

An Efficient Sky Detection Algorithm Based on Hybrid Probability Model

Chi-Wei Wang, Jian-Jiun Ding *, and Po-Jen Chen

Graduate Institute of Communication Engineering, National Taiwan University

E-mail: r02942055@ntu.edu.tw, jjding@ntu.edu.tw, r03942106@ntu.edu.tw

TEL: +886-2-33669652

Abstract— Sky is one of the most significant subject matter commonly seen in outdoor photos. We propose a highly efficient sky detection algorithm. First, we detect a rough sky-ground boundary. Then, we calculate the parameters related to appearance of sky. Finally, we use these parameters to construct a hybrid probability model that indicates how possible a pixel belongs to sky. Moreover, an image processing library with parallel processing techniques is embedded in the proposed algorithm. The proposed algorithm has both high accuracy and high efficiency and can process a video image in real time. If the input is a VGA size image, the computation time of the proposed algorithm is less than 35ms when using a common desktop.

I. INTRODUCTION

Sky is highly visually important for human beings. Also it often appears in video sequence and photos. A quick sky-detection system can be used for lots of applications. For image content analysis, it can be a strong feature in indoor/outdoor classification problem [1][2]. For another example, it can be used for content-based image manipulation. We can get some color or noise information from sky parts, and then decide how to apply, say, some kind of image quality enhancement algorithms or adjust parameters for these algorithms[3]. Furthermore, sky-detection system can also serve as a part of background detector for 3-D depth-map generation.

In general, there are two categories of sky-detection algorithm. One kind is to find the sky-ground boundary vector and another kind is to find and label all pixels belonging to sky. Algorithms aimed for sky-ground boundary vector often have better accuracy and generality than those aimed for labeling all sky-pixels. It is because these pixel-based algorithms usually need a pre-defined color center for sky [4][5]. So we can only find blue-like sky if we define color-center as blue, orange-like sky if we define color-center as orange, and so on. But they have a desired property – they can detect sky with any shape. On the other hand, algorithms for finding sky-ground boundary vector are often independent of color. Their main concern is texture and gradient[6]. Specifically, they design a cost function varying with texture or gradient and try to optimize it. However, although good accuracy and generality, they have an obvious drawback – they cannot handle sky with any shape. They can only find a vector from one endpoint to another endpoint. Fig. 1 shows the difference between these two categories of algorithms. It is obvious that one cannot detect sky-pixels below cloud while another can if a suitable pre-defined color center is provided. To overcome this drawback, we propose a method

that combine advantages of two categories of algorithms and avoid their disadvantages. We redesign algorithms that detect sky-ground boundary vector and let this vector as input of pixel-based algorithms. We focus on better performance and less time-consumption. In our implementation and testing platform, real-time processing for 480p video is possible.

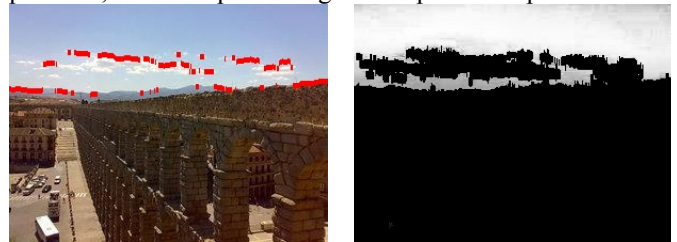


Fig. 1 Difference between two categories of algorithm.

This paper is organized as follows. In Section II proposed method is explained. Simulations will be shown in Section III. In Section IV, a conclusion will be given.

II. PROPOSED METHOD

The framework of the proposed sky-detection system can be split into two parts: estimate sky color and detect sky region. We introduce what kinds of algorithms are used for obtaining these two parts in this section. Some key components are explained first and algorithm details follows.

A. Estimation of Sky Color

Inspired by [6], we use an objective function to detect a rough sky-ground boundary for estimating sky color which serve as input of probability models, which will be introduced latter.

We can assume that sky-ground boundary has quite strong gradient – or our human beings would not perceive sky-ground boundary as a boundary. Some sharp transition in color occurs so our visual system consider it as certain kind of boundary. We calculate gradients as candidates for sky-ground boundary[6].

