymbol{p}^{(0)}. \tag{18}$$

Since $\mathbf{R}_{uv}^{(i)}$ is rotation only, $\mathbf{T}_{uv}^{(i)}$ and $\mathbf{T}_c^{-1}$ are translation only, therefore, $\mathbf{M}^{(i)}$ is a rigid affine transformation (rotation-translation only) matrix.

$\square$

B. Flow Propagation

To estimate the motion of the background, the most straightforward way is to estimate directly based on the optical flow from the sky region. However, when the sky has no textures or only covers a small area, there could be very few good features that can be used for reliable motion estimation. Therefore, we propose "Flow Propagation" that aims to recover background motion based on the optical flow of the foreground objects. Fig. 3 shows an overview of this method.

In the foreground, even with the same speed, objects at different depths would have different optical flow magnitudes. The smaller the depth is, the object is closer to the camera, and its optical flow magnitude tends to be larger. By contrast, the background is assumed at infinitely far away, and its optical flow is depth-independent. Since the flow propagation network aims to predict the motion of background given the motion of foreground, it's quite natural that we feed the depth map into the network to help it immune to the effect of depth on optical flow.

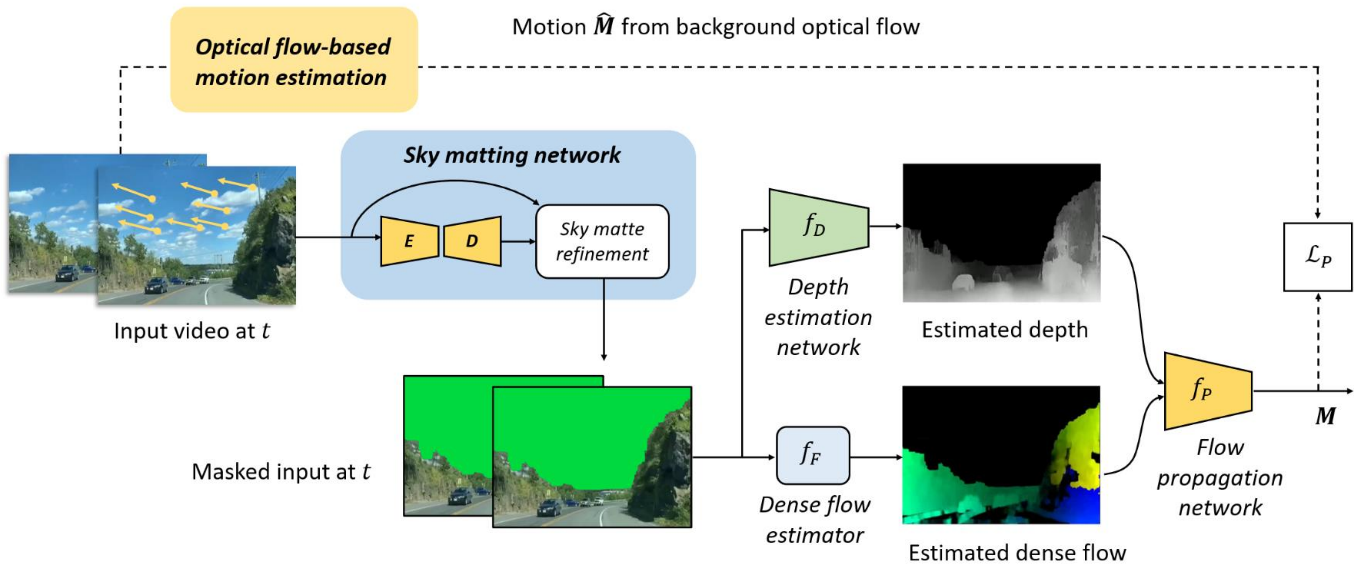

Figure 3. We propose a method called Flow Propagation to estimate the background motion from the foreground dense flow and the depth map. We train a flow propagation network f_P to minimize the difference between the predicted motion parameters $\mathbf{M}$ and those computed from the sky region. The dashed arrow lines show the data flow during the training, and those solid ones show the data flow in both training and inference.

Given two adjacent video frames, we first mask out the sky regions and then introduce a depth estimation network f_D and a dense flow estimator f_F to calculate the depth map and dense flow from the foreground region. Suppose $H^{(t)}$ denotes the input frame with its sky background masked out at the time t. The predicted depth map and dense flow thus can be written as $x_D^{(t)} = f_D(H^{(t)})$ and $x_F^{(t)} = f_F(H^{(t)}, H^{(t+1)})$. We then introduce another network f_P, which takes in both $x_D^{(t)}$ and $x_F^{(t)}$ (concatenated along the channel dimension), and predicts the motion matrix $\mathbf{M} \in \mathbb{R}^{3\times3}$ of the sky. We name f_P as flow propagation network.

We train the propagation network f_P without any external data annotation - the f_P is trained only on the video frames with rich sky textures. We use the motion $(dx, dy, d\alpha)$ that is directly computed from the the sky region as the reference for training. We train f_P to minimize the objective function below:

$$\mathcal{L}_P = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=0}^{T}\|f_P(x_D^{(t)}, x_F^{(t)}) - (\hat{dx}, \hat{dy}, \hat{d\alpha})\|_1 \quad (19)$$

where $\|\cdot\|_1$ is the ℓ_1 norm of a vector, i.e., the sum of absolute value of all elements. N and T are the number of video sequences and number of frames in each sequence used for training. We choose the real-time monocular depth estimation network [11] as our network f_D and choose the Gunner Farneback's estimator [12] as our dense flow estimator f_F. During the training, we fix the f_D and f_F and only optimize f_P to minimize (19).

To calculate the reference motion $(\hat{dx}, \hat{dy}, \hat{d\alpha})$, we first calculate the optical flow using the iterative Lucas-Kanade method [13] with pyramids. Then, and a set of sparse feature points in the sky region can be frame-by-frame tracked. For each pair of adjacent frames, given two sets of 2D feature points, we use the RANSAC-based robust affine estimation to compute the optimal 2D motion matrix with three degrees of freedom limited to translation and rotation.

