Supplemental Text for: Sky Optimization: Semantically aware image processing of skies in low-light photography

Orly Liba Longqi Cai Yun-Ta Tsai Elad Eban
Yair Movshovitz-Attias Yael Pritch Huizhong Chen Jonathan T. Barron

Google Research

1. The modified guided filter algorithm

In this section we describe our modified guided-filter-based mask refinement method. Pseudo code for our modified guided filter is as follows:

$$
\begin{aligned}
&\text{modified\_guided\_filter}(I, P, C, s, \epsilon_\ell, \epsilon_c):\\
&\quad I_\downarrow = \text{weighted\_ds}\,(I, C, s)\\
&\quad P_\downarrow = \text{weighted\_ds}\,(P, C, s)\\
&\quad \Sigma_\downarrow = \text{weighted\_ds}\,(I \otimes I, C, s) - I_\downarrow \otimes I_\downarrow\\
&\quad \sigma_\downarrow = \text{weighted\_ds}\,(I \circ P, C, s) - I_\downarrow \circ P_\downarrow\\
&\quad \Sigma_\downarrow = \Sigma_\downarrow + \begin{bmatrix} \epsilon_\ell^2 & & \\ & \epsilon_c^2 & \\ & & \epsilon_c^2 \end{bmatrix}\\
&\quad A_\downarrow = \text{solve\_image\_ldl3}(\Sigma_\downarrow, \sigma_\downarrow)\\
&\quad b_\downarrow = P_\downarrow - A_\downarrow \cdot I_\downarrow\\
&\quad A = \text{smooth\_us}\,(A_\downarrow, s)\\
&\quad b = \text{smooth\_us}\,(b_\downarrow, s)\\
&\quad Y = A \cdot I + b
\end{aligned}
\tag{1}
$$

Inputs to the filter are: a 3-channel reference image I (assumed to be in YUV), the quantities to be filtered, P (in our case, the sky mask), a confidence map C, and hyper-parameters: s, the downsampling factor, and ϵ_ℓ, and ϵ_c, the regularization factors for the luma and chroma, respectively. The output of the filter is Y, a mask that resembles P where C is large, and adheres to the edges in I. Regarding notation, $\circ$ is the Hadamard product (where 1-channel images are "broadcasted" to match the dimensions of images with more channels), and $\cdot$ is a dot product (Hadamard product that is then summed over channels). The outer product of two images $A = X \otimes Y$ is defined as taking two 3-channel images X and Y and producing a 6-channel image A representing the upper-triangular portion of the outer product of each pixel of X and Y:

$$
\begin{aligned}
A_{1,1} = X_1 \circ Y_1, \quad A_{1,2} = X_1 \circ Y_2, \quad A_{1,3} = X_1 \circ Y_3\\
A_{2,2} = X_2 \circ Y_2, \quad A_{2,3} = X_2 \circ Y_3\\
A_{3,3} = X_3 \circ Y_3
\end{aligned}
\tag{2}
$$

Our weighted downsample, weighted_ds, is simply a standard bilinear downsample operator applied using homogeneous coordinates:

$$
\text{weighted\_ds}\,(X, C, s) = \frac{\text{ds}(X \circ C, s)}{\text{ds}(C, s)} \tag{3}
$$

where division is element-wise, and $\text{ds}(\cdot, s)$ is bilinear downsampling according to a spatial bandwidth s. $\text{smooth\_us}\,(\cdot, s)$ is the smooth upsampling procedure, described next. $\text{solve\_image\_ldl3}(A, b)$ is an LDL-decomposition based linear solver designed to operate on 3-channel images, from [4]. For completeness, this algorithm is reproduced below.

The traditional guided filter uses a box filter to compute local expectations of various quantities. Because the box filter is applied twice, and because the convolution of two box filters is a triangle-shaped ("tent") filter, the output of the traditional guided filter tends to contain triangle-shaped artifacts. Though there exist fast techniques for applying smoother blur kernels than box filters [5], these techniques are still significantly more expensive than box filters, and do not reduce the number of linear systems to be solved. In our algorithm, smooth upsampling is achieved by applying the triangle-shaped convolution kernels consecutively (in our case, in 3 steps), which effectively changes the shape of the interpolation kernel to be smoother, and significantly reduces upsampling artifacts. For example, for a downsampling factor $s = 64$, instead of upsampling with a single kernel with a support of 64×64, we use triangle kernels with a support of 4×4 three times, one after the other. We chose this method of linear upsampling rather than a more advanced method owing to its separability and efficient implementation. The effect of this smooth upsampling can be seen in Figure 1.