After converting images into gray level, we use the 5x5 Sobel operators to calculate gradient images, as Fig. 2. Results of the two directional Sobel operators are combined by (1)

$$G = \sqrt{G_x^2 + G_y^2} \quad (1)$$

where G is the gradient image and G_x , G_y are results of the x - and y -directional Sobel operators.

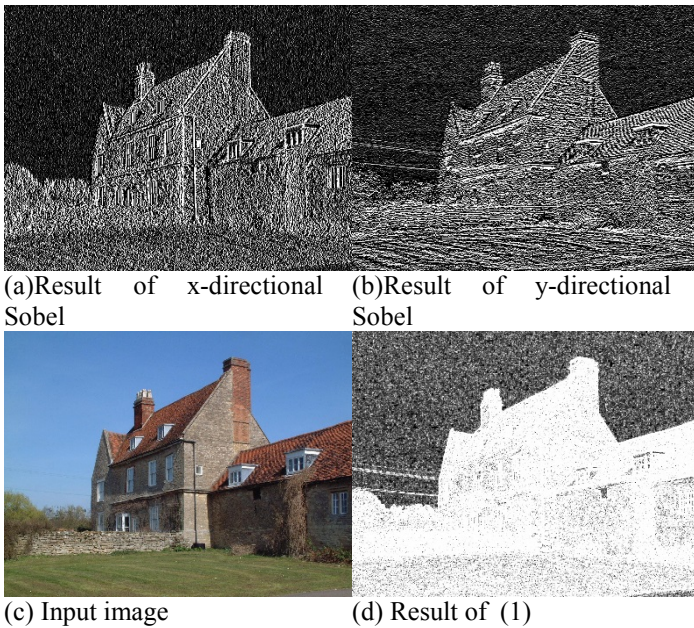


Fig. 2 Gradient Image obtained by Sobel operators.

It is quite obvious in Fig. 2 that gradient values are small at sky region (look gray), and strong at non-sky region (look bright).

Now we are going to use the object function (2) to find a best sky-ground boundary within our searching space.

$$J = \frac{1}{\max(\gamma |\Sigma_s| + |\Sigma_g|, small_value)} \quad (2)$$

where Σ_s , Σ_y are covariance matrices of sky and ground parts respectively. They are defined as (3) and (4), both 3x3 matrices.

$$\Sigma_s = \frac{1}{N_s} \sum_{x=1}^{x=\lfloor W/\alpha \rfloor} \sum_{y=1}^{y=b(x)} (\mathbf{I}^s(\alpha x, y) - \boldsymbol{\mu}^s)(\mathbf{I}^s(\alpha x, y) - \boldsymbol{\mu}^s)^T, \quad (3)$$

$$\Sigma_g = \frac{1}{N_g} \sum_{x=1}^{x=\lfloor W/\alpha \rfloor} \sum_{y=b(x)+1}^{y=H} (\mathbf{I}^g(\alpha x, y) - \boldsymbol{\mu}^g)(\mathbf{I}^g(\alpha x, y) - \boldsymbol{\mu}^g)^T \quad (4)$$

where α is an integer indicating step between two columns taken into account. In our work we pick up $\alpha = 3$. That is, only 1st, 4th, 7th, 10th, ... are considered while calculating covariance matrices. Also, N_s and N_g are the number of pixels belong to sky and ground respectively. $\boldsymbol{\mu}^s$ and $\boldsymbol{\mu}^g$ are vectors that represent mean values of RGB values in the sky and ground region respectively. $|\cdot|$ denotes the determinant of matrix. *small values* are introduced in order to deal with circumstances under which video frames lose color information or images are grayscale. The parameter γ in (2) emphasizes sky-region. It is experimentally decided. We chooses $\gamma = 2$. As $b(x)$, it is sky-ground boundary in our iterating procedure of optimizing (2).