Note that although we only train on the images with rich sky textures, during the inference phase, we do not make any assumptions on that. Sky with rich textures usually contains clouds to provide high quality feature points to track. In practice, we use the optical flow from the sky regions to compute the sky motion when there are enough feature points detected. Also, when there are no sky pixels, the key point matching algorithm will also fail track. Therefore, when there is no sky region detected or the number of matched points is limited, we use the proposed flow propagation to estimate the motion.

After obtaining the movement of each adjacent frame, the affine transformation matrix $\widetilde{\mathbf{M}}^{(t)}$ across between the initial frame and the t-th frame in the video can be written as the following matrix multiplication form:

$$\widetilde{\mathbf{M}}^{(t)} = \mathbf{M}^{(c)}(\mathbf{M}^{(t,t-1)}\mathbf{M}^{(t-1,t-2)}\dots\mathbf{M}^{(1,0)}), \quad (20)$$

where $\mathbf{M}^{(i-1,i)}$ $(i = 1, \dots t)$ represents the estimated motion parameters between frame $i-1$ and i. $\mathbf{M}^{(c)}$ are the center crop parameters of the skybox template, i.e., shift and scale, depending on the field of view set by the virtual camera.

C. Sky Matting

Image matting is an important research topic in image and video editing [14–16] that aims at separating foreground objects from the background. Image matting usually involves predicting a soft "matte" of objects of interest, which is highly related to our sky replacement task.

In our method, we train a deep CNN for sky matte prediction. Our sky matting networks consist of a segmentation encoder E, a mask prediction decoder D, and a soft refinement module. The encoder aims at learning intermediate feature representations of a down-sampled input image. The decoder is trained to predict a coarse sky matte. The refinement module takes in both the coarse sky matte and the original high-resolution input image and produces a refined sky matte. Note that since the sky region usually appears at the upper part of the image, we replace the standard convolution layers with coordinate convolution layers [17] at the encoder's input layer and all the decoder layers, where the normalized y-coordinate values are encoded. We show such a simple modification brings noticeable accuracy improvement on the prediction of sky matte.

Suppose I and I_l represent an input image with full resolution and its down-sampled copy. Our coarse segmentation network $f = \{E, D\}$ takes in the I_l and is trained to produce a sky matte $A_l = f(I_l)$ with the same resolution as I_l. We train the f to minimize the following objective function:

$$\mathcal{L}_f(I_l) = \frac{1}{N_I N_P} \sum_{I_l \in \mathcal{D}_l} \{\|f(I_l) - \hat{A}_l\|_2^2\}, \tag{21}$$

where $\|\cdot\|_2^2$ is the pixel-wise ℓ_2 distance between two images, and $\mathcal{D}_l$ is the dataset of down-sampled images. $\hat{A}_l$ is the ground truth sky alpha matte. N_I and N_P are the number of images and the number of pixels in each output image.

After we have the coarse sky matte, we up-sample the sky matte to the original input resolution and then use the guided filter [18] to recover the detailed structures. We use the full resolution image I as the guidance image. We denote $h(A_l)$ and A as the predicted full-resolution sky matte before and after the refinement, and A can be represented as follows:

$$A = f_{gf}(h(A_l), I, r, \epsilon), \tag{22}$$

where f_{gf} and h are the guided filtering and bilinear up-sampling operations. r and ϵ are the predefined radius and regularization coefficient of the guided filter. In the sky matte refinement stage, we set $r = 20$ and $\epsilon = 0.01$. Although recent image matting literature [16, 19] shows that using upsampling convolutional architecture and adversarial training may also produce fine-grained matting output, we choose guided filter mainly because it is simple and efficient.

D. Sky image blending

After motion estimation and sky matting, we render the virtual sky background by warping a 360 degree skybox template onto the camera view. The skybox can be simply generated by cropping and warping a high-resolution image according the Fig. 2. Suppose $I^{(t)}$, $A^{(t)}$, and $B^{(t)}$ are the video frame, the predicted sky alpha matte, and the warped sky template image at time t. Based on the image matting equation [20], we represent the newly composed frame $Y^{(t)}$ as the linear combination of $I^{(t)}$ and $B^{(t)}$, with $A^{(t)}$ as their pixel-wise combination weights:

$$Y^{(t)} = (1 - A^{(t)})I^{(t)} + A^{(t)}B^{(t)}, \tag{23}$$

where in $A^{(t)}$, a higher output pixel value means a higher probability the pixel belongs to the sky background.

Since the foreground image and the skybox template may have different color tone and intensity, directly performing the linear blending may produce unrealistic result. We thus apply recoloring and relighting to correct the colors and intensity of the foreground image before the linear combination (23):

$$\widehat{I}^{(t)} \longleftarrow I^{(t)} + \alpha(\mu_{B(A)}^{(t)} - \mu_{I(1-A)}^{(t)}), \tag{24a}$$

$$I^{(t)} \longleftarrow \beta(\widehat{I}^{(t)} + \mu_I^{(t)} - \widehat{\mu}_I^{(t)}), \tag{24b}$$

where $\mu_I^{(t)}$, $\widehat{\mu}_I^{(t)}$, are the mean color value of the image $I^{(t)}$ and image $\widehat{I}^{(t)}$. $\mu_{B(A)}^{(t)}$ and $\mu_{I(1-A)}^{(t)}$ are the mean color value of sky pixels in image $I^{(t)}$ and the mean color value of non-sky pixels in image $B^{(t)}$. α and β are pre-defined recoloring and relighting factors. In the above correction steps, the (24a) transfers the regional colortone from the background to the foreground image while the (24b) corrects the pixel intensity range of the re-colored foreground and make it compatible with the skybox background.

Table I
Details of our sky matting network.

