MobileSky: Real-time Sky Replacement for Mobile AR

Xinjie Wang, Qingxuan Lv, Guo Chen, Jing Zhang, Zhiqiang Wei, Junyu Dong, *Member, IEEE,* Hongbo Fu, Zhipeng Zhu, Jingxin Liu, and Xiaogang Jin *Member, IEEE*

Abstract—We present *MobileSky*, the first automatic method for real-time high-quality sky replacement for mobile AR applications. The primary challenge of this task is how to extract sky regions in camera feed both quickly and accurately. While the problem of sky replacement is not new, previous methods mainly concern extraction quality rather than efficiency, limiting their application to our task. We aim to provide higher quality, both spatially and temporally consistent sky mask maps for all camera frames in real time. To this end, we develop a novel framework that combines a new deep semantic network called *FSNet* with novel post-processing refinement steps. By leveraging IMU data, we also propose new sky-aware constraints such as temporal consistency, position consistency, and color consistency to help refine the weakly classified part of the segmentation output. Experiments show that our method achieves an average of around 30 FPS on off-the-shelf smartphones and outperforms the state-of-the-art sky replacement methods in terms of execution speed and quality. In the meantime, our mask maps appear to be visually more stable across frames. Our fast sky replacement method enables several applications, such as AR advertising, art making, generating fantasy celestial objects, visually learning about weather phenomena, and advanced video-based visual effects. To facilitate future research, we also create a new video dataset containing annotated sky regions with IMU data.

Index Terms—semantic segmentation, mobile augmented reality, sky replacement.

1 Introduction

The development of Augmented Reality (AR) technology blurs the line between reality and the virtual world. The sky is ubiquitous in everyday environments, and its augmentation is starting to attract attention in AR applications. For example, the stargazing app Stellarium Mobile[1] helps users identify stars, constellations, planets, and other deep-sky objects in real-time through interactive AR. The sky replacement problem is one of the challenging problems for sky augmentation. In mobile AR applications, we expect a user to lift a mobile phone with its camera and get a real-time sky replacement effect - the sky area is accurately separated and replaced with a stylized skybox. Moreover, the user can move the phone freely to get an immersive experience of the virtual sky mixed with reality. Solving this problem enables many AR applications, such as displaying rare astronomical views, showing fantasy celestial objects, creating fireworks, thunderstorm experience for automotive and aerospace training [1], future sky visualization for Architecture, Engineering and Construction (AEC) industries [2], and many more possible contents.

In order to solve the sky replacement problem, matting techniques [3], [4], [5] can be used to extract sky regions for each individual video frame. However, these methods have limitations to achieve our goal since (1) they often require user interactions [6]; (2) they are often too slow to run in real-time, especially on mobile devices [7], [8]. In recent years, video editing methods dedicated to sky replacement have emerged [9], [10], [11]. However, most of them require high computing power or do not support real-time performance. In addition, the above methods do not consider virtual-real world fusion in AR since they have mainly focused on offline images or videos. As far as we know, no existing method is capable of providing real-time, frame-by-frame sky mask maps with good quality at low computation on mobile devices today.

Therefore, the sky replacement problem for mobile AR applications poses new challenges that existing image (video) editing methods do not have. First, we must extract sky regions on mobile processors with low latency and low computation while maintaining high segmentation accuracy. Second, we want to achieve a temporally consistent composition of video frames. That means that the extracted regions must not have significant instability (jitter) over frames. Third, we want the composition to be visually pleasing enough, especially with minimal artifacts at the edges of the sky and non-sky regions. Finally, we must ensure that the solution can be integrated into a real-time mobile AR system.

We tackle these challenges through a novel and fast sky replacement system in a coarse-to-fine fashion. Our method leverages the real-time learning-based automatic segmentation method and refines the segmentation result using a real-time matting method to achieve an automatic, real-time matting result. First, we present an efficient automatic deep

- X. Wang (wangxinjie@ouc.edu.cn), Q. Lv (lvqingxuan@stu.ouc.edu.cn), Z. Wei (weizhiqiang@ouc.edu.cn) and J. Dong (dongjunyu@ouc.edu.cn) are with the Department of Computer Science and Technology, Ocean University of China.
- G. Chen (guo_chenoo@zju.edu.cn), J. Zhang (jing_z99@163.com) and X. Jin (jin@cad.zju.edu.cn) are with the State Key Lab of CAD&CG, Zhejiang University.
- H. Fu (hongbofu@cityu.edu.hk) is with the City University of Hong Kong.
- Z. Zhu (zhuzhipeng@oppo.com) and J. Liu (liujingxin@oppo.com) are with Guangdong OPPO Mobile Telecommunications Corp., Ltd.
- X. Jin is the corresponding author of this paper.
- Q. Lv, G. Chen, and J. Zhang contributed equally.

1. www.stellarium-labs.com/

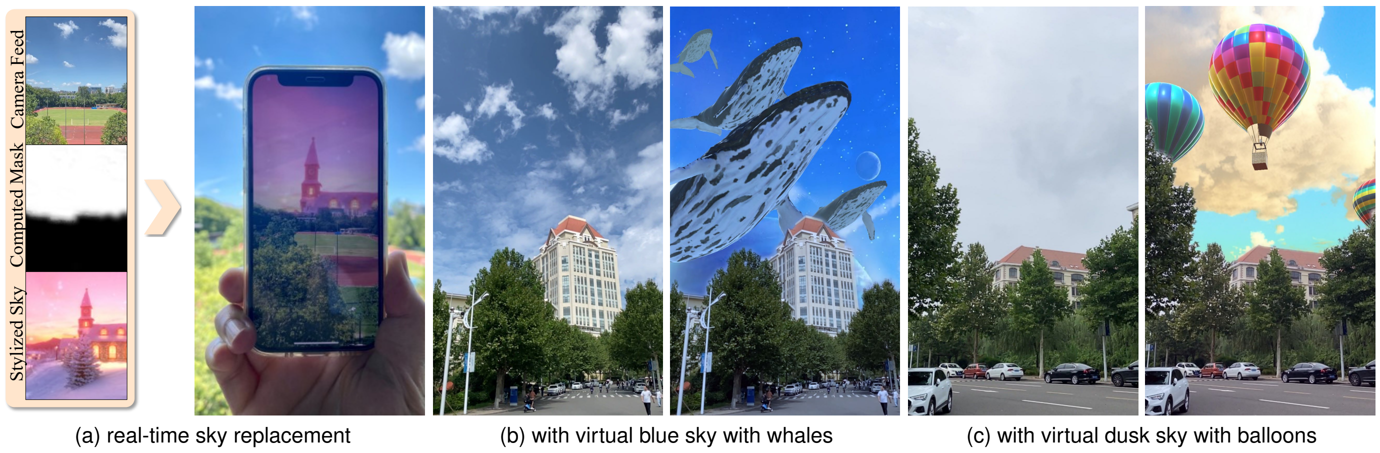

Camera Feed
Computed Mask
Stylized Sky
(a) real-time sky replacement
(b) with virtual blue sky with whales
(c) with virtual dusk sky with balloons

Fig. 1. Our *MobileSky* is able to naturally blend a stylized skybox (i.e., a pre-made panoramic sky video with virtual 3D objects) and camera feed into a visually pleasing AR scene in real time. This is enabled by real-time generation of a computed mask map based on the camera feed and IMU data for replacing sky regions at a real-time rate. We demonstrate: (a) An illustration of *MobileSky* application; (b) and (c) Two captured frames under different weather conditions and the composite examples with fantasy skies and imaginary objects.

semantic segmentation network to obtain a comparatively accurate pixel-wise two-class segmentation result in real time. It is a regular step for acquiring an original mask map, and the map inevitably contains rough segmentation edges and potentially misclassified pixels. There are two kinds of misclassification, (a) false-negative, where pixels belonging to the sky regions are misclassified as non-sky results, and (b) false-positive, where pixels belonging to the non-sky regions are misclassified as sky results. Direct use of this mask map for sky replacement would result in artifacts or flickering. Second, we design a series of sky-aware constraints: (1) for temporal consistency improvement: the false-negative mask pixels in the sky regions are corrected by the pixel-wise correspondence between the previous frame and current frame; (2) for position consistency improvement: the false-positive mask pixels in the non-sky regions are corrected by excluding the locations where the sky regions are not likely to appear in frames; and (3) for color consistency improvement: those weakly classified mask pixels with significantly different colors are re-evaluated by fully considering the color features of the sky regions and non-sky regions. With these constraints, we develop a per-pixel function to compute refined mask maps. Finally, we adopt an image matting algorithm called Guided Image Filter [12] to provide extra alpha values at the intersection boundaries with respect to the shape of edges.

We address the first challenge by optimizing all the steps of our pipeline in parallel so that our system is able to run fast enough on handheld devices (a smartphone in our experiments) with limited available resources. To tackle the temporal consistency challenge, we align previous frame and current frame using IMU (Inertial Measurement Unit) data, which are correlated over frames, and solve it using our newly developed temporal and position consistency constraints. For the third challenge, we improve the rough segmentation edges with more edge details and soft transitions by leveraging the color consistency constraint and image matting.

In summary, our work makes the following technical contributions:

- We present *MobileSky*, the first automatic sky replacement method for mobile AR experiences. By taking camera color frames and IMU data as inputs, our method produces a pleasing augmented sky effect running at around 30FPS (Hz). To the best of our knowledge, our work is the first sky replacement method that achieves real-time performance on mobile devices.
- We propose *FSNet*, a novel lightweight sky segmentation network, which outperforms lightweight baseline segmentation models in terms of inference speed and quality.
- We develop new sky-aware constraints to refine the sky regions by fully considering the color, sky position, and temporal coherence across neighboring video frames, and unify them into pixel-wise calculations of all the constraints to obtain better results.
- To facilitate future research, we create the first video dataset containing annotated sky regions with IMU data. The dataset contains 100 videos (around 29,000 camera frames) shot in various weather and environmental conditions. All videos were shot with handheld devices.[2]

We demonstrate the effectiveness of sky replacement maps produced by our method for mobile AR applications with enhanced effects. We have tested our method on a variety of video inputs and compared it against a set of state-of-the-art semantic segmentation and sky replacement algorithms. We also list evaluation metrics that reflect the challenges discussed above and perform extensive numerical evaluations. Finally, we conduct two user studies to show that our results meet good quality rendering results and accord with human preferences.

2 Related Work

This work aims to efficiently create high-quality and temporally-consistent video outputs where the sky regions

2. https://github.com/guoccoo/MobileSky

in an input video are replaced by stylized skyboxes. Collectively, this task entails image and video semantic segmentation, image and video matting, and image and video sky replacement. In this section, we review the closely related work in these areas.

2.1 Image and Video Semantic Segmentation

Sky segmentation in videos plays a significant role in the proposed sky replacement system. In order to find explicit sky regions, hand-crafted visual features were regarded as effective information to classify sky [13], [14], [15]. However, the results of such methods are often undesired because of the complex environment. On the other hand, with the development of deep convolutional neural networks (CNNs) [16], [17], the performance of semantic segmentation was boosted [18], [19], especially for complex real road scenes like Cityscapes [20].

Therefore, we employ deep learning-based methods to segment sky regions reliably because of their high performance [21], [22], [23], [24], [25]. After a comprehensive study of image or video segmentation techniques, we found that, in practice, sky segmentation (essentially a binary classification problem) was easier than the aforementioned general multi-class segmentation tasks. In other words, recent image segmentation methods are generally able to meet the segmentation accuracy requirement in our task. But the high computation cost required by the high-performance networks [18], [19], [21] is unaffordable for handheld devices. Video object segmentation methods, by taking temporal consistency into consideration, are usually designed for desktop PCs, and they are too time-consuming to be used on mobile devices [26], [27], [28]. We thus are more interested in building an efficient and effective segmentation network that is friendly for mobile devices.

One simple way to accelerate the segmentation performance is to use a lightweight backbone network, like MobileNetV2 [29], MixNet [30]. Whereas, with complex segmentation prediction modules, the efficiency on handheld devices is still not guaranteed. In terms of segmentation models used in previous sky replacement methods [9], [10], such a requirement of fast inference speed is still left without consideration.