Note that our modified guided filter formulation degrades naturally to the traditional formulation of the guided filter if 1) $\text{weighted\_ds}(\cdot, C, s)$ and $\text{smooth\_us}(\cdot, s)$ are both replaced with a box filter, 2) the reference image I is RGB and $\epsilon_\ell = \epsilon_c$, and 3) $\text{solve\_image\_ldl3}(\cdot, \cdot)$ is replaced with matrix inversion and a matrix multiply.

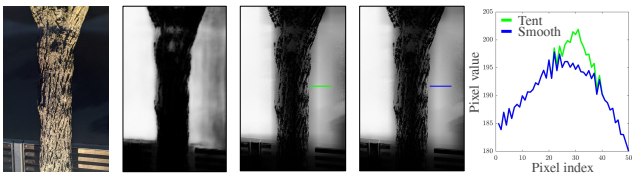
Our approach of accelerating part of the guided filter through the use of spatial downsampling is superficially similar with the "fast guided filter" [2], which also yields an acceleration from $\mathcal{O}(n)$ to $\mathcal{O}(n/s^2)$ for an intermediate step of the filter. This is accomplished by simply subsampling the input mask before computing the affine filter coefficients, which are then applied to the full resolution mask. Though fast, this approach ignores the vast majority of the input mask, and thereby assumes that the input to the filter is very smooth and regular. This does not hold in our use case: for example, if we had a single high-confidence pixel surrounded by many low-confidence pixels, we would require a guarantee that this single pixel's value would propagate to all nearby low-confidence pixels in the output, and a subsampling approach will not guarantee this (and worse, will cause the output of the model to vary significantly depending on whether or not the single high-confidence pixel happens to lie at one of the few spatial locations that is subsampled). In contrast, our approach ignores none of the input mask, is completely deterministic, and still yields the same asymptotic acceleration of the filter's internal linear solver step.

1.1. LDL-decomposition based linear solver

In the refinement algorithm, $\text{solve\_image\_ldl3}(A, b)$ is an LDL-decomposition based linear solver designed to operate on 3-channel images, from [4]. For completeness, we reproduce that algorithm here:

$$
\begin{aligned}
&\text{solve\_image\_ldl3}(A, b): \\
&\quad d_1 = A_{1,1} \\
&\quad L_{1,2} = A_{1,2}/d_1 \\
&\quad d_2 = A_{2,2} - L_{1,2} \circ A_{1,2} \\
&\quad L_{1,3} = A_{1,3}/d_1 \\
&\quad L_{2,3} = (A_{2,3} - L_{1,3} \circ A_{1,2})/d_2 \\
&\quad d_3 = A_{3,3} - L_{1,3} \circ A_{1,3} - L_{2,3} \circ L_{2,3} \circ d_2 \\
&\quad y_1 = b_1 \\
&\quad y_2 = b_2 - L_{1,2} \circ y_1 \\
&\quad y_3 = b_3 - L_{1,3} \circ y_1 - L_{2,3} \circ y_2 \\
&\quad x_3 = y_3/d_3 \\
&\quad x_2 = y_2/d_2 - L_{2,3} \circ x_3 \\
&\quad x_1 = y_1/d_1 - L_{1,2} \circ x_2 - L_{1,3} \circ x_3
\end{aligned} \tag{4}
$$

Where the inputs to this function are a 6-channel image A and a 3-channel image b, with channels in A corresponding to the upper triangular part of a 3×3 matrix. The output of this function is a 3-channel image x where for each pixel i in the input linear system, $x(i) = A(i) \backslash b(i)$ using an LDL decomposition.



(a) Input image (b) Inferred mask (c) GF + tent upsampling (d) GF + smooth upsampling e) Pixel values along line

Figure 1: Using bilinear upsampling within a guided filter (GF) results in noticeable triangle-shaped artifacts (c, e), while our three-step upsampling results avoids such artifacts (d, e).