Layer	Config	Filters	Reso
E_1	CoordConv-BN-Pool	64 / 2	1/4
E_2	3 × ResBlock	256 / 1	1/4
E_3	4 × ResBlock	512 / 2	1/8
E_4	6 × ResBlock	1024 / 2	1/16
E_5	3 × ResBlock	2048 / 2	1/32
D_1	CoordConv-UP + E_5	2048 / 1	1/16
D_2	CoordConv-UP + E_4	1024 / 1	1/8
D_3	CoordConv-UP + E_3	512 / 1	1/4
D_4	CoordConv-UP + E_2	256 / 1	1/2
D_5	CoordConv-UP	64 / 1	1/1
D_6	CoordConv	64 / 1	1/1

Table II
Details of our flow propagation network.

Layer	Config	Filters	Reso
E_1	CoordConv-BN-Pool	64 / 2	1/4
E_2	3 × ResBlock	256 / 1	1/4
E_3	4 × ResBlock	512 / 2	1/8
E_4	6 × ResBlock	1024 / 2	1/16
E_5	3 × ResBlock	2048 / 2	1/32
P_1	GF	/	1/1
P_2	FC	2048 / 256	1/1
P_3	FC	256 / 3	1/1

E. Implementation Details

1) Network architecture: We use the ResNet-50 [21] as the encoder of our sky matting networks (fully connected layers removed). The decoder part consists of five convolutional upsampling layers (coordinate conv + relu + bilinear upsampling) and a pixel-wise prediction layer (coordinate conv + sigmoid). We follow the configuration of the "UNet" [22] and add skip connections between the encoder and decoder layers with the same spatial size. We also use the ResNet-50 as the backbone of our flow propagation network f_P, where we replace the 1000-class prediction output with our motion parameter prediction output.

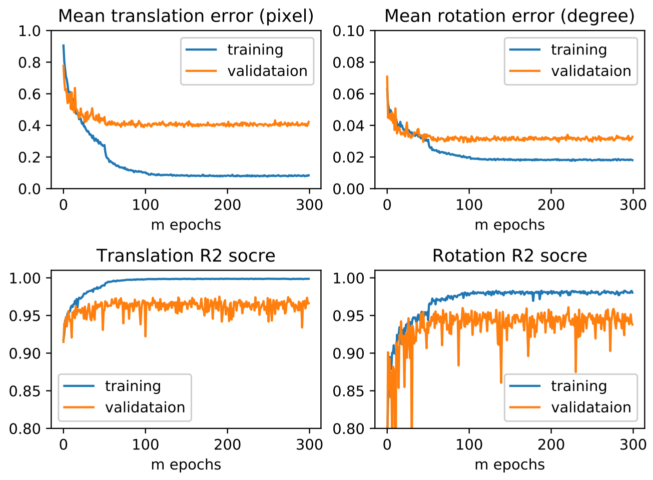

Figure 4. Training and validation error (mean absolute error) and R^2 score on training our flow proportion network f_P on outdoor videos. Both translation and rotation error converge nicely although no external supervision signal is used during the training.

Table I and Table II show the detailed configurations of our sky matting network and flow propagation network. In the two tables, "CoordConv" represents a coordinate convolution layer [17] followed by a ReLU layer. "BN", "UP", and "Pool" denote batch normalization, bilinear upsampling, and max-pooling layer respectively. "FC" represents a fully connected layer. "GF" represents a global pooling layer. In a c/s configured convolution layer, c denotes the number of filters, s denotes the filer's stride size. In a c/s configured fully connected layer, c denotes the number of input nodes, s denotes the number of output nodes. The "Reso" column gives the relative resolution of the feature maps compares to the size of the input image.

2) Dataset and training details: We train and evaluate our sky matting network on the dataset "ADE20K+DE+GF" introduced by Liba *et al.* [4]. This dataset is built based on the AED20K dataset [9]. There are 9,187 static images in the training set and 885 static images in the validation set.

We also build a video dataset for training our flow propagation network. The videos are collected from YouTube. There are 15 video clips in our training set, each with a length of 1-3 minutes (a total of 30 minutes long, 36,000 frames). There are 10 video clips in our testing set, each with a length of 5-30 seconds (a total of 3 minutes long, 3,600 frames). These videos are captured by using dash-cameras in urban and rural environments with clear sky background textures (with high-quality background feature points for tracking).

We train both our sky matte network and flow propagation network by using Adam optimizer [23]. We set the batch size to 12, the learning rate to 0.0001, and betas to (0.9, 0.999). We stop training after 200 epochs. We reduce the learning rate to its 1/10 every 50 epochs. When training the image matting network, we set the input size to 384×384. The input image size for training the flow propagation network is set to 480×845×3, where the first two channels are the x-y direction component of the dense optical flow and the last channel is the depth map. Fig. 4 shows the error and R^2 score between the predicted output of the flow propagation network and the ground truth. We finally achieves an average of 0.422 sub-pixel translation error and 0.0572 degrees of rotation error on our validation set. The validation R^2 score on translation is 0.966 and the validation R^2 score on rotation is 0.937.

Testing scenario	PI [27]	NIQE [28]
CycleGAN [26] (sunny-to-cloudy)	7.094	7.751
Ours (sunny-to-cloudy)	5.926	6.948
CycleGAN [26] (cloudy-to-sunny)	6.684	7.070
Ours (cloudy-to-sunny)	5.702	7.014

Table III
QUANTITATIVE EVALUATION ON THE IMAGE FIDELITY OF OUR METHOD AND CYCLEGAN UNDER DIFFERENT TESTING SCENARIOS. FOR BOTH PI AND NIQE, LOWER SCORES INDICATE BETTER.

IV. EXPERIMENTAL ANALYSIS

We evaluate our method on video sequences captured in the wild by using both handheld smartphones and dash-cameras. We also test our method on street scene images from the public dataset Cityscapes [24] and BDD100k [25].