In summary, our approach differs from the segmentation components in previous sky replacement methods [9], [10] in that we develop a novel and efficient segmentation model that allows our approach to outperform previous works in both processing speed and quality.

2.2 Image and Video Matting

Image or video matting is the most commonly used technique for background replacement. Matting can produce visually more pleasing composites than segmentation techniques but often requires human interactions (annotations) or known background information [31]. Examples include traditional trimap-based matting methods (e.g., Poisson matting [32], closed-form matting [6], KNN matting [33], information-flow matting [34]), neural network trimap-based matting (e.g., context-aware matting [35], index matting [36], and sampling-based matting [37]).

However, most of the aforementioned methods cannot achieve real-time performance on mobile devices since they mainly concern the matting quality rather than efficiency. In this paper, we do not directly perform the image matting process to generate the mask map. Instead, we generate high-quality segmentation results automatically in real time on mobile devices and then refine the results using the matting technique of [12] while keeping soft transitions at the segmentation boundaries.

2.3 Image and Video Sky Replacement

There have been several methods dedicated to sky replacement. For example, Lalonde et al. [14] used a traditional rule-based model to remove sun and clouds from the sky. Kaufman et al. [13] performed sky enhancement on photos by analyzing the color, position, and shape of sky regions.

On the other hand, sky regions can also be detected and segmented by performing semantic scene parsing on input images, which helps better understand the overall image content and layout [38]. Representative methods of scene parsing include exemplar-based label transfer [39] and superpixel matching [40]. To compose the replacement result, Halperin et al. [41] used a small CNN network to segment the sky region and estimate camera motion. This method, however, is more appropriate for far-field videos than for near-view videos. The latter, on the other hand, are frequently shot by AR applications. Tsai et al. [9] used FCNN segmentation both to segment the sky and to retrieve candidates from which to transfer sky, based on semantic layout similarity. Zou [10] proposed a purely vision-based approach with motion estimation of video for dynamic sky replacement and harmonization. However, most of them tend to focus on offline image/video editing that does not require real-time performance. Tsai et al. [9] mentioned that it took at least 0.1 seconds to get corresponding sky segmentation results on a desktop platform. Even with a modern desktop-level GPU, Zou's method [10] is only capable of achieving 24 FPS with a Resnet50 CNN backbone. This demonstrates that when it comes to achieving real-time speed on mobile devices, the computation costs of those methods are prohibitively expensive. To meet real-time requirements, we instead elaborately design a lightweight network structure and leverage spatial and temporal information from camera frames. We assess our approach's real-time performance on both desktop and mobile devices.

To our knowledge, there are only very few methods of video sky enhancement that achieve near-real-time performance on mobile devices. Fakeye [11] extracts sky regions in real time on smartphones by leveraging the y-coordinate and RGB color channels of each pixel. However, their method can only achieve an mIoU (an accuracy metric, see Sect. 4.3) of 76.89%, introducing noticeable segmentation errors. In contrast, we use deep learning to significantly improve prediction accuracy and refine spatial-temporal correlation steps to achieve better visual results.

3 Method

In this section, we will describe our fast sky replacement method. The overview of our method is shown in Fig. 2. In

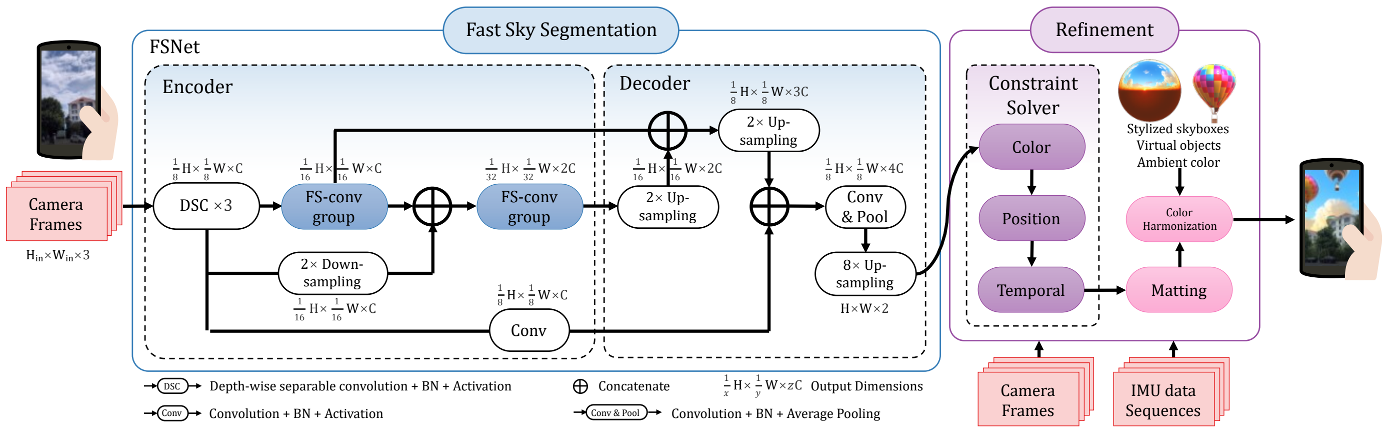


Fig. 2. The pipeline of our real-time sky replacement. It mainly consists of two parts: Fast Sky Segmentation and Refinement. With mobile camera frames and IMU data as inputs, we first estimate the sky regions using a pre-trained deep neural network *FSNet* to obtain rough mask maps. FSNet is an efficient single-image sky segmentation network following an encoder-decoder architecture. The encoder includes three Depth-wise Separable convolution [29] modules and two groups of our novel FS-conv (Fast Sky segmentation convolution) blocks as feature extraction modules. The decoder includes two multilevel 2× upsampling feature fusion modules and an 8× upsampling as the pixel-wise segmentation classifier. The dimensions above the each module are the output dimensions. The mask maps from FSNet are then refined by a highly parallelized constraint calculation to maintain temporal, position, and color consistency, and they are further improved with more edge details and soft transitions by leveraging an image matting method. Finally, the fine-tuned mask map is used to blend the color-harmonized camera frames with the virtual stylized skyboxes.

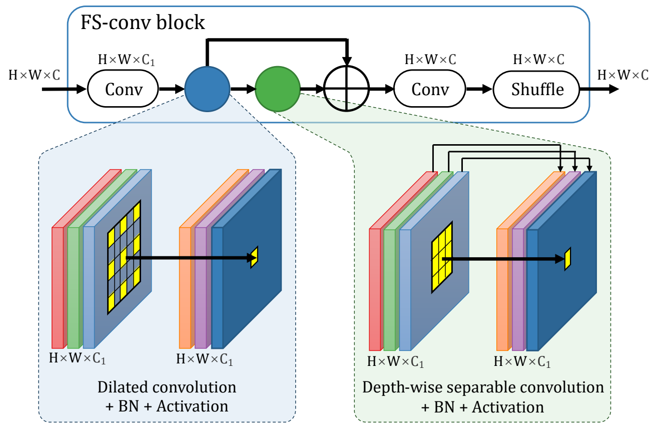
the first step, each camera video frame is passed through a lightweight encoder-decoder for real-time sky segmentation (Sect. 3.1), resulting in a binary mask image. In the refinement step (Sect. 3.2), the segmented mask map is then adjusted and refined by an efficient constraint calculation (Sect. 3.2.4) with the temporal (Sect. 3.2.1), position (Sect. 3.2.2), and color (Sect. 3.2.3) consistency constraints. The refined segmentation result is matted further using the Image Guided Filtering [12] algorithm to produce a result with soft edge details (Sect. 3.2.5). Following the completion of all of the preceding steps, the matte image is used as an alpha mask map to compose the actual frame and the stylized sky.

3.1 Real-time Sky Segmentation

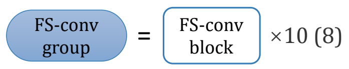
We will first introduce our FS-conv block, a novel lightweight feature extraction module (see Sect. 3.1.1). Then we will go over the details of FSNet (see Sect. 3.1.2), a lightweight encoder-decoder network that includes FS-conv groups. Finally, the FSNet segmentation results are converted and fed into the refinement step to correct minor segmentation errors.

3.1.1 FS-conv block

Fig. 3a depicts our newly developed deep feature extraction module, FS-conv block. An FS-conv block captures larger spatial correlations while simultaneously reducing the number of parameters by leveraging Depth-wise Separable Convolution [42] and Dilated Convolution [43]. The former splits the convolution computation into two steps: depth-wise and point-wise. Depth-wise convolution, as shown on the right side of Fig. 3a, employs a single convolutional filter for each input channel. Dilated Convolution, depicted on the left side of Fig. 3a, expands the kernel by inserting holes between its consecutive elements, thereby increasing the receptive field of the feature map. In addition, at the end of the FS-conv block, a channel shuffle operation is used


FS-conv block
H×W×C
H×W×C₁
Conv
Shuffle
Dilated convolution + BN + Activation
Depth-wise separable convolution + BN + Activation

(a) The structure of an FS-conv block



(b) The numbers of blocks in FS-conv groups

Fig. 3. Detailed illustrations of FS-conv block and FS-conv group. (a) The main components of an FS-conv block are dilated convolution and depth-wise separable convolution, with a channel shuffle operation to permeate the information. (b) The numbers of blocks in FS-conv groups are 10 and 8 (10 for the first FS-conv group and 8 for the second).

to allow information interaction across different channels. Fig. 3b shows the number of FS-conv blocks in each FS-conv group.

3.1.2 Lightweight Network Structure

We design a new lightweight network structure, FSNet, based on the FS-conv block, as shown in Fig. 2. Its goal is to improve segmentation efficiency and extract multi-scale features from different levels to refine prediction results.

FSNet is made up of an encoder and a decoder. First, the encoder uses three Depth-wise Separable Convolution modules to reduce the size of the feature map to 1/8 of

TABLE 1
The architecture details of FSNet. Output sizes are given for an example input of $3 \times 480 \times 480$.

	Operator	Channel	Number	Stride	Kernel	Output ($C \times H \times W$)
Encoder	DSC	64	3	2	7×7	$64 \times 60 \times 60$
	FS-conv	64	1	2	3×3	$64 \times 30 \times 30$
	FS-conv	64	9	1	3×3	$64 \times 30 \times 30$
	FS-conv	128	1	2	3×3	$128 \times 15 \times 15$
	FS-conv	128	7	1	3×3	$128 \times 15 \times 15$
Decoder	Upsampling	-	-	-	-	$192 \times 30 \times 30$
	Upsampling	-	-	-	-	$256 \times 60 \times 60$
	conv	256	1	1	1×1	$256 \times 60 \times 60$
	Upsampling	-	-	-	-	$2 \times 480 \times 480$

TABLE 2
Performance comparison on the ADE20K validation set with various FSNet network architectures, where f and s represent the number of blocks in the first and second groups, respectively. The best model performance is shown in bold.

f	s	Params	mIoU (see Sect. 4.3)
4	8	0.82M	81.25%
6	6	0.83M	82.94%
8	8	0.85M	86.46%
10	**8**	**0.87M**	**90.17%**
10	6	0.83M	84.68%
16	12	1.22M	88.31%

the original input image. Next, in order to exploit high-level semantic information and reduce the computational cost even further, we built two FS-conv groups composed of stacked FS-conv blocks. The decoder receives features with varying resolutions and fuses them together before upsampling them to the original size. A softmax layer is used at the decoder's end to make the final prediction (2 categories for each pixel: sky and non-sky). To keep our network lightweight, we simply use bilinear interpolation to upsample the 1/8 prediction result.

Table 1 shows the architecture of the best network model based on our experiments. DSC's best architecture is as follows: kernel=7×7, stride=2, and padding=3. The best architectures for standard convolution and FS-conv groups are: kernel=3×3, stride=1, and padding=1. It is worth noting that the first FS-conv block in each FS-conv group reduces the size of the feature map by half (i.e. stride=2). During training, we employ the cross entropy loss to optimize the entire network.