2. Density estimation algorithm

The density estimation algorithm is used to inpaint unlabeled pixels and partially automate the process of annotating sky masks. The probability that an unlabeled pixel belongs to the "sky" pixels is described in Equation 5, in which i indicates an "undetermined" pixel and j indicates a "sky" pixel:

$$p_i = \frac{1}{|\{\text{sky}\}|} \sum_{j \in \{\text{sky}\}} K(I_i, I_j) \tag{5}$$

$$K(I_i, I_j) = \frac{1}{(2\pi\sigma^2)^{3/2}} \exp\left(-\frac{\sum_c (I_i^c - I_j^c)^2}{2\sigma^2}\right) \tag{6}$$

$K(\cdot)$ is a multivariate Gaussian kernel with a Euclidean distance between the RGB values of pixels, assuming a diagonal covariance matrix (c indicates the color channel). We use $\sigma = 0.01$ as the kernel's standard deviation. In practice, to reduce computation time we sample 1024 sky pixels uniformly at random to compute these probabilities.

3. Model optimization

Table 1 shows a comparison of the size, latency, and IoU scores of the original model and of the two MorphNet steps. At the end of model optimization we arrive at a model that is more accurate and 27% smaller.

	Original UNet	MorphNet Shrink	MorphNet Expand
Model Size (MB)	5.1	1.7	3.7
$IoU_{0.5} \uparrow$	0.9186	0.8895	**0.9237**
Latency (ms)	430	290	443
Size Reduction		66%	27%

Table 1: The model performance before and after optimization with MorphNet and weight quantization to float-16. Inference latency is measured on a 256×256 image on a mobile CPU (Qualcomm Snapdragon 845) with 4 threads. Evaluation is performed on our internal dataset, and therefore these results cannot be directly compared with those in Table 1 of the main paper.

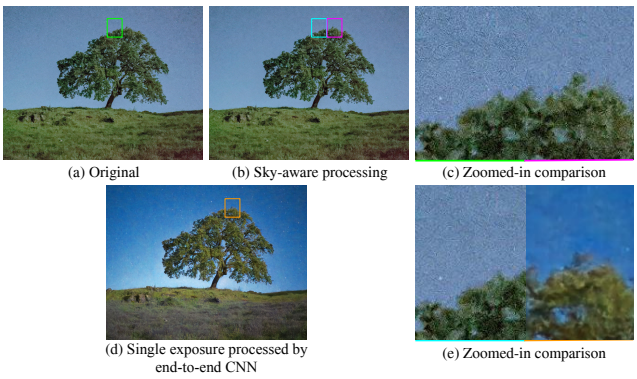
4. Sky denoise and comparison to CNN

In this section we show an example of sky-aware noise reduction in a low-light image and compare it to an end-to-end convolutional neural network (CNN) [1] that produces a low-light image from a single raw frame. In Figure 2a-c, our system is able to reduce the noise in the skies while preserving the details of the stars and the tree. Figure 2d-e shows a comparison of our result to the result produced by the CNN of [1]. In this comparison, we used a raw frame captured with a similar camera model as was used for training the network in [1] (Sony α7S II). Because our white balancing algorithm was not calibrated for the Sony camera, and since color variation can affect the perception of details, we used Photoshop's automatic tool to color-match our results to the results produced by the CNN (as detailed in [3]). Although the sky in the image produced by the CNN has less fine-grained noise compared to our result, our result has less low frequency noise in the skies and preserves more details of the foreground.

5. Refining the annotations of the ADE20K dataset

In the Experimental Results Section, we chose the ADE20K dataset [6] as our baseline dataset. We selected all the images that have the "sky" label and added to them 10% random images without the skies. We do this for both the validations and training parts of the dataset so that in total we have 9187 images from the training set and 885 images from the validation set. We then refine this dataset with two different methods: 1) using the guided filter only (annotated as ADE20K+GF), and 2) using density estimation and the guided filter (annotated as ADE20K+DE+GF).