A. Video Sky Replacement and Weather Simulation

Fig. 5 shows video sky replacement results by using our method. In each row, we show an input frame from the original video and the subsequent processing output at different time steps. The testing videos in the top-4 rows are downloaded from YouTube (video source 1 and 2) and the testing video in the last row is captured by ourselves with a handheld smartphone (Xiaomi Mi 8) in Ann Arbor, MI, USA. We generate dynamic sky effects of "sunset", "District-9 ship", "super moon", and "Galaxy" for the above video clips. Our method generates visually pleasing results with a high degree of realism. For more examples and animated demonstrations, please visit our project website.

Fig. 6 shows the results of our method for weather translation (sunny to cloudy, sunny to rainy, cloudy to sunny, cloudy to rainy). When we generate videos with rain, we add a dynamic rain layer (layer source) and a haze layer on top of the result by using screen blending [29]. We show with only minor modifications of the skybox template and the relighting factor, a visually realistic weather effect can be obtained. We also compare our method with CycleGAN [26], a well-known unpaired image-to-image translation method, which is built based on conditional generative adversarial networks. We train CycleGAN on the BDD100K dataset [25] which contains outdoor traffic scenes under different weather conditions. Fig. 7 shows the qualitative comparison results. In Table III, we give a quantitative comparison of the image fidelity of CycleGAN and our method under different weather translation scenarios. Note that since the CycleGAN does not consider temporal dynamics, we only test on static images. We randomly select 100 images from the validation set of BDD100K with corresponding weather conditions for each evaluation group. Two metrics Perception Index (PI) [27] and

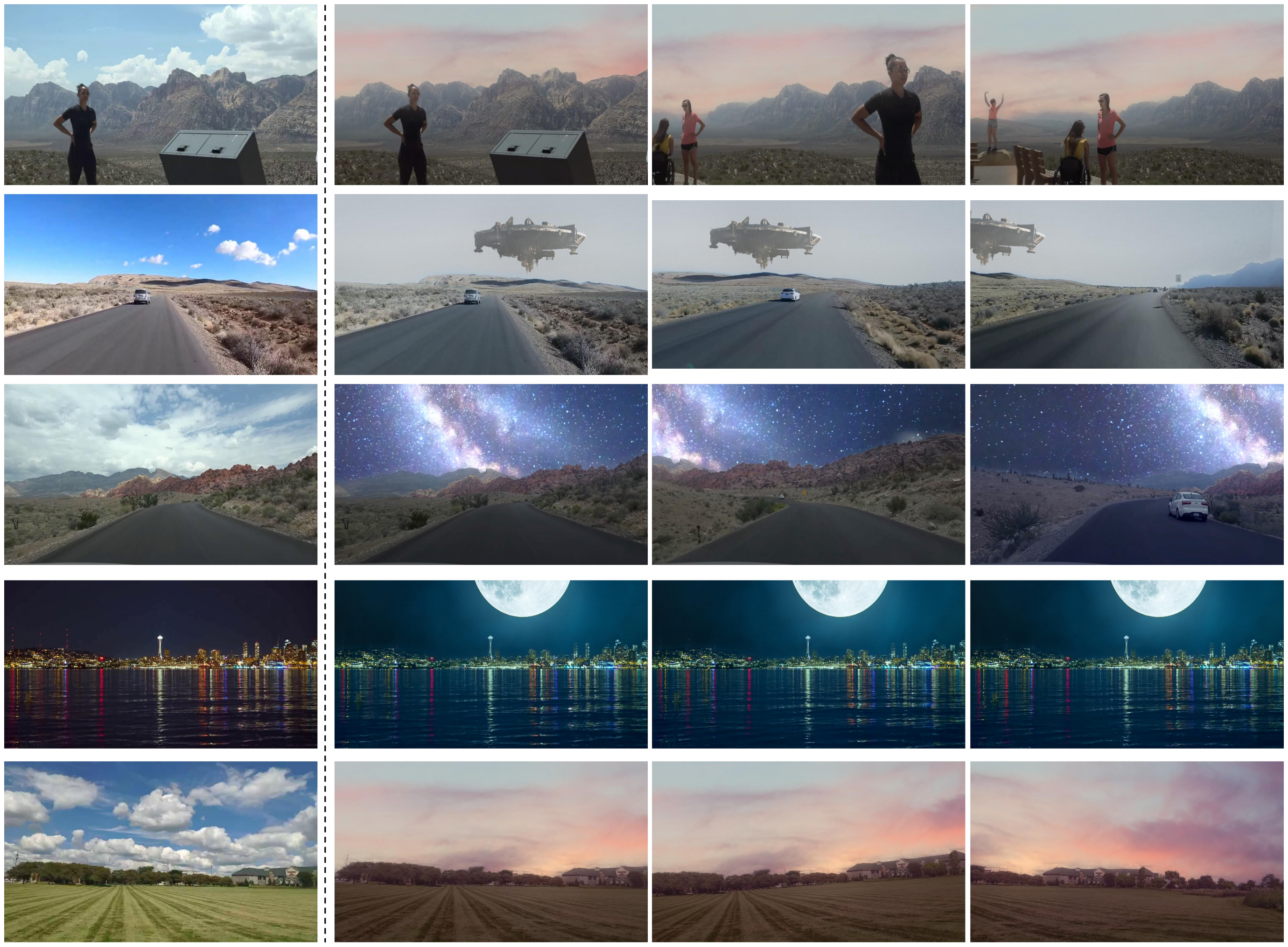

Figure 5. In each row, the leftmost is the starting frame of the input video, and the images on the right are our processed output at different time steps. We recommend readers check out our website for our animated demonstration.