In Table 2, we compare the performance of various FSNet network on the ADE20K validation set. The result was obtained using the ADE20K dataset and the identical training settings, with an input resolution of 512×1024. According to the experimental data, setting $f = 10$ and $s = 8$ yields the best segmentation accuracy.

3.1.3 Real-time Inference

After deploying to a mobile platform with a third-party inference [44], we observe that it can reach real-time inference speed under 480 × 480 (or 640 × 360) input, and the mask maps are upsampled to the original size (for example, 1920 × 1080) to get the final composite results.

Let $\mathbf{I}$ represent an input frame from the camera, and $f_i^{seg} \in [0, 1]$ denote the classification value for the i-th pixel. This value indicates the probability that a pixel is within the sky regions. All the values f_i^{seg} form a vector, denoted as $\mathbf{f}^{seg}$. The binary classification for $\mathbf{f}^{seg}$ by using a simple threshold of 0.5 gives us $\mathbf{P}^{seg}$. Fig. 4 gives several examples of $\mathbf{I}$, $\mathbf{f}^{seg}$, and $\mathbf{P}^{seg}$, separately.

Although the output of the network $\mathbf{P}^{seg}$ might already achieve overall high classification accuracy, directly treating the classification results as a sky mask map would easily cause blending artifacts at the segmentation boundaries [10]. For example, minor errors are obtained in the bottom right figure in Fig. 4. Such errors dramatically occur over consecutive frames, leading to unstable results. To solve this issue, we consider refining the segmented images in a post-process to obtain more desirable matting results.

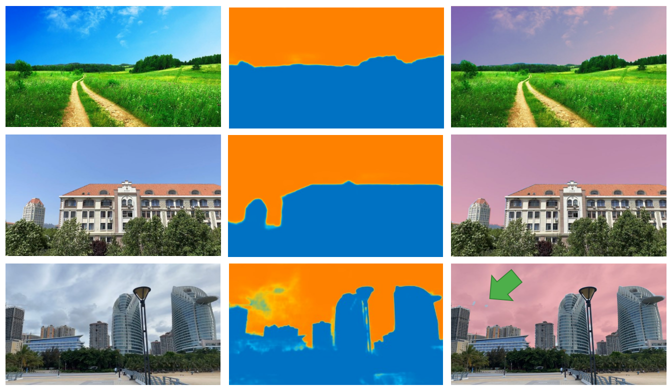


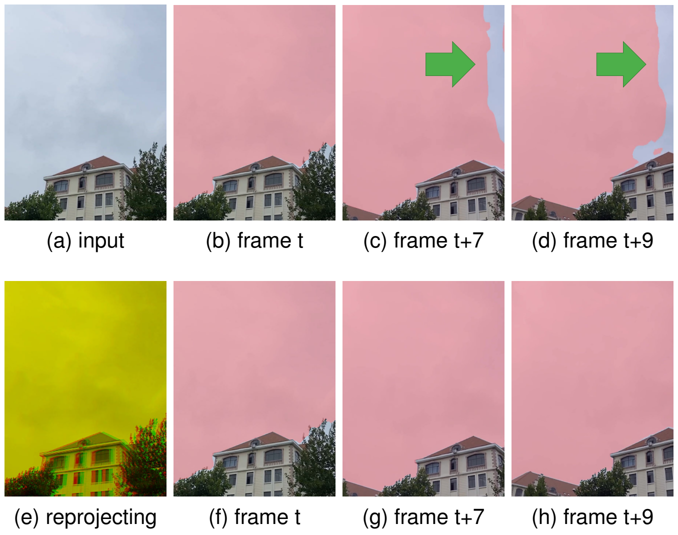
Fig. 4. Input and output examples of our sky segmentation network. (a) Sampled frames of a mobile camera $\mathbf{I}$. (b) Network output results $\mathbf{f}^{seg}$, mapped to a colormap for illustration. (c) The binary classification masks $\mathbf{P}^{seg}$ computed from $\mathbf{f}^{seg}$. The sky pixels are displayed with the semi-transparent light red color for illustration. Note that some pixels are classified as wrong results, which will be fixed in the refinement step.

3.2 Mask Map Refinement

In this section, we aim to correct those misclassified pixels that were mentioned in the previous section efficiently.

After experimenting with a variety of sky replacement scenarios, we summarize three types of pixels or objects that might lead to common and visually noticeable misclassifications: (1) some pixels of clouds in the sky regions; (2) some objects on the ground with colors similar to the sky regions, such as windows or water surfaces; (3) some objects close to the sky regions with complex boundaries (e.g., roofs, leaves).

To address these misclassification cases, we design a temporal constraint of correlating between the previous and current frames to refine the first type of misclassification (Sect. 3.2.1). We solve the second one by identifying the position where the sky appears in the viewfinder (Sect. 3.2.2). For the third one, we propose an adaptive color consistency constraint to refine complex boundaries and objects close to the sky regions but with significantly different colors against the sky regions (Sect. 3.2.3). Note that this constraint cannot solve all the third misclassification cases, since some error pixels might have similar colors to the sky regions. As the next step in this section, a parallelized constraint equation is used to unify the constraints into a single framework, to compute a refined mask map in parallel Sect. 3.2.4. Following that, we employ a fast image matting method to add extra alpha values to the mask map Sect. 3.2.5.


(a) input (b) frame t (c) frame t+7 (d) frame t+9

(e) reprojecting (f) frame t (g) frame t+7 (h) frame t+9

Fig. 5. Illustration of mask maps with and without temporal consistency refinement. In this example, we focus on the sky regions in frames t, $t+7$, and $t+9$ with obvious improvements after refinement. (a) An original frame from a camera video. (b), (c) and (d) Binary segmentation results by our network, showing temporal instability (jitter) over frames. Note that image-based semantic segmentation networks generally suffer from such a jitter problem. (e) A composite image of reprojecting the pixels of the previous frame to the current frame t (G channel) by using the projection matrices estimated based on IMU data (R channel). We only use one channel of each frame for illustration. Note that the pixels in sky regions are perfectly aligned. (f), (g) and (h) Corresponding segmentation results by our temporal consistency refinement. The sky regions now have a more stable appearance.

Finally, a simple color harmonization is used to achieve better composite results Sect. 3.2.6.

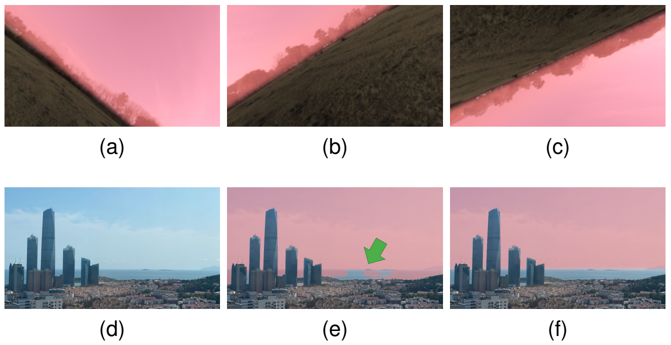
3.2.1 Temporal Consistency Constraint

In order to solve the potential instability caused by pixels of clouds in the sky regions that are not correctly classified, we rely on pixel-level correspondence between the previous and current frames to filter out the misclassification errors. Several commonly used correspondence methods are depth estimation [45], feature point method [46], and optical flow method [47]. However, these methods may fail due to the scarcity of features in sky regions, or too slow. Instead, we note that the sky regions can be considered to be at infinity from a user's perspective. Meanwhile, the motion of a handheld device is relatively continuous for a short period of time, meaning that the motion displacement of sky regions can be ignored between two adjacent frames.

Based on these observations, we use a homography matrix [48] $H_{\Delta t}$ to obtain the pixel-level correspondence between frames $t-1$ and t for the sky regions:

$$H_{\Delta t} = K R_{\Delta t} K^{-1} = K(R_t R_{t-1}^{-1})K^{-1}, \tag{1}$$

where K is the camera projection (or intrinsic) matrix, which can be retrieved from the device properties. R_{t-1} denotes the rotation matrix at time $t-1$ and R_t denotes the rotation matrix at t, which can be easily obtained from the IMU data. $H_{\Delta t}$ reprojects the pixels of frame $t-1$ to align with the pixels of frame t, as shown in Fig. 5e.


(a) (b) (c)

(d) (e) (f)

Fig. 6. Illustration of mask maps with and without position consistency refinement. (a), (b) and (c) Examples of aligned sky regions constrained by different IMU data. (d) An input video frame. (e) Incorrect segmentations typically occur at the edge where the sky meets the sea. (f) The sea regions are correctly distinguished from the sky regions by our position consistency constraint method.

Finally, we calculate the constraint $w_{i,t}^{temp}$ for temporal consistency. At each frame t, the per-pixel $w_{i,t}^{temp}$ is calculated as follows:

$$w_{i,t}^{temp} = \begin{cases} 1 & \text{if } D^2(I_{i,t}, I_{i,t-1}) < \epsilon_{temp}, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

where $t-1$ indicates the value computed in frame $t-1$ and reprojected by $H_{\Delta t}$. In practice, $H_{\Delta t}$ is applied on the UV coordinates of sampled textures on GPUs. D^2 denotes the Euclidean squared distance. We use the HSV color space to calculate D^2, since the HSV model corresponds to the human perception of color similarity [49]. ϵ_{temp} (Table 3) represents for a small quantity of error. The above equation is designed to maintain the sky regions as consistent as possible between consecutive frames.

Fig. 5 shows an example of the effect of enforcing temporal consistency. The top row shows segmentation results from Sect. 3.1. Some false negative pixels bring instability over frames. After using the constraint, these pixels are corrected, as shown in the bottom row of the figure.

3.2.2 Position Consistency Constraint

As we all know, the sky regions cannot appear in the lower part of the viewfinder of a smartphone when one holds the device strictly vertically (we called this *the standard pose*, assuming that a user is standing on or near the ground rather than high in the air). However, in practice, the local vertical direction of a captured frame is not always oriented perpendicularly to the ground, as the user may rotate the device freely when taking a photo. That is the reason why we do not use the vertical position constraint in the segmentation network. Fortunately, we can perform the constraint refinement by using IMU data after the segmentation step is finished.

The rotation matrix R_t derived from IMU data (also mentioned in Sect. 3.2.1) is actually the rotation matrix from the camera space to the world space, and the camera projection matrix K helps transform frames from local camera coordinates to Normalized Device Coordinates (NDCs, denoted as $\mathbf{n}$). Similar to Equation 1, we obtain the correspondence between $\mathbf{n}_{i,t}$ at frame t and $\mathbf{n}_{i,std}$ of *the standard pose* (i.e., people are positioned at the world space coordinate origin, holding the device strictly vertically, with horizon lines running parallel to the x-axis and perpendicular to the y-axis. Therefore, the matrix that transforms the standard pose from the world space to the local camera space is an identity matrix, E). More specifically, we have:

$$\mathbf{n}_{i,std} = (K(ER_t)K^{-1}) \cdot \mathbf{n}_{i,t}. \tag{3}$$

Then, we use the y-direction (vertical) of $\mathbf{n}_{i,std}$ to obtain the position constraint. It can be written as:

$$w_{i,t}^{pos} = smooth((0,1,0,0) \cdot \mathbf{n}_{i,std}), \tag{4}$$

where $w_{i,t}^{pos}$ denotes the position constraint that the sky regions are not allowed to appear in the lower part of $\mathbf{n}_{i,std}$. $smooth(x) \in [0,1]$ is a smooth transition function to avoid sharp borders. For example, $smooth(x) = 0.5 + 0.5 * tanh(20x)$ is used in practice in Fig. 6.

The top row of Fig. 6 shows examples of aligned sky regions constrained by different IMU data, i.e., using the IMU data to identify the position of horizon lines [50]. The regions below the horizon will never have sky pixels. The bottom row of Fig. 6 shows the correction of false positive pixels in daily shooting scenes by the position constraint. In addition, the constraint does not refine those non-sky objects above the horizon (such as trees and buildings or objects that obscure the user when shooting from an elevated perspective). These objects will be refined in the next step.