From observation of Fig. 2, sky-ground boundaries appear as strong gradient. Thus, a parameter t can be defined as a threshold which helps to split sky and ground. Specifically, we search gradient image from top to down to find where

gradient value is larger than t . And in each iteration, t is calculated as (5).

$$t = threshold_{min} + \frac{threshold_{max} - threshold_{min}}{iterations - 1} \cdot (n - 1) \quad (5)$$

where *iterations* is defined by user that how many iterations will be applied and n is n^{th} iteration. Here pseudo-code for finding rough sky-ground boundaries is provided.

Function 1 – split sky and ground by parameter t .

```

b(x) = Split_Sky_Ground(gradient(y, x), t, b_old(x))
for i = 1 to W
  for j = b_old(x) to 1
    if gradient(j, i) > t
      b(i) = j
      break
  
```

Function 2 – optimize object function

```

b_opt(x) = Find_Opt_Boundary(gradient(y, x), threshold_min,
threshold_max, iterations)
b_opt(x) = {0}, b_tmp(x) = {0}, J_max = 0, b_old = {0}
for n = 1 to iterations
  
```

$$t = threshold_{min} + \frac{threshold_{max} - threshold_{min}}{iterations - 1} \cdot x(n-1)$$

```

  b_tmp = Split_Sky_Ground(gradient(y, x), t, b_old)
  b_old = b_tmp
  J = objection_function(b_tmp)
  if J > J_max
    J_max = J
    b_opt = b_tmp
  
```



Fig. 3 Results of **Function 1** and **Function 2**

B. Probability Model


Now we can obtain sky boundary results as Fig. 3. Next step is to sum up a color center and standard deviation for corresponding image. For example, a sky-ground boundary is calculated based on **Function 1** and **Function 2**. Color center of sky is obtained by averaging all pixels belonging to sky, that is, all pixels above sky-ground boundary. Standard deviation is also calculated from these sky-pixels.

Here a notable improvement can be found: we only need statistic result. So it is possible to reduce original image size first. Then, we perform **Function 1** and **Function 2** and calculate the color center and the standard deviation.

The image pyramid is a widely used technique that can help reduce computational cost, especially for an algorithm which do not care about image size during intermediate processing – that is our case. Strictly speaking, an image pyramid is a collection of images all arising from a single original image. They are downsampled until some desired stopping point is reached.

In our implementation, we set QVGA 320x240 resolution as stopping point. The smallest image among image pyramid is chosen. We compare statistic results calculated from original image and the smallest image among image pyramid as Table I. A dramatic boost in speed occurs while little sacrificing accuracy.

Table I Comparison of color center and standard deviation between downsampling or not.

	
Input image with size [699, 524] for comparing statistic results	
original size [699, 524]	QVGA size image
Color center of RGB = [204.508, 225.807, 243.914]	Color center of RGB = [203.234, 225.065, 243.891]
Standard deviation of RGB = [28.863, 17.353, 7.432]	Standard deviation of RGB = [28.684, 17.464, 7.429]
Computing time: 0.2673 sec	Computing time: 0.0207 sec

Now we use color center and standard deviation in probability model. Three kind of probability models are chosen [3]. First one is color-model, (6).

$$P_{color} = \exp\left(-1 \cdot \left(\left(\frac{R-r}{\beta\sigma_r}\right)^2 + \left(\frac{G-g}{\beta\sigma_g}\right)^2 + \left(\frac{B-b}{\beta\sigma_b}\right)^2\right)\right) \quad (6)$$

where β is a user-defined parameter, r, g, b are color center and $\sigma_r, \sigma_g, \sigma_b$ are standard deviation with respect to three channels of color. R, G, B are three channels color of original image. We choose $\beta=5$ in implementation.

Second probability model is gradient-value probability model, as (7).

$$P_{grad} = \exp(-([G]_{T_{min}}^\infty)^2) \quad (7)$$

where G is obtained by (1), but this time we use the 3x3 Sobel operators instead of 5x5. T_{min} is a threshold that filters out some small gradient values caused by noise or ignorable changes. $[\cdot]_a^b$ is a clipping function which replaces these value smaller than a with a , and those bigger than b with b . And for filtering out area too small to be considered as texture, dilation is used. Note that dilation should be applied before $[\cdot]_{T_{min}}^\infty$ operator in (7).

