Mobile Haze Removal Application

Holly Chiang
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
Email: hchiang1@stanford.edu

Yifan Ge
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
Email: gyifan@stanford.edu

***Abstract*— In this paper, we implement a dehazing algorithm using dark pixel detection in MatLab and C++/OpenCV. The algorithm is originally proposed in Yu *et al.*'s paper. The simplicity and effectiveness of the algorithm make it possible for us to run the C++/OpenCV implementation on Android phones through android NDK with reasonable speed.**

Keywords—dehaze; image processing; mobile application; computer vision

I. Introduction

In the last couple decades, China has developed its economy by largely expanding heavily polluting industries. Public concern over the environmental consequences of this growth has exploded in recent years. China's hyperactive microblogs logged 2.5 million posts on "smog" in a single month in 2013 [1]. Smog filled images of Chinese cities went viral online and have come to represent the severe pollution problem in China. The generation born after 2000 has probably never seen an unpolluted China. In this project, we plan to build a mobile haze removal application that lets users see a different side of China. By removing the haze and reconstructing a clear sky, the application will present an image of an unpolluted China.

The effect of haze on an image is described by the following equation:

$$I(x) = R(x)\, t(x) + A \times (1 - t(x)) \qquad (1)$$

I is the viewed image intensity, R is the radiance from light reflecting off the viewed object, t is the amount of light transmitted through, and A is the atmospheric light. Our goal is to recover the haze free image $R(x)$. We chose to solve for $R(x)$ by loosely following the methods outlined by Yu *et al.*, which uses segmentation and linear fitting to estimate the thickness of the haze [2]. Yu *et al.* method claimed to be at least 10 times faster than comparable algorithms, including another algorithm we were considering by He *et al.*, and as we intended to implement the program on a mobile device with limited computing power, speed and efficiency were our main criteria for algorithm selection.

II. Implementation

We based our implementation on Yu *et al.*'s paper. The paper breaks down the dehazing algorithm into six steps. Following those steps, we implemented the algorithm in both MatLab and C++ with OpenCV. The steps we took are described below.

Identify applicable sponsor/s here. If no sponsors, delete this text box (*sponsors*).

A. *Original Image*

To be able to compare our results with Yu *et al.*'s, we selected an image they ran their algorithm on. This image is shown in Fig. 1(a). The result of each of the algorithm steps listed below is shown in Fig.1 under the corresponding letter.

B. *Computing the Dark Map*

The first step is computing the dark map of the image. Based on equation (1), we can obtain:

$$t(x) = (A - I(x)) / (A - R(x)) \qquad (2)$$

If we compute the transmission function on the dark pixels, those pixels would appear to reflect little light, and we get

$$t(x) = 1 - I(x) / A \qquad (3)$$

by setting the object radiance to 0. $R(x)$ is closest to 0 in the color channel with the smallest value. We can therefore estimate $t(x)$ by $1 - I^c(x) / A^c$, where c indicates the minimum color channel at x. Since $-1 / A^c$ is a constant after white balancing the image with respect to A, $- I^c(x)$ is the only variable and we can use $- I^c(x)$ as a guide for generating the transmission map. The image constructed by Ic at each pixel is called the dark map. To find the dark map, we simply loop through all 3 channels for every pixel and use the minimum value to construct a single channel gray value image.

C. *Size-controlled Segmentation Based on the Dark Map*

The second step is segmenting the image into local blocks. An assumption made in Yu *et al.*'s paper states that transmission *t* would not change sharply in a local area with continuous depth. Based on this assumption, Yu *et al* performed linear fitting on each block of the dark pixels to model the transmission map, which will be discussed in Section II part E. As the dark map is an estimation of transmission, we performed segmentation on the dark map using the variable size segmentation algorithm proposed by Felzenszwalb and Huttenlocher [3]. This step is only implemented in C++. To achieve proper block size for accurate linear fitting, we used σ = 0.5, k = 150 and *min_size* = 400. After some experiments, we found that a smaller block size produces better results. Thus, our implementation tends to have more blocks than Yu *et al.*'s implementation.