3.2.3 Color Consistency Constraint

As described at the beginning of Sect. 3.2 as well as in the previous paragraph, there might be some objects close to the sky regions with complex boundaries, thus leading to misclassification. These objects or pixels could not be refined by the temporal or position constraint since they tend to be near the segmentation boundaries, where the pixels fail to be aligned by IMU data or be judged as non-sky regions. To solve this problem, we present a new color-based sky-aware constraint to improve pixels that have significantly different colors against the sky regions. Since common color-based background segmentation methods such as histogram-based [51], clustering-based [52], and fuzzy-based [53] methods do not support real-time performance, we design a thresholding-like color constraint by taking the average color of the sky regions and the average color of neighbor pixels into account.

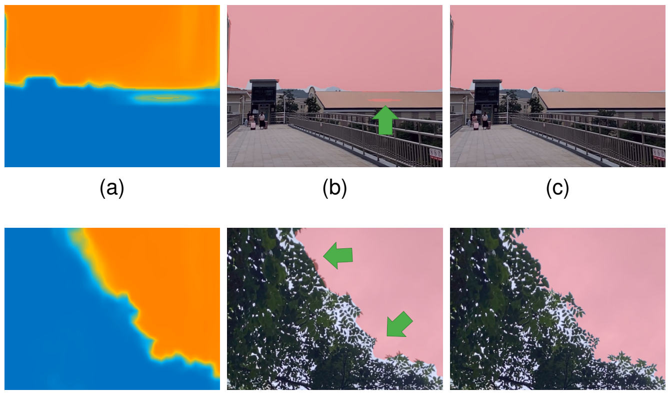


Fig. 7. Illustration of color consistency improvement. (a) and (d) denote $\mathbf{f}^{seg}$ from Sect. 3.1. (b) and (e) are mask maps without color consistency improvement. (c) and (f) are mask maps with color consistency improvement. Note that color consistency is not always correct since it loses local color features. Note that we only refine the pixels with weakly classified values which $(f_i^{seg} - 0.5) < \epsilon_{col}$. This figure shows two types of scenarios that can be refined using color-based correction methods: the top row shows the ability to refine incorrectly classified pixels with significant color differences from the same class; the bottom row shows the ability to preserve outlines of small objects like leaves.

We use $\mathbf{I}^{sky}$ to represent a color image copied from $\mathbf{I}$ ($\mathbf{I}$ has already been mentioned above), but only containing the pixels of the sky regions (the colors of the rest of the image are always set to black). We use N to indicate the number of all pixels in $\mathbf{I}$, $\mathbf{f}^{seg}$, $\mathbf{P}^{seg}$ and $\mathbf{I}^{sky}$, since these images share the same size as sampled textures on GPUs. Similarly, we use N^{sky} to indicate the number of pixels belonging to the sky regions.

Then we start to estimate the mean color in the sky regions $c^{sky} = (R,G,B)$, which is a vector of three color channels. Here, the sky regions are obtained from the previous sky segmentation step. The fundamental way to calculate c^{sky} is by summing the colors of all-sky pixels and dividing the result by the number of sky pixels: $c^{sky} = \frac{1}{N^{sky}} \sum_{i \in N} I_i * P_i^{seg}$. However, the time complexity of this step is $O(N)$ in the number of pixels N. A time complexity $O(N)$ of such computation of each frame would result in hundreds of milliseconds, making real-time performance difficult to achieve. Thus, we exploit a speedup strategy using a box filter with a $k \times k$ kernel to achieve parallel calculation. The GPU implementation of a box filter is data-paralleled at each pixel. First, the mean colors of $\overline{\mathbf{I}^{sky}}$ and $\overline{\mathbf{P}^{seg}}$ can be written as:

$$\overline{\mathbf{I}^{sky}} = \frac{1}{N} \sum_{i \in N} I_i * P_i^{seg}, \text{ and } \overline{\mathbf{P}^{seg}} = \frac{1}{N} \sum_{i \in N} P_i^{seg} = \frac{1}{N} N^{sky}. \tag{5}$$

In Equation 5, P_i^{seg} is a binary classification value. If its value is 1, the pixel is in the sky region, and vice versa. Thus $\overline{\mathbf{P}^{seg}}$ can be simplified by summing the number of pixels in all the sky regions and dividing by N. Substituting Equation 5 into the fundamental equation of c^{sky}, we can obtain:

$$c^{sky} = \frac{1}{N^{sky}} \sum_{i \in N} I_i * P_i^{seg} = \frac{1}{N\overline{\mathbf{P}^{seg}}} N\overline{\mathbf{I}^{sky}} = \frac{1}{\overline{\mathbf{P}^{seg}}} \overline{\mathbf{I}^{sky}}. \tag{6}$$

Then we replicate the filter $\log_k N$ times of downsampling to get the mean colors of $\overline{\mathbf{I}^{sky}}$ and $\overline{\mathbf{P}^{seg}}$, and then get c^{sky}. The time complexity of calculating the mean color of an image is $k \log_k N$.

Our task is to evaluate whether a pixel is closer to c^{sky} or closer to the mean color of its neighbor pixels c_i^{blur}, i.e., $\frac{1}{R^2} \sum_{j \in S_i} I_j$ (where R is the size of a squared window S). We use B_i^{col} to indicate the estimated classification value. It is described as:

$$B_i^{col} = D(I_i, c^{sky}) < D(I_i, c_i^{blur}). \tag{7}$$

Similar to Equation 2, the Euclidean distance function D is determined in the HSV color space.

It is worth noting that this straightforward color-based classification may not produce good segmentation results.

As a result, only pixels with low classification values are refined using this method. As stated in Sect. 3.1.3, the range of f_i^{seg} is $[0, 1]$, which means that the closer the value of f_i^{seg} is to 1 (or 0), the more probable the i-th pixel is in the sky (or non-sky) areas. Therefore, we only refine pixels with low f_i^{seg} values (0.5 in our experiments). The constraint range is described using w_i^{col}, which is expressed as follows:

$$w_i^{col} = \begin{cases} 1, & \text{if } (f_i^{seg} - 0.5)^2 < \epsilon_{col} \\ 0, & \text{otherwise} \end{cases}, \tag{8}$$

where ϵ_{col} (listed in Table 3) is a threshold indicating the active range of the color consistency constraint. For the sake of clarity, we omit the subscripts showing the time since we only address one frame at a time in this paragraph. That is, $w_{i,t}^{col}$ and $B_{i,t}^{col}$ (which will be further used in Equation 9) are simplified to w_i^{col} and B_i^{col}, respectively.

Two examples in Fig. 7 show the impact of color consistency constraint. In the top row of the figure, the roof pixels are successfully corrected because the colors of these pixels have D values closer to c_i^{blur}. The bottom row shows the corrections around boundaries. Such artifacts are less obvious after the color refinement step.

3.2.4 Parallelized Constraint Calculation

To obtain a per-pixel refinement calculation for the fast sky replacement task, we put all the constraints together as follows:

$$\begin{aligned} P_{i,t} =& w_{i,t}^{pos} * \\ & (w_{i,t}^{temp} * P_{i,t-1} + (1 - w_{i,t}^{temp}) * \\ & (w_{i,t}^{col} * B_{i,t}^{col} + (1 - w_{i,t}^{col}) * P_{i,t}^{seg})), \end{aligned} \tag{9}$$

where $P_{i,t}$ and $P_{i,t-1}$ denote the refined pixel at frames $t-1$ and t, respectively. $w_{i,t}^{temp}$ is the temporal consistency constraint (see Equation 2). $w_{i,t}^{pos}$ is the position consistency constraint (see Equation 4). $w_{i,t}^{col}$ and $B_{i,t}^{col}$ are color consistency parameters defined in Equation 8 and Equation 7, respectively. All the values of $P_{i,t}$ form the refined mask map $\mathbf{P}_t$.

From Equation 9, we can see that $\mathbf{P}_t$ is only related to frames $t-1$ and t of the i-th pixel. Therefore, we can perform a highly parallelized calculation for $\mathbf{P}_t$, which will greatly improve the execution speed. Fig. 8 is a typical example showing the effectiveness of the refinement steps.

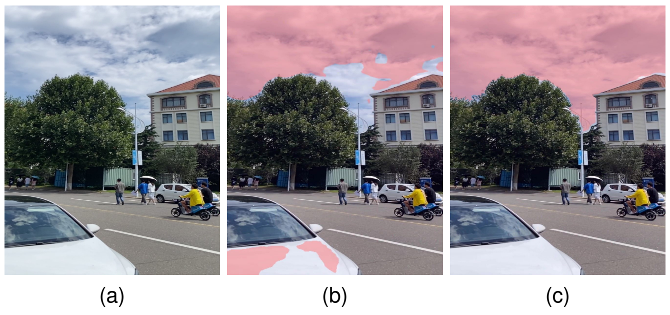


Fig. 8. A typical example showing the effectiveness of the entire refinement step. (a) An input camera frame. (b) The poor segmentation result without refinement. Note that the car below the horizon and the regions close to the building are misclassified. (c) The result with our refinement method.

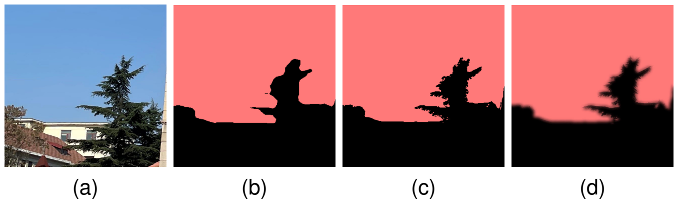


Fig. 9. An example of image matting improvement. (a) Part of an input camera frame. (b) $\mathbf{P}^{seg}$ from Sect. 3.1. (c) Our refined mask map. (d) A gradual transition at the intersection boundaries is obtained after matting the mask map.

3.2.5 Fast Image Matting

To obtain a final matting map $\mathbf{Q} = G(\mathbf{I}, \mathbf{P}, \epsilon_{mat}, R_{mat}, s_{mat})$, we use an efficient implementation [54] of Guided Image Filtering [12] to add extra alpha values to the pixels of edges. Here, G represents their guided filtering function, and $\epsilon_{mat}, R_{mat}, s_{mat}$ are the regularization parameter, size of the filter window, and subsampling ratio, respectively. Fig. 9 shows a comparison of outputs obtained with and without taking the guided filtering step.

3.2.6 Color Harmonization

Since the stylized skybox is pre-defined, it may not match the style of a natural environment. To achieve better composite results, we multiply the pixels of non-sky regions by the ambient color of the skybox in the final shader. Note that while color (style) transfer techniques [55], [56] might achieve better harmonization, we have to make trade-offs to ensure the efficient execution of our method.

4 Results and Analysis

We evaluate the proposed algorithm for rendering composite images with stylized skyboxes by casually capturing a large number of videos involving different types of scenes from handheld smartphones.

4.1 Implementation Details

In order to test the AR effects on off-the-shelf smartphones, we choose Unity® software[3] as a cross-platform running environment and embed the network module as a C++ native plugin. We also adopt the ARFoundation package, which is a package officially supported by Unity® software, for accessing the camera frames and IMU data of smartphones. Fig. 10 shows a complete pipeline for producing the *MobileSky* AR application.