Third probability model is about vertical position [3]. It is assumed that sky is above ground in a single image.

$$P_{position} = \exp\left(-\left(\frac{r}{N_h}\right)^2\right) \quad (8)$$

where r is vertical position of pixels(the first row in the top of image is defined as 0) and N_h is the height of image.

Final probability map are obtained as (9). And for some refinements, close operation is applied. That is, we apply dilate and then erode on P_{sky} .

$$P_{sky} = P_{color} \cdot P_{grad} \cdot P_{vert} \quad (9)$$

III. SIMULATIONS

Comparisons among proposed method and other algorithms are shown here. The testing platform is a common desktop PC with Intel i5 CPU and 8GB RAM.

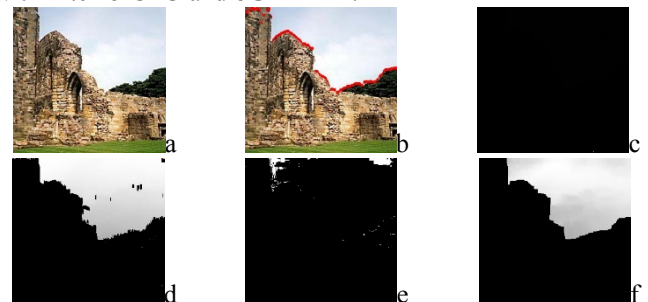


Fig. 4 Results of corresponding reference work. {a, b, c, d, e, f} = {input, [6], [3], [9], [4], proposed method}

Note that [3] and [4] fail to detect sky in Fig. 4 because they highly depend on per-defined sky color. They can have better performance if pre-defined sky color matches input image

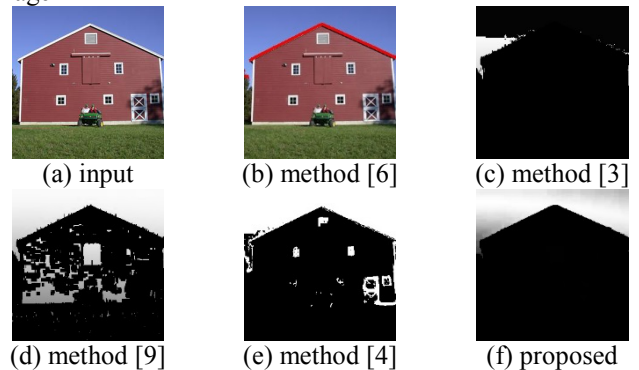


Figure 5 Results of corresponding reference work.

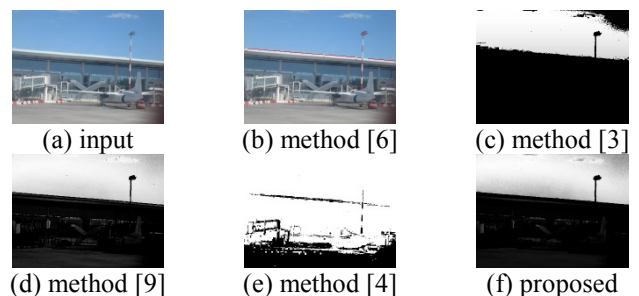


Figure 6 Results of corresponding reference work.

It is obvious that methods highly depending on color are not stable. They may have good performance if pre-defined color matches input image while bad performance if not.

As for [9], we can compare Figure 5(d) and Figure 6(d). If the sky-ground boundary is strong straight line, we have good result as Figure 5(d). However if not, we will have worse result as Figure 6(d).

Method proposed by [6] has relatively better and robust performance. One obvious problem of [6] is that this boundary cannot include those sky-pixels underneath some strong gradient values, as Fig. 1. Our approach avoids such problems by detecting sky in the per-pixel level. It does not matter if there are sky-pixels underneath strong gradient values caused by cloud or roof or something else.

Finally running time are shown here. We implement our algorithm with C++ and OpenCV which is compiled with Intel Threading Building Blocks library support for multithread processing.