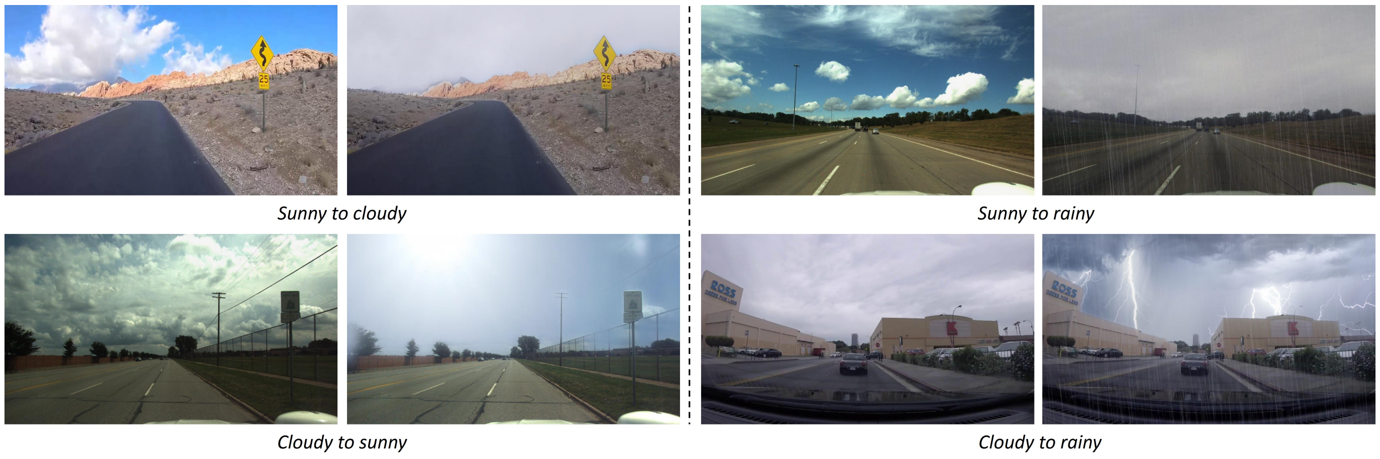

Figure 6. Weather translation results by using our method. First row: sunny to cloudy, sunny to rainy. Second row: cloudy to sunny, cloudy to rainy. In each group, the left shows the original frames, and the right shows the translation result.

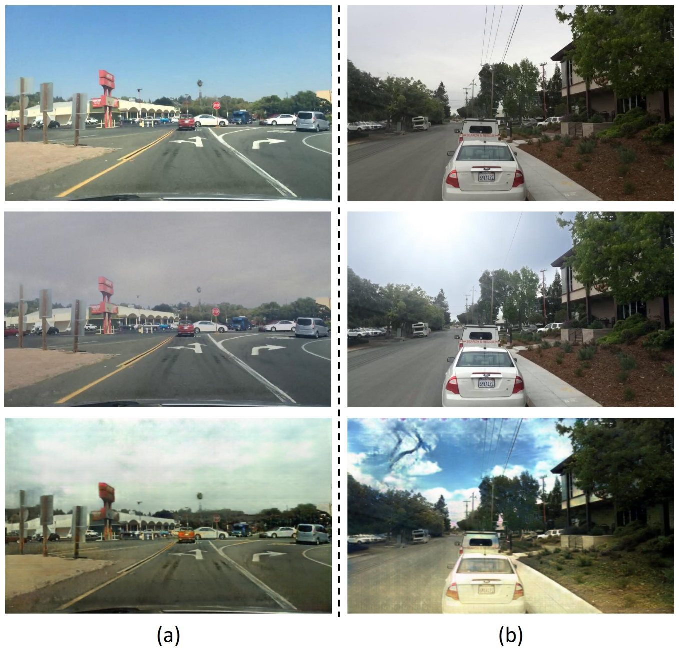

Figure 7. Qualitative comparison between our method and CycleGAN [26] on BDD100K [25]. (a) Sunny to cloudy translation, (b) Cloudy to sunny translation. 1st row: input frames; 2nd row: our results; 3rd row: CycleGAN's results.

	City→ City		City → BDD	
Training mode	mIOU	pxlAcc	mIOU	pxlAcc
No augment	0.7775	0.9609	0.3689	0.8307
Simple-augment	0.7719	0.9596	0.3974	0.8559
Sky-augment (ours)	0.7798	0.9607	0.4376	0.8706

Table IV
SEMANTIC SEGMENTATION ACCURACY OF A SOTA MODEL HARDNET [30] ON CITYSCAPES [24] AND BDD100 [25]. WE TRAIN HARDNET ON CITYSCAPES WITH DIFFERENT AUGMENTATION METHODS AND THEN TEST ON THE TWO DATASETS ACCORDINGLY.

Naturalness Image Quality Evaluator (NIQE) [28] are used for evaluation. The two metrics were originally introduced as a no-reference image quality assessment method and are recently widely used for evaluating image synthesis results. We see our method outperforms CycleGAN with a large margin in both quantitative metrics and visual quality.

B. Content-aware Data Augmentation

Another potential application of our method is data augmentation. Domain gap between datasets and complex real-world environment poses great challenges for data-driven computer vision methods [31]. Even in modern large-scale datasets like MS-COCO [32] and ImageNet [33], the sampling bias is still a problem, limiting the generalization of the model to the real world. For example, domain-sensitive visual perceptron models in self-driving may face problems at night or on rainy days due to the limited examples used for training.

In this experiment, we take traffic scene semantic segmentation as an example and test whether our method can help improve the generalization ability of perception models. We choose HarDNet [30], a state-of-the-art real-time traffic scene segmentation model, as our baseline. We test on a challenging cross-domain evaluation task - training on Cityscapes [24] and then testing on BDD100k or Cityscapes validation set. Table IV shows the accuracy w/ or w/o applying sky augmentation during the training. In the 2nd row of Table IV, we also compare with conventional "content-agnostic" data augmentation method, including random gamma, random brightness, and random saturation. We set γ=(1.0, 1.5), brightness_factor=(0.5, 1.5), and saturation_factor=(0.5, 1.5) for a fair comparison. We can see that by using our content-aware augmentation method, the segmentation accuracy can be greatly improved on BDD100k. When both training and testing on Cityscapes, using our augmentation method can provide comparable accuracy to its baseline method.