Training Details. Most well-studied segmentation datasets do not cover rich sky scenes, such as CamVid [57] and Cityscapes [20], which are limited to data captured in urban landscapes from a driving car. We thus do not adopt such data to train our model. As for other segmentation datasets, like MS-COCO [58], Pascal-Context [59], COCO-Stuff [58], they only focus on "things" such as salient objects but not on "stuff" like major scene background components

3. https://unity.com/

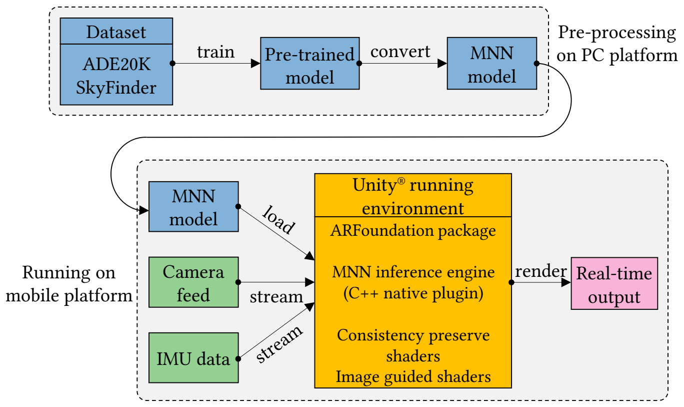


Fig. 10. The structure of our implementation.

so that they are not able to provide sufficient information of sky. Hence, we prefer to use ADE20K [60], a dataset containing indoor or outdoor sky scenes, as the main training dataset. To make the network be more sensitive to changing weather and environments, we combine the ADE20K [60] and SkyFinder [15] datasets to train our segmentation model. In detail, ADE20K is annotated with 150 different classes covering the most common things. SkyFinder only contains sky annotation information, which accounts for providing annotated sky under different weather conditions. We discard all the image patches with sizes less than 480×480 to fit with the crop size. To avoid overfitting, we perform commonly used data augmentation, such as randomly scaling the image in the range of 0.5 to 2.0 and randomly horizontal flipping with probability 0.5. On two RTX 2080Ti GPUs, all models are trained with 160 epochs. The Cross-Entropy loss is used as the objective function. We adopt mini-batch stochastic gradient descent (SGD) [61] to optimize the network with momentum 0.9 and weight decay 0.0005 and set the batch size of each iteration to 16. Meanwhile, the "poly" learning rate strategy is also applied during training where the initial rate is multiplied by $(1 - \frac{iter}{max_{iter}})^{power}$ per iteration with power 0.9 [24] to drop the learning rate. The initial learning rate of training is set to 0.01.

Mobile Network Inference Engine. Currently, there are not many options for mobile deep learning frameworks. Several representative and released mobile deep inference frameworks are MNN[4] [44], NCNN[5], PyTorch Mobile [62] and so on. According to their publicly available documentation, the performance among these frameworks does not differ significantly. Therefore, we select MNN as the inference engine of the network in our implementation for its easy-to-use APIs. We first utilize PyTorch [62] on a PC platform to build and train FSNet. The trained model is then converted to an MNN model file through MNN Convert Tools. Both the training and converting steps harness two NVIDIA RTX 2080Ti GPUs. The model file and inference engine library are then packaged into an executable and deployed to a smartphone. The MNN inference engine is deeply optimized for parallel computing, which is particularly suitable for implementing our parallel network.

4. https://github.com/alibaba/MNN
5. https://github.com/Tencent/ncnn

TABLE 3
Parameters of Sect. 3.2 and their default settings. All results shown in the paper and the supplementary material were generated with the same parameter settings.

Parameter	Value	Section	Description
ϵ_{temp}	0.01	3.2.1	Error threshold parameter
ϵ_{col}	0.0225	3.2.3	Error threshold parameter
R	5	3.2.3	Size of each squared window S_i
ϵ_{mat}	0.01	3.2.5	Regularization parameter
R_{mat}	16	3.2.5	Size of the filter window
s_{mat}	4	3.2.5	Subsampling ratio

It is worth noting that our converted MNN model file and the MNN inference engine are relatively small in size. Specifically, the size of our model file is only 6.34MB, and the size of the engine library is 1.91MB and 5MB for Android and iOS, respectively. With the lightweight models, our algorithm is friendly to those mobile applications that prefer to keep a small file size.

GPU Parallel Shaders. To implement the map refining method from Sect. 3.2 and guided image matting from Sect. 3.3, we calculate all the functions mentioned above by either using fragment (pixel) shaders (which are executed in GPU parallel) or designing a list of graphics commands. In particular, we use an efficient version of the Image Guided Filter [12], which is also mentioned in [54] and implemented in GPU parallel. Please refer to their work for more details.

AR Scene Generation. We use the obtained matting maps as alpha masks to blend real camera frames with a predefined virtual stylized skybox. The rotation of the virtual AR camera is controlled by IMU data. In addition, we use PBS (Physically-Based Shading) to render virtual objects for a realistic appearance.

Parameters. Table 3 lists all the parameters of Sect. 3.2 and their default settings. All results shown in the paper and the supplementary material were generated with the same parameter settings.

4.2 Effects and Performance

Fig. 11 shows a subset of the generated composite video sequences, and more comparisons and results are available in the supplementary materials. Experimental results demonstrate that our algorithm can handle input images with various scenes and generate a diverse set of visually pleasing sky backgrounds.

Besides this, we test the execution time of our method on several smartphones. As shown in Table 4, the chosen segmentation network takes around 27.4∼30.3 ms per frame using the CPU (note: the GPU implemented version has a generally consistent cost time, which is commonly the case according to MNN's experimental performance). Mask map refinement takes about 1.3∼1.7 ms. Fast guided image matting takes about 2.2∼2.9 ms. The total execution times include tracking with rotation measurements from the device's sensor (i.e., gyroscope), streaming camera frames, and rendering composite videos. The reported data shows the abilities of our method to replace sky as a powerable AR effect in real time.

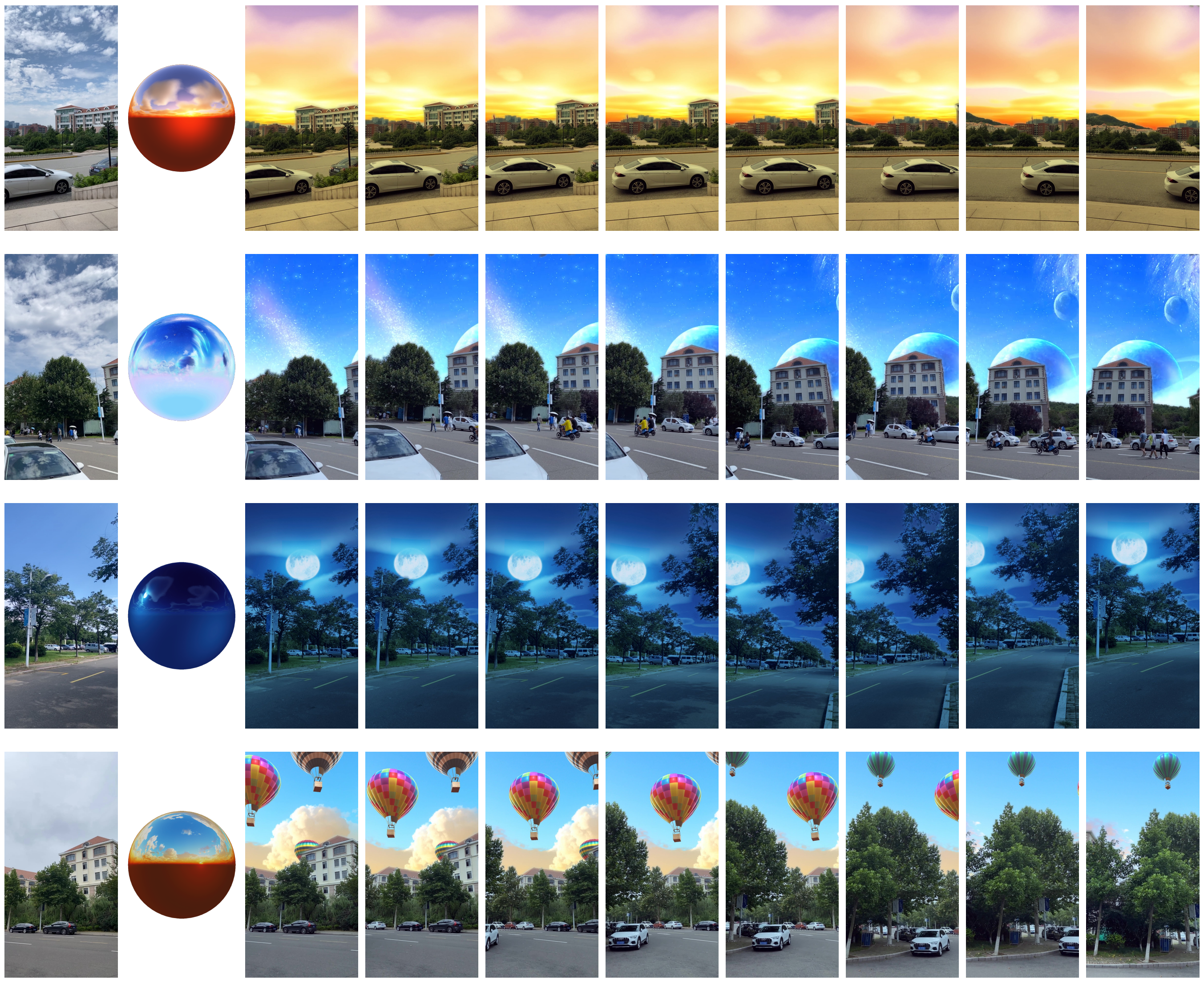


Fig. 11. Results of the proposed method: sampled frames of composite videos with stylized sky backgrounds. All the results are processed with color harmonization (see Sect. 3.2.6).

TABLE 4
Breakdown of the average timings (in ms) of the different stages in our pipeline on mid- and high-end phones. These values were obtained by running the whole application on real smartphones.

Device	Video Size	Segment	Refine	Matting	Others	Total
iPhone 11 Pro	640*360	30.3	1.7	2.6	1.2	35.8
iPhone 12	640*360	26.9	1.4	2.2	0.9	31.4
XiaoMi 11	640*360	28.3	1.3	2.9	0.9	33.4
Oppo Find X3 Pro	640*360	27.4	1.7	2.8	0.8	32.7

4.3 Evaluation Metrics

To quantitatively evaluate the quality of the proposed method, we use three metrics, including mIoU [21], MPA [21], FPS, and conduct two user studies, each of which measures a slightly different aspect of the quality.

mIoU. mIoU (mean Intersection over Union) is one of the most commonly used metrics in semantic segmentation. It essentially quantifies the percent overlap between the target mask and our prediction output. We use this metric to evaluate the quality of semantic segmentation results and the quality of the refined results. Specifically, this metric is defined as follows:

$$mIoU = \frac{1}{k}\sum_{i=0}^{k-1}\frac{p_{ii}}{\sum_{j=0}^{k-1}p_{ij}+\sum_{i=0}^{k-1}p_{ji}-p_{ii}}, \tag{10}$$

where p_{ij} means the number of the pixels of class i predicted to belong to class j. In other words, for a specific class i, p_{ii}, p_{ij} and p_{ji} denote true positive, false positive, and false negative, respectively. k is the number of categories. Since the sky segmentation network classifies pixels into two categories, $k = 2$. mIoU is evaluated in Sect. 4.4, Sect. 4.5 and Sect. 4.6.

MPA. Although MPA (Mean Pixel Accuracy) may not comprehensively reflect the segmentation ability, it is meaningful for evaluating our refined matting results. It can be formulated as the following equation:

$$MPA = \frac{1}{k}\sum_{i=0}^{k-1}\frac{p_{ii}}{\sum_{j=0}^{k-1}p_{ij}}. \tag{11}$$

MPA is evaluated in Sect. 4.4.

FPS. The above two metrics focus on the accuracy of methods. Since our method is eventually intended to be executed in the viewfinder of smartphones in real time, efficiency metrics are also important. We capture the execution time measured in FPS (Frame Per Second) to predict whether a method will be perceived as stuttering. Usually, an FPS of 30 or more is required in most mobile real-time applications. FPS is evaluated in Sect. 4.4 and Sect. 4.6.

User Studies. In order to qualitatively evaluate our method, we conducted two user studies. In particular, we design two different tasks to evaluate the perceptual quality and the rendering speed experience of the proposed algorithm, respectively. Please refer to Sect. 4.7 for more details.