Creating ADE20K+GF is straightforward: we input the annotated raw masks and the ADE20K images into the weighted guided filter algorithm (described in Section 3.1) with a confidence map of ones in each pixel. The parameters



(a) Original (b) Sky-aware processing (c) Zoomed-in comparison

(d) Single exposure processed by end-to-end CNN (e) Zoomed-in comparison

Figure 2: Comparison between a low-light image, with and without sky-aware processing, and the end-to-end trained CNN described in [1]. A raw frame was captured with a Sony α7S II, the same camera that was used to train the CNN, with an exposure time of 0.1 s. a) The original result. b) The result in (a) with sky-aware processing. (c) Zoomed-in regions of the original image (green) and sky-processed image (magenta). d) The result from the CNN. e) Zoomed-in regions of our result (cyan) and the result of the CNN (orange).

of the guided filter are $s = 48$ and, $\epsilon_\ell = \epsilon_c = 0.01$. In order to create ADE20K+DE+GF we had to create a heuristic for the "undetermined" label, in which we would apply inpainting using the density estimation algorithm described in Section 3.2.3. We used the following method: a) Run a Laplacian edge filter on the raw ADE20K sky masks to find the sky boundaries; b) Dilate the boundaries with an ellipse kernel with a radius of 4 pixels to generate the "undetermined" label; c) Add areas labeled as trees in ADE20K to our "undertermined" region. We do this because often the sky can be seen through tree branches and we have found that inpainting the entire tree to be more accurate than using raw annotations. Then, we inpaint the "undetermined" region using density estimation. For these experiments, we used a probability threshold of $p_c = 0.97$. We then create the confidence map with values: $c_{det} = 0.8$ (for the "sky" and "not sky" original raw labels), $c_{inpaint} = 0.6$ is for pixels inpainted as skies and $c_{undet} = 0.4$ for the remaining pixels. Finally, we apply the weighted guided filter, with parameters: $s = 16$ and, $\epsilon_\ell = \epsilon_c = 0.01$, and inputs: the inpainted annotations, the new confidence maps, and the original ADE20K images. Following the guided filter, in order to drive the intermediate mask values towards the edges of the range: 0 and 1, we applied sharpening to the mask, using Equation 7.

$$S(x) = \frac{h(t_s(x - 1/2)) - h(-t_s/2)}{h(t_s/2) - h(-t_s/2)} \tag{7}$$

In which x is the value of the sky mask, normalized to a range of $[0, 1]$, $h(x) = 1/(1 + exp(-x))$ is the sigmoid

	Raw	GF	DE+GF
Raw versus GF	629 (30%)	1471 (70%)	
Raw versus DE+GF	365 (17.4%)		1735 (82.6%)
GF versus DE+GF		476 (22.7%)	1624 (77.3%)
Total	994 (15.8%)	1947 (30.9%)	3359 (53.3%)

Table 2: The results of a user study comparing raw ground truth sky masks and refined sky masks from the ADE20K dataset.

(a) Original image (b) Annotation (c) GF refinement (d) DE + GF refinement

Figure 3: Example images from the user study. In these images, the sky was darkened using either the original sky mask annotation of the ADE20K dataset or refined sky annotations. The user study evaluated the user's preference for images (b), (c), or (d). Sky darkening was applied to emphasize the shape of the sky mask, and is not an indication of our proposed darkening scheme which is described in Section 3.4 of the main paper

.

function, and t_s is the sharpness factor. We found that a sharpness factor, $t_s = 15$ produces visually accurate masks.

6. User study results

The goal of the user study is to show that raw binary annotations, demonstrated here using the ADE20K dataset [6], are too rough for computational photography applications. Because refined masks are more suitable for sky editing, a quantitative evaluation of mask accuracy should be performed using refined annotations as the ground-truth. The subjects were asked to rate which masks are more accurate: the raw annotations, the annotations refined only with the guided filter (GF) and the annotations refined with both density estimation and guided filter (DE+GF). Example images are shown in Figure 3. The annotations were evaluated one pair at a time. The images for the study were 100 randomly picked images with skies from the ADE20K validation set. We had 21 participants take the study. The results are in Table 2. From the results we see that the users preferred masks refined with both DE and GF over only GF and any refinement was preferred over the raw annotations.