		Phased time overhead (s)		
Resolution	Speed	Phase I	Phase II	Phase III
640×360 pxl	24.03 fps	0.0235	0.0070	0.0111
854×480 pxl	14.92 fps	0.0334	0.0150	0.0186
1280×720 pxl	7.804 fps	0.0565	0.0329	0.0386

Table V
SPEED PERFORMANCE OF OUR METHOD AT DIFFERENT OUTPUT RESOLUTIONS. THE SPEED IS TESTED ON VIDEOS WITH RICH SKY TEXTURES. PROCESSING PHASES: I. SKY MATTING; II. MOTION ESTIMATION; III. BLENDING.

C. Speed Performance

Table V shows the speed performance of our method. When testing the speed, we assume the videos are with rich sky textures and only use optimal flow in the background to compute the sky motion. We test on a desktop PC with an NVIDIA Titan XP GPU card and an Intel I7-9700k CPU. We can see our method reaches a real-time processing speed (24 fps) at the output resolution of 640×320 and a near real-time processing speed (15 fps) at 854×480 but still has large rooms for speed up. Since there is a considerable part of the time spent in the sky matting stage, the processing speed can be easily improved by replacing the ResNet-50 with a more efficient CNN backbone, e.g. MobileNet [34, 35] or EfficientNet [36].

D. Controlled Experiments

We design the following experiments to evaluate the effectiveness of different components of our method, including soft matting, positional embedding, recoloring, and relighting.

1) Segmentation vs Soft Matting: To evaluate the sky matting network, we design the following controlled experiments and visually compare the blending results generated by 1) hard pixel segmentation, 2) soft matting before refinement, and 3) soft matting after refinement. We also compare the matting accuracy on the validation set of the dataset [4] (ADE20K+DE+GF). Fig. 8 shows the sky matte and the sky image blending results generated by the above settings. We can see the hard pixel segmentation produces undesired artifacts near the sky boundary, and the refinement stage in our method helps generate a more accurate sky region and a much more visually pleasing blending result.

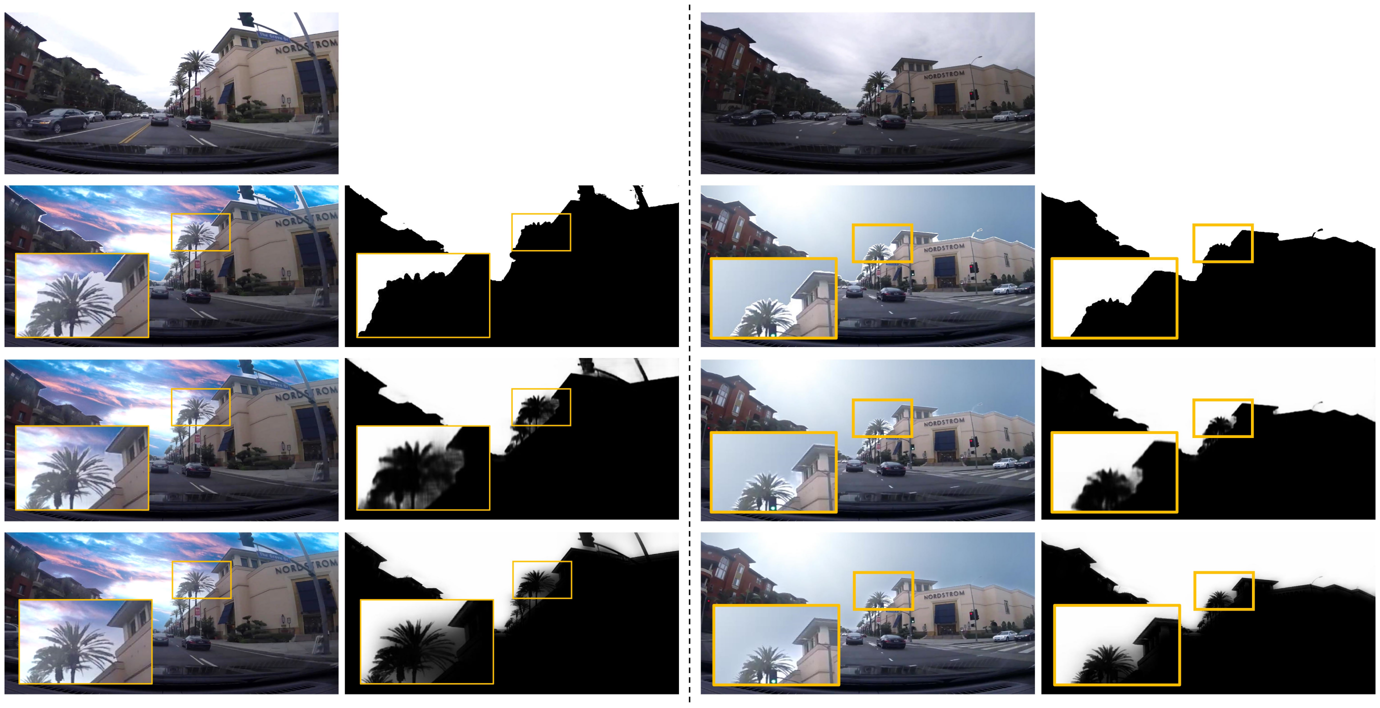

Figure 8. 1st row: an input frame. 2nd row: blending result by using hard pixel sky masks. 3rd-4th row: blending result based on soft sky matting w/o and w/ applying refinement.

	w/ refinement	w/o refinement
w/ positional encoding	27.31 / 0.924	27.17 / 0.929
w/o positional encoding	27.01 / 0.919	26.91 / 0.914

Table VI
MEAN PIXEL ACCURACY (PSNR / SSIM) OF OUR SKY MATTING MODEL ON THE DATASET [4] W/ AND W/O USING POSITIONAL ENCODING ("COORDCONV"S IN TABLE I). THE ACCURACY BEFORE AND AFTER THE REFINEMENT IS ALSO REPORTED. HIGHER SCORES INDICATE BETTER.