4.4 Network Quantitative Comparison

In this section, we conduct comprehensive quantitative comparisons of the state-of-the-art sky segmentation models to study the superior efficiency of our segmentation model. Recent works regard sky segmentation or matting as a key component to achieve sky replacement, while their evaluation is not very thorough [9], [10]. Since previous works only report the performance of their own models, we reproduce most segmentation models and evaluate them under the same training process in order to make a fair comparison. The comparisons are performed on the SkyFinder and ADE20K datasets. It is worth noting that FCN [21] was used in previous work [9] and thus is considered as a competitor. Besides FCN, two extra lightweight segmentation models and another high-performance segmentation model, ENet [63], FastSCNN [64] and DeeplabV3+ [18], are also included. Meanwhile, we substitute the backbone of DeeplabV3+ by MobileNetV2 [29] to evaluate the deployment performance of such a lightweight network. The comparison includes mIoU, MPA, inference speed (FPS), and parameters, which are able to reflect efficiency.

Table 5 summarizes all results, and we have two obvious observations. First, among all the compared methods, FSNet achieves the best mIoU. We use Resnet50 [16] as the backbone for FCN and DeeplabV3+. We set the downsampling stride at 16 to achieve high efficiency at the cost of decreasing accuracy to some extent. Even though replacing Resnet50 by MobileNetV2, the segmentation accuracy is still lower than FSNet. For Enet and FastSCNN, due to the over-lightweight design, they are not able to converge on a good solution. Second, FSNet also has considerable efficiency. We test FCN with a 640×360 image, and it runs only at 5.61 FPS, which is unacceptable for real-time processing systems. Note that the inference speed of ENet and FastSCNN are not presented in Table 5, mainly because some operators of those models are not supported when deploying them to mobile devices. We experimentally find that they are able to achieve real-time speed on a general computing platform. However, the performance of FSNet is better than that of others by a large margin. For all models, we uniformly scale the video size to $640 \times H$, ensuring a balance of speed and precision, where H is derived based on the phone's aspect ratio. The above analysis shows that FSNet is the most suitable for real-time AR applications.

4.5 Ablation Studies for Consistency Constraints

Dataset with IMU data. IMU data is required for constraint ablation studies. However, none of the public datasets, including ADE20K and SkyFinder, include this information. To that end, we created a new video dataset with camera rotation information. The dataset is created specifically in the following steps: (1) We use handheld devices to shoot videos in various weather (sunny, cloudy, and rainy) and environmental (including buildings, trees, people, vehicles, and so on) conditions, using the IMU data acquisition interface provided by the Unity® ARFoundation package. We save all IMU data and the exact recording moment of each camera frame for further alignment because the frame rates of the IMU device and the video recording module are not synchronized. Each video lasts approximately 5$\sim$15 seconds. There are 100 videos in total, with 50 in landscape mode and 50 in portrait mode. (2) We align each camera frame with the IMU data based on the exact moment of recording. Redundant IMU information is removed, and camera frames without IMU information are discarded. The IMU data is then converted into 3D rotation matrices of the camera and saved to a file. (3) We manually label the segmentation mask map of each image frame as the ground truth.

Ablation studies. We conduct ablation studies using the new video dataset to evaluate the efficacy of the three constraints described in Sect. 3.2. Table 6 displays the ablation studies of four different implementations with varying consistency constraints. Because the dataset used in this section differs from that used in Sect. 4.4, the mIoU of the baseline model (FSNet) is slightly different.

For most frames, we find that FSNet works well. Because of the temporal consistency constraint, our method is able to avoid obvious segmentation errors that appear in some frames (see Fig. 5). The position consistency constraint is usually beneficial in scenes with vehicle or ground puddle reflections (see Fig. 6). Color consistency primarily deals with boundary segmentation details (see Fig. 7), so its improvement on mIoU is not as significant as the first two.

4.6 Comparison of Sky Replacement Methods

Next, to quantitatively assess our method and objectively compare our method to other baseline algorithms, we compare against four vision-based image (video) sky replacement methods [10], [11], [65]. The evaluated results are shown in Table 7. Since two of these methods [11], [65] do not provide publicly available implementations, we have to use the data in their papers to make rough comparisons. Compared with [65], our method achieves an mIoU score of 90.17%, higher than the 88.7% mIoU score with their method. Note that due to the lack of missing IMU data in the videos in the SkyFinder and ADE20K datasets, our scores are measured before using the refinement steps, i.e., by using FSNet only, while theirs are measured after their own refinement procedure. However, our method achieves a much higher FPS (31.84 vs. 0.035). In comparison to [11], our method has a significantly higher mIoU (90.17% vs. 76.89%).

TABLE 5
Performance comparison of main lightweight segmentation techniques. The best model performance is shown in bold. "-" indicates the corresponding model is not compatible with mobile platforms. Additional information of the execution time on a PC with an NVIDIA RTX 1080 GPU is given for reference.

Model	Input Size	Params	Package Size	mIoU	MPA	FPS (mobile)	FPS (PC)
FCN [21]	640 × 360	22.43M	90MB	77.88%	87.86%	5.61	58.98
DeepLabV3+ [18]	640 × 360	37.92M	151MB	75.54%	89.96%	3.47	27.67
MobilenetV2 & DeepLabV3+ [29]	640 × 360	12.12M	48.6MB	82.99%	**94.10%**	6.03	27.67
ENet [63]	640 × 360	**0.4M**	**1.5MB**	74.45%	81.80%	-	56.01
FastSCNN [64]	640 × 360	1.11M	4.5MB	81.22 %	85.84%	-	**160.74**
FSNet (Ours)	640 × 360	0.87M	1.7MB	**90.17%**	94.01%	**37.17**	132.02

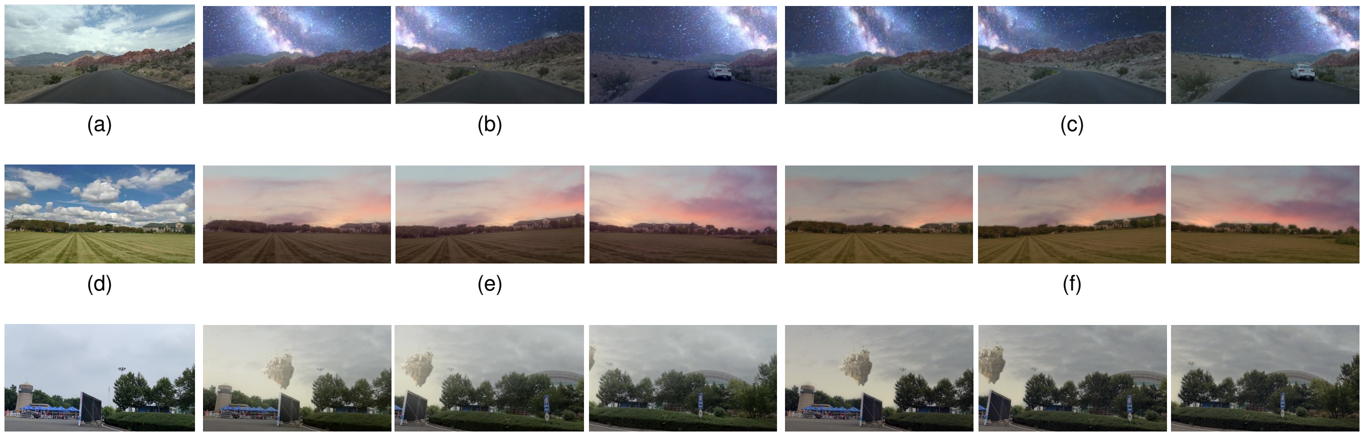


Fig. 12. Side-by-side visual comparisons between our method and Zou [10]. (a) and (d) Input videos proposed by Zou [10]. (b) and (e) Corresponding composite results from Zou [10]. (c) and (f) Our composite results running in real time on iPhone 12 Pro. Due to the lack of IMU data information in the pre-captured videos, we set $w_i^{temp} = 0$ and $w_i^{pos} = 0$ in these comparisons. Despite this, we still get competitive visual results. (g) The input videos captured by ourselves with a handheld smartphone. (h) The composite video generated by using the publicly available library released by Zou [10]. (i) Our composite results running in real time on iPhone 12 Pro.

TABLE 6
The ablation studies of eight implementations: the FSNet only, the FSNet with temporal (T), position (P), color consistency (C) constraint, and combinations of them, respectively. All the experiments are based on our video dataset with additional IMU data. Because these constraints may have overlapping refinement regions, the mIoU cannot be simply added.

Model	mIoU
Baseline (FSNet)	90.25%
Baseline + T	91.02%(+0.77%)
Baseline + P	90.87%(+0.62%)
Baseline + C	90.71%(+0.46%)
Baseline + T + P	91.08%(+0.83%)
Baseline + T + C	91.07%(+0.82%)
Baseline + P + C	90.94%(+0.69%)
Baseline + T + P + C	91.12%(+0.87%)

TABLE 7
Performance comparison of main sky replacement segmentation techniques. The best model performance is shown in bold. "-" indicates the corresponding result is not informed. Due to the lack of ground truth for handheld video with IMU data, our mIoU score is measured before the refinement steps (i.e., using FSNet only), while the mIoU scores of [65] are reported after their own refinement steps. In contrast, [11] has no refinement step.

Method	Device Type	Image Size	mIoU	FPS
Tsai et al., 2016 [65]	3.4GHZ Core Xeox CPU	800 × 800	88.7%	0.035
Tran and Le, 2020 [11]	Android	-	76.89%	-
Zou, 2020 [10]	TiTan XP GPU, I7-9700K CPU	640 × 360	86.16%	24.03
Ours	iPhone 12	640 × 360	**90.17%**	**31.84**

In addition, we compare against the method in [10], using the publicly available library released by the authors[6]. The fourth row in Table 7 shows the quantitative scores of their method. Our method achieves comparable mIoU and better FPS scores running on a mobile chip against the desktop PC they adopted when the other conditions are the same.

We show the composite frames of Zou [10] and ours for side-by-side comparisons. In the top two rows of Fig. 12, the input videos are from [10]. Due to the lack of the IMU data information in the pre-captured videos, we set $w_i^{temp} = 0$ and $w_i^{pos} = 0$ in this comparison. Despite this, we still get competitive visual results. In the bottom row, the input videos were captured by ourselves with a handheld smartphone. Our method has the ability to generate visually pleasing results with high efficiency.

Finally, we compare our method with a few commercially available alternatives, i.e., the real-time sky-segmentation filters on TikTok and Snapchat. The comparison is made at the level of rendering effects since there is no publicly available information regarding their specific methodology and performance, making quantitative comparison impossible. As shown in Fig. 13, the sky-segmentation effect filters on Snapchat and TikTok have varying degrees of error in discriminating between sky and non-sky regions and exhibit more obvious artifacts

6. https://github.com/jiupinjia/SkyAR

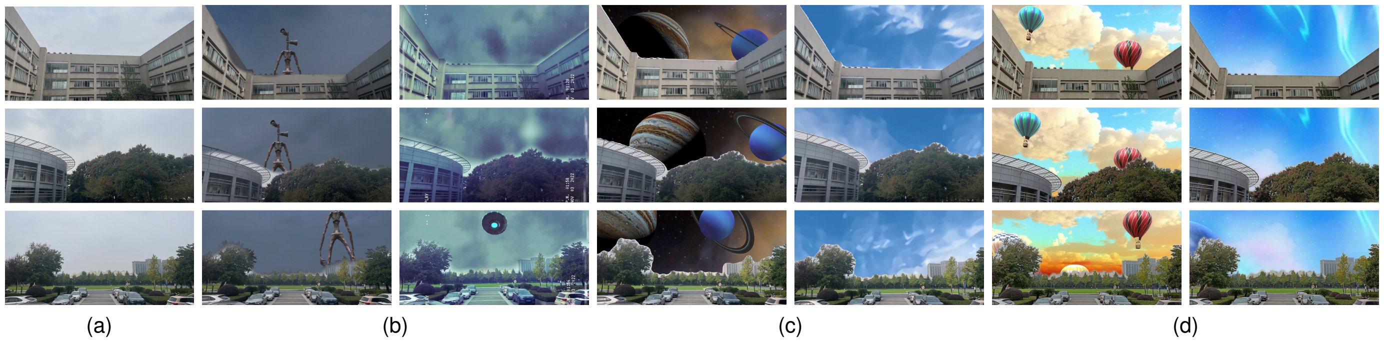


Fig. 13. A visual comparison of our approach to TikTok and Snapchat. (a) Input camera frames. (b) Results of Snapchat's sky segmentation. (c) Results of TikTok's sky segmentation. (d) Results of our method.