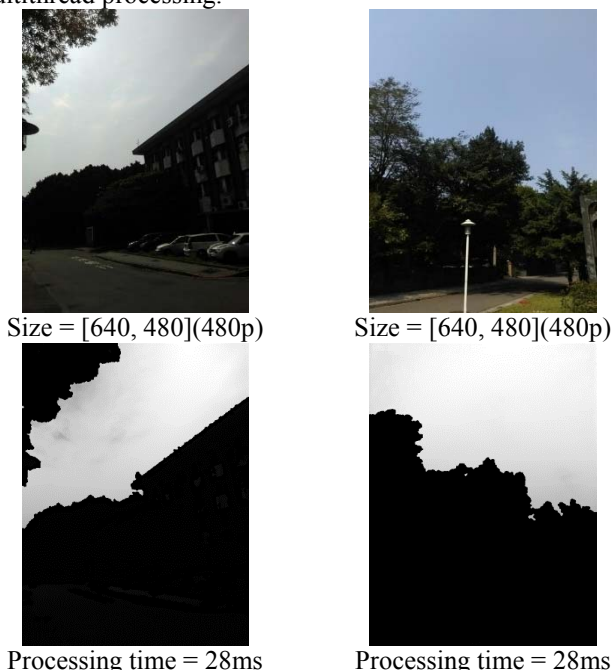


Fig. 7 Processing time of several images.

Table II Testing results on the human-labeled database [10]

Algorithm	Error Avg.	Error Std.
Method [6]	0.07419	0.05625
Method [3]	0.24418	0.08389
Method [9]	0.09504	0.06650
Method [4]	0.32697	0.08232
Proposed Method	0.05066	0.05795

In Table II, we compare the accuracy proposed algorithm with that of other methods using the human-labeled database in [10]. We can see that proposed method has quite good performance compared to other works. Moreover, the proposed algorithm can be performed in real-time in the 480p video while only 28ms is required to process each 640x480 image on a common desktop.

Note that the database in [10] includes the images with and without the sky. The proposed algorithm can well distinguish the two cases.

IV. CONCLUSION

A more robust and efficient algorithm for sky detection is proposed. The algorithm is based on a hybrid probability model which adopts the information of gradients, colors, and locations. The computation time of the proposed algorithm is less than 28ms for each image. Table II shows that our accuracy measurement on a human-labeled database is better than that of other work.

ACKNOWLEDGEMENT

This work was supported by Qualcomm Technologies, Inc.

REFERENCES

- [1] M. Szummer, and R. W. Picard, "Indoor-outdoor image classification," in *IEEE International Workshop on Content-Based Access of Image and Video Database*, pp. 42-51, Jan. 1998.
- [2] J. Luo, and A. Savakis, "Indoor vs outdoor classification of consumer photographs using low-level and semantic features," in *IEEE International Conference on Image Processing*, vol. 2, pp. 745-748, Oct. 2001.
- [3] B. Zafarifar, "Blue sky detection for picture quality enhancement," in *Advanced Concepts for Intelligent Vision Systems* (pp. 522-532). Springer, Berlin Heidelberg, Jan. 2006.
- [4] K. H. Irfanullah, Q. Sattar, and A. A. Sadaqat-ur-Rehman, "An efficient approach for sky detection," *IJCSI International Journal of Computer Science*, pp. 1694-0814, 2013.
- [5] J. Luo and S. P. Etz, "A physical model-based approach to detecting sky in photographic images," *IEEE Trans. Image Processing*, vol. 11, issue 3, pp. 201-212, 2002.
- [6] Y. Shen and Q. Wang, "Sky region detection in a single image for autonomous ground robot navigation," *International Journal of Advanced Robotic Systems*, vol. 10, pp. 362, 2013.
- [7] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA Engineer*, vol. 29, issue 6, pp. 33-41, 1984.
- [8] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *IEEE conference on Computer Vision and Pattern Recognition*, pp. 3485-3492, June 2010.
- [9] B. Zafarifar and H. Weda, "Horizon detection based on sky-color and edge features," in *Electronic Imaging, International Society for Optics and Photonics*, pp. 682220-682220, Jan. 2008.
- [10] human labeled database:
https://drive.google.com/folderview?id=0B_kITe-PQbEvmfh4a0xaTUZHVC12S1VEVFBvZkIQdINXV0U3aXoybjBheU5wR043S3Z3ckE&usp=sharing