Table VI shows the mean pixel accuracy (PSNR and Structural Similarity (SSIM) index [37]) of our matting model w/ and w/o the refinement. We can see the refinement brings a minor improvement in PSNR (+0.14%) but does not improve SSIM. Although the improvement is marginal, we find that the visual quality can be greatly improved, especially at the boundary of the sky regions.

2) Positional embedding: We also test the matting network w/ or w/o using the coordinate convolution. Here we replace all the "coordinate conv" layers with standard convolution layers. Table VI shows the accuracy of the matting network under different configurations. We see a noticeable accuracy drop when we replace those coordinate conv layers (PSNR -0.26 and SSIM -0.015 before refinement; PSNR -0.30 and SSIM -0.005 after refinement). This suggests the position encoding provides important priors for the sky matting task.

3) Recoloring and relighting: We compare the sky blending result of our method w/ or w/o using recoloring and relighting. In Fig. 9, we give two groups of examples and compare the results of "linear blending only", "linear blending + recoloring", and " linear blending + recoloring + relighting". The results illustrate the importance of our two-step correction - the recoloring can help eliminate the color-tone conflict between foreground and background while the relighting can correct the intensity of ambient light and further improve reality.

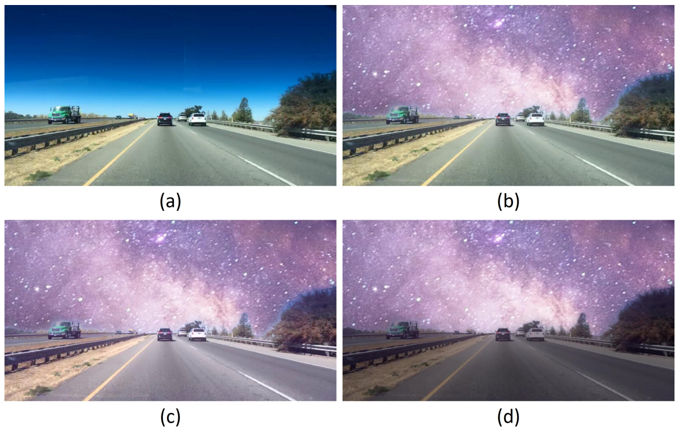

Figure 9. (a) An input frame from BDD100K [25]. (b) linear blending result (Eq. 23 only). (c) linear blending + recoloring (Eq. 24a). (d) linear blending + recoloring + relighting (Eq. 24a and 24b).

E. Limitation

The proposed framework also has limitations. Since the sky matting and motion estimation in our framework are both designed and implemented based on a data-driven fashion, when the testing scenarios are not covered by the training data and there is not enough sky texture for tracking, the method may fail in such cases.

Fig. 10 shows a failure case of our method, where the sky matting network fails to detect the sky regions at nighttime

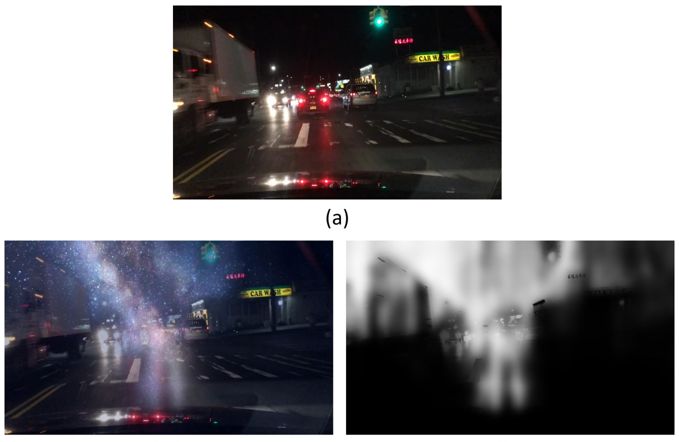

Figure 10. A failure case of our method. (a) An input frame from BDD100K [25] at nighttime. (b) Sky augmentation result. (c) Wrongly detected sky regions.

and thus generates a wrongly blended result. Meanwhile, in this example, the virtual camera also fails to synchronize with the motion of the real camera. Also, if there is water on the road under poor light condition, the sky matting method tends to recognize the water as a part of the sky.

V. Conclusion

We study video sky replacement - a new problem in computer vision and propose a purely vision-based solution for this task. We decompose video sky replacement into three proxy tasks: motion estimation, soft sky matting, and sky blending. Our method neither relies on the inertial measurement unit integrated on the camera devices nor requires manual interaction. Our method achieves generating highly realistic sky effects in real-time. As a by-product, our method can be also used for weather simulation and can be used as a new data augmentation strategy to improve the generalization ability of perception models.

VI. Author contribution

Zhengxia Zou started this project, completed the first version of this paper, designed the motion propagation algorithm. Rui Zhao implemented the flow propagation and traffic scene data augmentation. Zhengxia Zou and Shuang Qiu analyzed the projected motion equation of the camera. Tianyang Shi conducted the semantic segmentation experiment on Cityscapes and BDD100k. Zhenwei Shi provided overall support for this project.

References

[1] L. Tao, L. Yuan, and J. Sun, "Skyfinder: attribute-based sky image search," *ACM transactions on graphics (TOG)*, vol. 28, no. 3, pp. 1–5, 2009.

[2] Y.-H. Tsai, X. Shen, Z. Lin, K. Sunkavalli, and M.-H. Yang, "Sky is not the limit: semantic-aware sky replacement." *ACM Trans. Graph.*, vol. 35, no. 4, pp. 149–1, 2016.

[3] S. Rawat, S. Gairola, R. Shah, and P. Narayanan, "Find me a sky: A data-driven method for color-consistent sky search and replacement," in *International Conference on Multimedia Modeling*. Springer, 2018, pp. 216–228.

[4] O. Liba, L. Cai, Y.-T. Tsai, E. Eban, Y. Movshovitz-Attias, Y. Pritch, H. Chen, and J. T. Barron, "Sky optimization: Semantically aware image processing of skies in low-light photography," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 526–527.