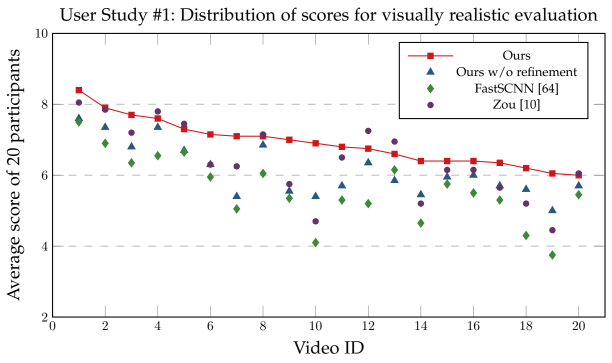


Fig. 14. Average evaluation scores for each video with x-axis sorted by our score. Overall, the visual quality of our method is comparable to that of FastSCNN and Zou. In addition, our method outperforms the method without the refinement step in most videos.

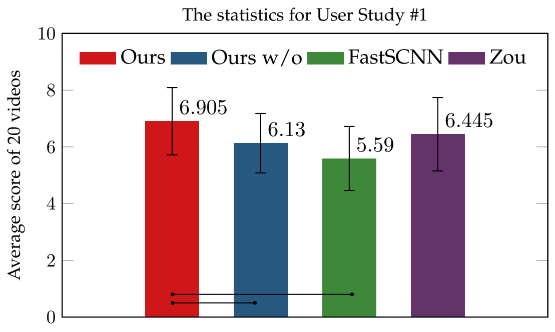


Fig. 15. Average visually realistic scores of different methods. Mean values and standard deviations are expressed via bar heights and error bars, respectively. A line connecting two bars indicates a statistically significant difference between our method and the others (ANOVA with post-hoc tests, $p < 0.05$). Ours vs Ours w/o: $p = 0.025$, Ours vs FastSCNN: $p < 0.001$.

near the margins. Moreover, some sky-segmentation effects on Tiktok and Snapchat have obvious latency, while ours always runs smoothly. In summary, our method performs sky segmentation more efficiently and effectively than these alternative filters.

4.7 User Studies

We have devised two user studies to objectively verify the quality and effectiveness of our sky replacement results. We designed two tasks for 20 participants (14 male and 6 female, aged from 22 to 61), including 12 university students, 5 university teachers, and 3 IT company employees. All the participants are skilled with mobile devices but have little AR experience.

For the first task, we would like to evaluate the visual effects of sky replacement videos produced by several of the most competitive methods. Among the well-known semantic segmentation models [18], [21], [29], [63], [64], we chose FastSCNN [64] because it has a potentially real-time performance on mobile devices and a competitive mIoU. For the existing sky replacement methods, we chose Zou [10] because it is dedicated to offline sky replacement and has a publicly available library. Additionally, to validate the effectiveness of the refinement steps of our method, we compared our method with and without the refinement steps. For a fair comparison, we also added the same matting step (Sect. 3.3) to FastSCNN. Zou's method [10] has its own matting step. Next, we captured 20 composite videos with stylized skyboxes for our method (w and w/o) using a smartphone (iPhone 12 Pro) at five common urban locations (campus, beach, neighborhood, park, rooftop) with both sunny and cloudy weather conditions. Since FastSCNN cannot be deployed to the mobile MNN engine (perhaps due to that FASTSCNN uses some operators incompatible with the mobile platform) and the method of Zou cannot perform in real time, we produced offline composite videos using the same original videos for both methods. The input size of the computed mask maps for all the methods was 640 × 360. For each video, we showed the composite results of the four methods (randomly scrambled) to every participant and asked each of them to rate the level of visual pleasingness for each result (10 being the best and 1 the worst). The rating scores for this task are shown in Fig. 14, with four average scores for each video and sorted in descending order by the scores of our method. The figure shows that in most cases, the visual quality of our method is comparable to that of Zou, and better than that of FastSCNN. In a few cases, our method even outperforms these two methods, possibly because our training process focuses on generalization. In addition, in most videos, our full method outperforms the method without the refinement step (especially for videos with low scores of semantic segmentation methods), indicating that the refinement step

TABLE 8
ANOVA and Kruskal-Wallis analyses, as well as Shapiro-Wilk and Levene tests, were used to rate videos produced in various ways. This table compares the four approaches statistically: our method (w) versus our method (w/o), our method (w) versus FastSCNN, and our method (w) versus Zou's method.

	between the Four Methods	between Our Method (w) and Our Method (w/o)	between Our Method (w) and FastSCNN	between Our Method (w) and Zou's Method
Shapiro–Wilk Test	$p = 0.162$	$p = 0.471$	$p = 0.396$	$p = 0.007$
Levene Test	$p = 0.543$	$p = 0.606$	$p = 0.963$	$p = 0.396$
ANOVA with post-hoc tests	$p < 0.001$	$p = 0.025$	$p < 0.001$	N.A.
Kruskal-Wallis H Test	N.A.	N.A.	N.A.	$p = 0.336$

TABLE 9
ANOVA and Kruskal-Wallis analyses, as well as Shapiro-Wilk and Levene tests, were used to calculate user experience scores for several AR apps. This table provides statistical comparisons between the four AR applications: our MobileSky and FCS, our MobileSky and DL3+, and our MobileSky and DL3+MN2.

	between the Four AR Applications	between Our MobileSky and FCS	between Our MobileSky and DL3+	between Our MobileSky and DL3+MN2
Shapiro–Wilk Test	$p < 0.001$	$p < 0.001$	$p < 0.001$	$p = 0.001$
Levene Test	$p = 0.991$	$p = 1.000$	$p = 0.787$	$p = 1.000$
ANOVA with post-hoc tests	N.A.	N.A.	N.A.	N.A.
Kruskal-Wallis H Test	$p \ll 0.001$	$p \ll 0.001$	$p \ll 0.001$	$p \ll 0.001$

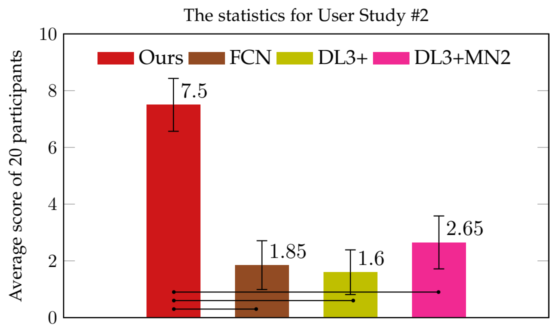


Fig. 16. Average user experience scores of four different AR applications. DL3+ stands for DeeplabV3+ [18] and DL3+MN2 for DeeplabV3+ & MobileNetV2 [29]. Mean values and standard deviations are expressed via bar heights and error bars, respectively. A line connecting two bars indicates a statistically significant difference between the corresponding systems (Kruskal-Wallis H Test, all the connecting lines have $p \ll 0.001$). Since none of the methods except ours can run in real time, the proposed technique significantly outperforms the other three methods in real-time rendering experience.

further improves the quality of the composite results. Fig. 15 and Table 8 give the statistics for this task, as well as the statistical significance of the four techniques. We used ANOVA [66] and post-hoc tests (Tukey's HSD test [67]) on the rating scores. Before running ANOVA, Shapiro-Wilk [68] and Levene [69] tests with Bonferroni correction [70] were used to ensure that the assumptions of normality and equal variance were not violated (p >0.025). We utilized ANOVA and post-hoc tests to demonstrate a significant difference between these methods after noting that the results differed between the four methods, our method (w) and our method (w/o), and our method and FastSCNN (p <0.05). The results of our approach (w) and Zou's method, on the other hand, demonstrate that the assumption of normality was violated, and thus we used a different non-parametric test (Kruskal-Wallis H Test [71]).

In the second task, we expected to evaluate the user experience of the real-time sky replacement method for AR applications. Since not all methods can be deployed on mobile, we only asked the participants to experience FCN [21], DeeplabV3+ [18], MobileNetV2 & DeeplabV3+ [29], and our *MobileSky*. All the methods were deployed through the MNN inference engine. The participants could use these applications as long as they wanted but were unaware of the correspondence between these applications and the underlying methods. Finally, the participants rated their usage experiences (10 being the best and 1 being the worst). The scores for this task are shown in Fig. 16. We used the Kruskal-Wallis H Test to evaluate the data because the results between our MobileSky and the other three AR applications, as well as the results between the four AR applications as a whole, violated the assumption of normalcy. The results are shown in Table 9. Since none of the methods except ours can run in real time, which primarily affects the experience, the scores of our method would be significantly better.

4.8 Limitations and Discussions

Our method has a number of failure cases inherited from the underlying semantic segmentation network and is also limited to the computing power of mobile processors. We leave the limitations to open discussions for future work.

Light Style Mismatching: Currently, the proposed algorithm does not take lighting conditions into account. If the replaced sky has a new lighting direction, the reflections and shadows of the objects in the scene remain unchanged.

In the future, we plan to use fast light estimation methods to estimate light directions, which can be used to align the directional light in the virtual skybox with that of in the real world. Reflections and shadows should be eliminated and re-synthesized to accommodate the changing light direction. To alter the shadow masks, multi-view information or 3D geometry of the scene is required.

Low Accurate Refinement for Very Heterogeneous Skies: Since our color consistency constraint is based on the mean color, it may not work well for heterogeneous skies such as sunsets. To alleviate this issue, we might resort to multi-color feature extraction methods, such as histograms or clustering. However, they are usually not efficient enough to meet our real-time requirements. Fortunately, for sunny, cloudy, or gloomy skies, the use of the mean color is mostly workable. It is because the mean color of these skies is off-white or light blue and is able to capture the key color feature of the skies. This is why we need to refer to the mean color of neighbor pixels since the color of non-sky pixels tends to be closer to the neighbors' mean color than to the sky's mean color.

Failure Replacement for Sky Reflections on Water or Windows: Our method does nothing special for reflected pixels. This is problematic for those reflected pixels above the horizon since the color consistency constraint might mark them as sky regions. For the reflected pixels below the horizon, the position consistency constraint will successfully mark them as non-sky regions.

During the evaluation of our method, we found that its time bottleneck mainly lies in the inference of the segmentation network. We believe that it is challenging to reach 60 FPS on mobile devices at this stage, but our method outperforms other methods in terms of speed. In practical use, 30 FPS already leads to a good visual experience. In addition, we believe that we can achieve higher execution speeds by optimizing our implementation, e.g., reducing floating-point precision, reducing the number of network layers of our method, etc.

5 Conclusion

We have presented a novel system for real-time sky replacement for mobile AR applications. We develop a lightweight, efficient semantic image segmentation network to obtain a binary segmentation result for individual video frames and design sky-aware constraints, including temporal consistency, position consistency, and color consistency, to refine binary mask maps. We show through quantitative validation that our method achieves higher performance than the previous sky replacement methods, achieving an average of around 30 FPS on off-the-shelf smartphones. We also conduct two user studies to show that our results meet good rendering results and accord with human preferences. Our technique can expand capabilities to AR advertising, arts, showing fantasy celestial objects, visually learning about weather phenomena and advanced video-based visual effects.

Acknowledgments

Xinjie Wang was supported by the Shandong Provincial Natural Science Foundation of China (Grant No. ZR2021QF124), China Postdoctoral Science Foundation (Grant No. 2021M703031), and the Open Project Program of the State Key Lab of CAD&CG (Grant No. A2219), Zhejiang University. Xiaogang Jin was supported by the National Natural Science Foundation of China (Grant No. 62036010), and the Key R&D Program of Zhejiang (Grant No. 2022C03126).