7. Evaluation metrics

Here we define the segmentation evaluation metrics used in our experiments. We use two metrics that take as input the binarized versions (at 0.5) of our ground-truth alpha mattes and of our predictions: mean intersection-over-union (mIOU) and misclassification rate (MCR):

$$\begin{aligned} \text{mIOU}_{0.5} &= \sum \frac{TP}{TP+FP+FN} \\ \text{MCR}_{0.5} &= \sum \frac{FP+FN}{M} \end{aligned} \tag{8}$$

where TP, FP and FN are the true positive, false positive, and false negative, respectively, and M is the number of pixels. We also present a series of non-binarized error metrics: root mean square error (RMSE), mean absolute error (MAE), boundary loss (BL), and Jensen-Shannon Divergence (JSD):

$$\begin{aligned} \text{RMSE}(X,Y) &= \sqrt{\frac{1}{M}\sum_i (X_i - Y_i)^2} \\ \text{MAE}(X,Y) &= \frac{1}{M}\sum_i |X_i - Y_i| \\ \text{BL}(X,Y) &= \sqrt{\frac{1}{M}\sum_i (\nabla X_i - \nabla Y_i)^2} \\ \text{JSD}(X \parallel Y) &= \frac{1}{M}\sum_i \left(\frac{1}{2}\,\text{KL}\left(X_i \parallel \frac{X_i+Y_i}{2}\right) + \right. \\ &\qquad \left. \frac{1}{2}\,\text{KL}\left(Y_i \parallel \frac{X_i+Y_i}{2}\right)\right) \end{aligned} \tag{9}$$

Where X and Y are the predicted and true alpha mattes, ∇ indicates the spatial gradient of an image, and $\text{KL}(\cdot)$ is the KL divergence between two Bernoulli distributions (the true and predicted alpha matte at each pixel).

8. Examples of the sky effects

The effects shown in Figure 4 and Figure 5 demonstrate the sky-aware processing steps performed by our pipeline. As shown in these figures, our procedure is able to accurately segment the sky and automatically improve its appearance for a variety of scenes. The effects are relatively subtle, as we have calibrated them to maintain the reliability of the scene and only alleviate the challenges of low-light imaging, without changing the image too much or potentially introducing new artifacts.

References

[1] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. *CVPR*, 2018. 3

[2] Kaiming He and Jian Sun. Fast guided filter. *CoRR*, abs/1505.00996, 2015. 2

(a) No sky effects (b) Sky denoise (c) Adding sky darkening (d) Adding sky white balance