[5] Y. L. Anh-Thu Tran, "Fakeye: Sky augmentation with real-time sky segmentation and texture blending," in *Fourth Workshop on Computer Vision for AR/VR*, 2020.

[6] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in nd images," in *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, vol. 1. IEEE, 2001, pp. 105–112.

[7] R. P. Mihail, S. Workman, Z. Bessinger, and N. Jacobs, "Sky segmentation in the wild: An empirical study," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–6.

[8] C. La Place, A. Urooj, and A. Borji, "Segmenting sky pixels in images: Analysis and comparison," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1734–1742.

[9] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 633–641.

[10] T. Halperin, H. Cain, O. Bibi, and M. Werman, "Clear skies ahead: Towards real-time automatic sky replacement in video," in *Computer Graphics Forum*, vol. 38, no. 2. Wiley Online Library, 2019, pp. 207–218.

[11] A. Atapour-Abarghouei and T. P. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2800–2810.

[12] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*. Springer, 2003, pp. 363–370.

[13] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International journal of computer vision*, vol. 56, no. 3, pp. 221–255, 2004.

[14] D. E. Zongker, D. M. Werner, B. Curless, and D. H. Salesin, "Environment matting and compositing," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1999, pp. 205–214.

[15] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 228–242, 2008.

[16] N. Xu, B. Price, S. Cohen, and T. Huang, "Deep image matting," in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski, "An intriguing failing of convolutional neural networks and the coordconv solution," in *Advances in Neural Information Processing Systems*, 2018, pp. 9605–9616.

[18] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 6, pp. 1397–1409, 2012.

[19] S. Lutz, K. Amplianitis, and A. Smolic, "Alphagan: Generative adversarial networks for natural image matting," *arXiv preprint arXiv:1807.10088*, 2018.

[20] A. R. Smith and J. F. Blinn, "Blue screen matting," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 259–268.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[24] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[25] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, “Bdd100k: A diverse driving dataset for heterogeneous multitask learning,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[26] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[27] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor, “The 2018 pirm challenge on perceptual image super-resolution,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[28] A. Mittal, R. Soundararajan, and A. C. Bovik, “Making a “completely blind” image quality analyzer,” *IEEE Signal processing letters*, vol. 20, no. 3, pp. 209–212, 2012.

[29] “Blend modes,” accessed Oct., 2020, https://en.wikipedia.org/wiki/Blend_modes.

[30] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin, “Hardnet: A low memory traffic network,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3552–3561.

[31] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, “Visual domain adaptation: A survey of recent advances,” *IEEE signal processing magazine*, vol. 32, no. 3, pp. 53–69, 2015.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[34] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.

[35] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[36] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” *arXiv preprint arXiv:1905.11946*, 2019.

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

Zhengxia Zou received his B.S. degree and his PhD degree from the Image Processing Center, School of Astronautics, Beihang University in 2013 and 2018. He is currently an Associate Professor at the School of Astronautics, Beihang University. During 2018-2021, he was a postdoc research fellow at the University of Michigan, Ann Arbor. His research interests include computer vision and related problems in remote sensing and autonomous driving. He has published more than 40 peer-reviewed papers including top-tier journals and conferences like the IEEE Transactions on Image Processing, the IEEE Transactions on Geoscience and Remote Sensing, the IEEE Conference on Computer Vision and Pattern Recognition, and the IEEE International Conference on Computer Vision. His research has been featured in more than 30 global tech media outlets and adopted by multiple application platforms with over 50 million users worldwide. His personal website is https://zhengxiazou.github.io/.

Rui Zhao received his B.S. degree and M.S. degree from the Image Processing Center, School of Astronautics, Beihang University in 2019 and 2022, respectively. He is currently a researcher in Netease Fuxi AI Lab. His research interests include computer vision, deep learning, and related problems in remote sensing and video games.

Tianyang Shi received his B.S. degree and M.S. degree from the School of Astronautics, Beihang University in 2016 and 2019, respectively. He is currently a researcher in Netease Fuxi AI Lab. His research interests include image processing, deep learning, and their application in games. He has published more than 20 peer-reviewed papers including top-tier conferences like the IEEE Conference on Computer Vision and Pattern Recognition, and the IEEE International Conference on Computer Vision. His homepage is https://www.shitianyang.tech/

Shuang Qiu is currently a postdoctoral Principal Researcher in the Booth School of Business, University of Chicago. He received his Ph.D. degree from the Department of Electrical Engineering and Computer Science, University of Michigan. His research interests include optimization, reinforcement learning, and computer vision. He serves as the reviewer or PC member for top conferences including International Conference on Machine Learning, Conference on Neural Information Processing Systems, AAAI Conference on Artificial Intelligence, and International Joint Conference on Artificial Intelligence etc.

Zhenwei Shi (M'13) received his Ph.D. degree in mathematics from Dalian University of Technology, Dalian, China, in 2005. He was a Postdoctoral Researcher in the Department of Automation, Tsinghua University, Beijing, China, from 2005 to 2007. He was Visiting Scholar in the Department of Electrical Engineering and Computer Science, Northwestern University, U.S.A., from 2013 to 2014. He is currently a professor and the dean of the Image Processing Center, School of Astronautics, Beihang University. His current research interests include remote sensing image processing and analysis, computer vision, pattern recognition, and machine learning.

Dr. Shi serves as an Editor for the Pattern Recognition, the ISPRS Journal of Photogrammetry and Remote Sensing, and the Infrared Physics and Technology, etc. He has authored or co-authored over 200 scientific papers in refereed journals and proceedings, including the IEEE Transactions on Pattern Analysis and Machine Intelligence, the IEEE Transactions on Image Processing, the IEEE Transactions on Geoscience and Remote Sensing, the IEEE Geoscience and Remote Sensing Letters, the IEEE Conference on Computer Vision and Pattern Recognition and the IEEE International Conference on Computer Vision. His personal website is http://levir.buaa.edu.cn/.