References

[1] P. Meister, K. Wang, M. C. Dorneich, E. Winer, L. Brown, and G. Whitehurst, "Augmented reality enhanced thunderstorm learning experiences for general aviation," *Journal of Air Transportation*, vol. 30, no. 4, pp. 113–124, 2022.

[2] J. M. Davila Delgado, L. Oyedele, P. Demian, and T. Beach, "A research agenda for augmented and virtual reality in architecture, engineering and construction," *Advanced Engineering Informatics*, vol. 45, p. 101122, 2020.

[3] N. Xu, B. Price, S. Cohen, and T. Huang, "Deep image matting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2970–2979.

[4] Y. Sun, C.-K. Tang, and Y.-W. Tai, "Semantic image matting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 11 120–11 129.

[5] Y. Liu, J. Xie, Y. Qiao, Y. Tang, and X. Yang, "Prior-induced information alignment for image matting," *IEEE Transactions on Multimedia*, pp. 1–1, 2021.

[6] 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.

[7] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski, "Video matting of complex scenes," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 243–248. [Online]. Available: https://doi.org/10.1145/566570.566572

[8] Y. Sun, G. Wang, Q. Gu, C.-K. Tang, and Y.-W. Tai, "Deep video matting via spatio-temporal alignment and aggregation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 6975–6984.

[9] Y. Tsai, X. Shen, Z. Lin, K. Sunkavalli, and M. Yang, "Sky is not the limit: semantic-aware sky replacement," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 149:1–149:11, 2016. [Online]. Available: https://doi.org/10.1145/2897824.2925942

[10] Z. Zou, "Castle in the sky: Dynamic sky replacement and harmonization in videos," *arXiv preprint arXiv:2010.11800*, 2020.

[11] A. Tran and Y. Le, "Fakeye: Sky augmentation with real-time sky segmentation and texture blending," in *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*, 2020.

[12] 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, 2013.

[13] L. Kaufman, D. Lischinski, and M. Werman, "Content-aware automatic photo enhancement," *Computer Graphics Forum*, vol. 31, no. 8, p. 2528–2540, 2012.

[14] J.-F. Lalonde, S. G. Narasimhan, and A. A. Efros, "What do the sun and the sky tell us about the camera?" *International Journal of Computer Vision*, vol. 88, no. 1, pp. 24–51, 2010.

[15] L. Tao, L. Yuan, and J. Sun, "Skyfinder: Attribute-based sky image search," *ACM Transactions on Graphics*, vol. 28, no. 3, p. 68, 2009.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[17] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[18] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 833–851.

[19] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, "Denseaspp for semantic segmentation in street scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3684–3692.

[20] 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/CVF International Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.

[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.

[23] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3640–3649.

[24] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[25] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2881–2890.

[26] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7794–7803.

[27] S. Cho, M. Cho, T.-y. Chung, H. Lee, and S. Lee, "Crvos: Clue refining network for video object segmentation," *arXiv preprint arXiv:2002.03651*, 2020.

[28] S. W. Oh, J.-Y. Lee, K. Sunkavalli, and S. J. Kim, "Fast video object segmentation by reference-guided mask propagation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7376–7385.

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[30] M. Tan and Q. V. Le, "Mixconv: Mixed depthwise convolutional kernels," *arXiv preprint arXiv:1907.09595*, 2019.

[31] J. Wang and M. F. Cohen, "Image and video matting: A survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 2, p. 97–175, 2007.

[32] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum, "Poisson matting," in *ACM SIGGRAPH 2004 Papers*, ser. SIGGRAPH '04, 2004, p. 315–321.

[33] Q. Chen, D. Li, and C.-K. Tang, "Knn matting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 9, pp. 2175–2188, 2013.

[34] Y. Aksoy, T. O. Aydin, and M. Pollefeys, "Designing effective inter-pixel information flow for natural image matting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2017, pp. 228–236.

[35] Q. Hou and F. Liu, "Context-aware image matting for simultaneous foreground and alpha estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 4129–4138.

[36] H. Lu, Y. Dai, C. Shen, and S. Xu, "Indices matter: Learning to index for deep image matting," in *2Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 3265–3274.

[37] J. Tang, Y. Aksoy, C. Oztireli, M. Gross, and T. O. Aydin, "Learning-based sampling for natural image matting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3050–3058.

[38] D. Hoiem, A. A. Efros, and M. Hebert, "Recovering surface layout from an image," *International Journal of Computer Vision*, vol. 75, no. 1, pp. 151–172, 2007.

[39] C. Liu, J. Yuen, and A. Torralba, "Nonparametric scene parsing via label transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 12, pp. 2368–2382, 2011.

[40] J. Tighe and S. Lazebnik, "Superparsing: Scalable nonparametric image parsing with superpixels," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010, pp. 352–365.

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

[42] Y. Guo, Y. Li, L. Wang, and T. Rosing, "Depthwise convolution is all you need for learning multiple visual domains," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33018368

[43] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017. [Online]. Available: https://arxiv.org/abs/1706.05587

[44] X. Jiang, H. Wang, Y. Chen, Z. Wu, L. Wang, B. Zou, Y. Yang, Z. Cui, Y. Cai, T. Yu *et al.*, "Mnn: A universal and efficient inference engine," *arXiv preprint arXiv:2002.12418*, 2020.

[45] C. Liu, J. Gu, K. Kim, S. G. Narasimhan, and J. Kautz, "Neural rgb®d sensing: Depth and uncertainty from a video camera," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 978–10 987.

[46] J. Shi and Tomasi, "Good features to track," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[47] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*, 2003, pp. 363–370.

[48] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[49] M. Loesdau, S. Chabrier, and A. Gabillon, "Hue and saturation in the rgb color space," in *Image and Signal Processing*, 2014, pp. 203–212.

[50] J. Lobo and J. Dias, "Relative pose calibration between visual and inertial sensors," *The International Journal of Robotics Research*, vol. 26, no. 6, pp. 561–575, 2007. [Online]. Available: https://doi.org/10.1177/0278364907079276

[51] K. Siang Tan and N. A. Mat Isa, "Color image segmentation using histogram thresholding – fuzzy c-means hybrid approach," *Pattern Recognition*, vol. 44, no. 1, pp. 1–15, 2011.

[52] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[53] O. Tobias and R. Seara, "Image segmentation by histogram thresholding using fuzzy sets," *IEEE Transactions on Image Processing*, vol. 11, no. 12, pp. 1457–1465, 2002.

[54] K. He and J. Sun, "Fast guided filter," *arXiv preprint arXiv:1505.00996*, 2015.

[55] S. Liu and T. Zhu, "Structure-guided arbitrary style transfer for artistic image and video," *IEEE Transactions on Multimedia*, pp. 1–1, 2021.

[56] Z. Xu, M. Wilber, C. Fang, A. Hertzmann, and H. Jin, "Learning from multi-domain artistic images for arbitrary style transfer," in *Proceedings of the 8th ACM/Eurographics Expressive Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, ser. Expressive '19. Goslar, DEU: Eurographics Association, 2019, p. 21–31. [Online]. Available: https://doi.org/10.2312/exp.20191073

[57] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[58] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1209–1218.

[59] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2014, pp. 891–898.

[60] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5122–5130.

[61] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, 2010, pp. 177–186.

[62] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.

[63] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[64] R. P. K. Poudel, S. Liwicki, and R. Cipolla, "Fast-scnn: Fast semantic segmentation network," *arXiv preprint arXiv:1902.04502*, 2019.

[65] 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)*, 2016, pp. 1–6.

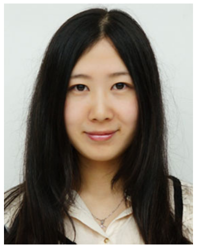
[66] E. R. Girden, *ANOVA: Repeated measures*. Sage, 1992, no. 84.

[67] J. W. Tukey, "Comparing individual means in the analysis of variance," *Biometrics*, pp. 99–114, 1949.

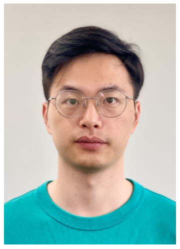
[68] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[69] H. Levene, "Robust tests for equality of variances," *Contributions to probability and statistics. Essays in honor of Harold Hotelling*, pp. 279–292, 1961.

[70] C. E. Bonferroni, "Il calcolo delle assicurazioni su gruppi di teste," *Studi in onore del professore salvatore ortu carboni*, pp. 13–60, 1935.
[71] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.


Xinjie Wang is a Lecturer of Computer Science at Ocean University of China. She received the BSc degree in computer science from Wuhan University, and the PhD degree in computer science from the State Key Laboratory of CAD&CG, Zhejiang University in 2015. Her main research interests include augmented reality, scientific visualization, and computer animation.

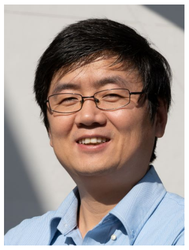
Qingxuan Lv received his BS in Computer Science and Technology from the Shanxi University of Finance and Economics in 2018. He received his master's degree in Computer Science and Technology from the Ocean University of China (OUC) in 2021. He is currently a candidate of a doctor' degree at the VisionLab OUC. His research interests include computer vision and deep learning. Specifically, he is interested in domain adaptation and semantic segmentation.


Guo Chen is a master student of the State Key Lab of CAD&CG, Zhejiang University, China. He received his bachelor's degree in computer science and technology from Shandong University in 2021. His research interests include machine learning, computer vision, and real-time semantic segmentation and detection.

Jing Zhang is a master student of the State Key Lab of CAD&CG, Zhejiang University, China. She received her bachelor's degree in digital media technology from Zhejiang University in 2021. Her research interests include virtual reality, augmented reality, and computer graphics.

Zhiqiang Wei is currently a professor with the Ocean University of China, Qingdao, China. He received the PhD degree from Tsinghua University, Beijing, China, in 2001. His current research interests are in the fields of intelligent information processing, social media and big data analytics.

Junyu Dong (Member, IEEE) received the BSc and MSc degrees in applied mathematics from the Department of Applied Mathematics, Ocean University of China, Qingdao, China, in 1993 and 1999, respectively, and the PhD degree in image processing from the Department of Computer Science, Heriot-Watt University, Edinburgh, U.K., in November 2003. He is currently a Professor and the Head of the Department of Computer Science and Technology, Ocean University of China. His research interests include machine learning, big data, computer vision, and underwater image processing.


Hongbo Fu is a Professor in the School of Creative Media, City University of Hong Kong. Before joining CityU, he had postdoctoral research training at the Imager Lab, University of British Columbia, Canada, and the Department of Computer Graphics, Max-Planck-Institut Informatik, Germany. He received a PhD degree in computer science from the Hong Kong University of Science and Technology in 2007 and a BS degree in information sciences from Peking University, China, in 2002. His primary research interests fall in the fields of computer graphics and human-computer interaction. He has served as an associate editor of The Visual Computer, Computers & Graphics, and Computer Graphics Forum.

Zhipeng Zhu received the BSc degree in Electronic and Information Engineering and the MSc degree in Information and Communication Engineering from University of Electronic Science and Technology of China, in 2017 and 2020, respectively. He joined OPPO in 2020. His current research interests include image quality assessment, image and video editing, and lightweight neural network.

Jingxin Liu received the master and PhD degree in EEG based affective computing from Brunel University London, UK, in 2014, 2019, respectively. He is a leader of on-device intelligence technology development of software engineering system, OPPO. His current research interests include semantic CV, VR/AR, video content understanding, model compression, NAS, on-device inference acceleration, multimodality data analysis, multi-modal feature fusion and multimodal pretrained model.

Xiaogang Jin (Member, IEEE) received the BSc degree in computer science and the MSc and PhD degrees in applied mathematics from Zhejiang University, P. R. China, in 1989, 1992, and 1995, respectively. He is a professor with the State Key Laboratory of CAD&CG, Zhejiang University. He was the recipient of the ACM Recognition of Service Award in 2015 and the Best Paper Awards from CASA 2017 and CASA 2018. His current research interests include virtual reality, cloth animation, digital human, and computer games. He is a member of the IEEE and ACM.