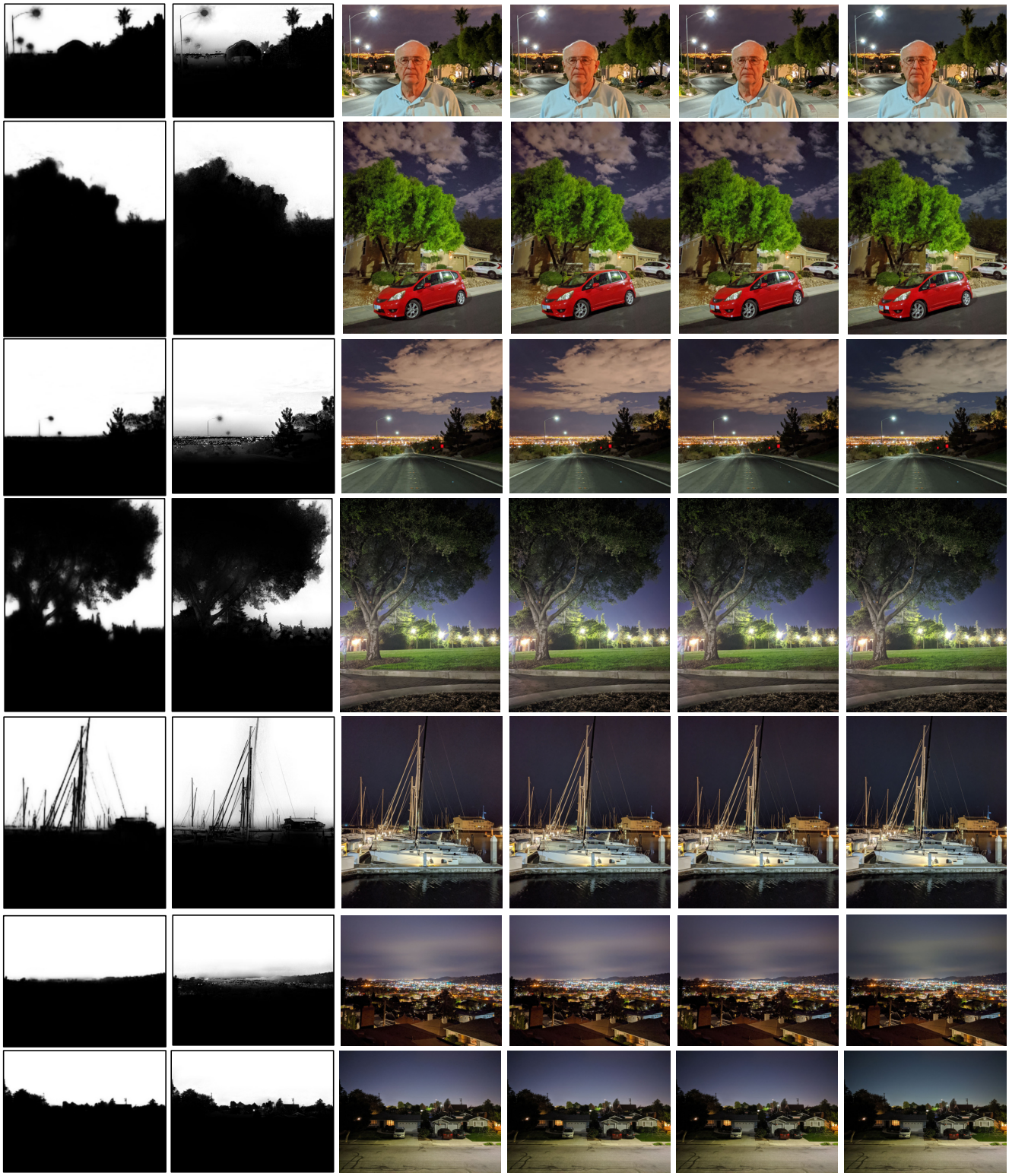
Figure 4: The sky effects we propose, applied in sequence. Note that these effects are independent and do not rely on one another. a) The original image. b) Sky-specific noise reduction. c) Tonemapping applied to darken the sky. d) sky-inferred auto white balance gains applied according to the sky mask.

[3] Orly Liba, Kiran Murthy, Yun-Ta Tsai, Tim Brooks, Tianfan Xue, Nikhil Karnad, Qiurui He, Jonathan T Barron, Dillon Sharlet, Ryan Geiss, et al. Handheld mobile photography in very low light. *SIGGRAPH Asia*, 2019. 3

[4] Julien Valentin, Adarsh Kowdle, Jonathan T. Barron, Neal Wadhwa, Max Dzitsiuk, Michael Schoenberg, Vivek Verma, Ambrus Csaszar, Eric Turner, Ivan Dryanovski, Joao Afonso, Jose Pascoal, Konstantine Tsotsos, Mira Leung, Mirko Schmidt, Onur Guleryuz, Sameh Khamis, Vladimir Tankovitch, Sean Fanello, Shahram Izadi, and Christoph Rhemann. Depth from motion for smartphone AR. *SIGGRAPH Asia*, 2018. 1, 2

[5] Ian T. Young and Lucas J. van Vliet. Recursive implementation of the gaussian filter. *Signal Processing*, 1995. 1

[6] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. *CVPR*, 2017. 3, 4



(a) Inferred sky mask | (b) Refined mask | (c) No sky effects | (d) Sky denoise | (e) Adding sky darkening | (f) Adding sky white balance

Figure 5: The sky affects applied one after the other. a) The inferred sky mask. b) The upsampled and refined mask, using our modified weighted guided filter. c) The original image, without sky effects. d) Sky-specific noise reduction is applied to the sky. The readers are encouraged to zoom-in to see the difference in noise characteristics. e) Tonemapping is applied to darken the sky. f) The sky-inferred auto white balance gains are applied to the sky pixels.