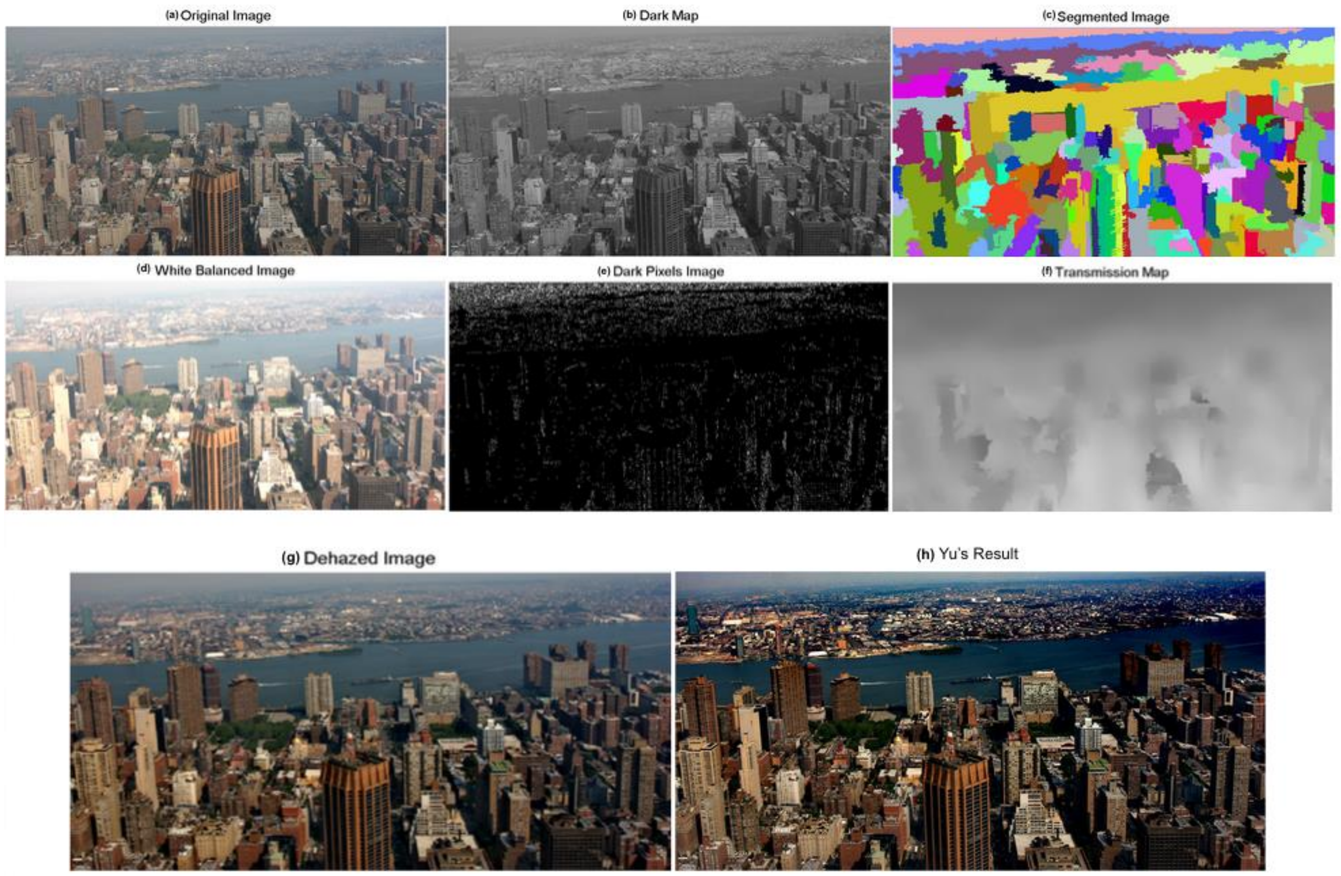

Fig. 1: Implementation of Yu *et al.*'s algorithm. (a-g) correspond to steps outlined in the Image Processing section. (h) is the result shown in Yu *et al.*'s paper.

D. *White Balancing*

The third step is estimating the constant airlight value A. As Yu *et al*. argue in their paper, the brightest color may not be the best estimation of airlight as it may be attributed to noise. Instead, Yu *et al*. finds the block with the brightest average color value and calls it the haze-opaque region. The average *RGB* value of the haze-opaque region is then used as the airlight value to white balance the image.

E. *Detecting Dark Pixels*

The fourth step is detecting the dark pixels, which will be used to achieve linear fitting of the transmission function within each block. Before dark pixel detection, we need to employ a median filter to denoise the image. Based on Yu *et al.*'s observations, there are two substeps for selecting dark pixels.

1) Channel Transformation: According to Yu *et al*., if we transform I from the *RGB* space into the YC_rC_b space, the C_r and C_b of dark pixels will be close to 128, with C_r and C_b ranging from 16 to 240. In the implementation, we threshold the L_2 distance of C_r and C_b value to 128 to get the dark pixels.

$$(C_r - 128)^2 + (C_b - 128)^2 < \theta \qquad (4)$$

2) Neighboring Domain: To avoid false positive detection, Yu proposes that a dark pixel has a smaller luminance than pixels in its neighborhood. By ensuring the maximum value in *RGB* channels of the dark pixel is less than the minimum value in the *RGB* channels of its neighboring pixels, we are able to eliminate a large number of falsely detected dark pixels.

F. *Fitting the Transmission Map*

The fifth step is constructing the transmission map. We can model the transmission function in each block based on the transmission values of its dark pixels. To calculate the initial transmission values, we can derive the following transmission value of dark pixels from equation (1).

$$t(x) = 1 - \min_{r,g,b}\{I(x) - R(x)\, t(x)\} / A^c \tag{5}$$

$$t(x) \sim 1 - \min_{r,g,b} I(x) / A^c \tag{6}$$

Where $\min_{r,g,b} I(x)$ is the minimal color channel of $I(x)$ and A^c is the corresponding color channel of vector A. For a block containing more than 20 dark pixels, linear fitting is performed on t, row and col number of these dark pixels and their corresponding transmission values in each block individually. We use this $t = a \times row + b \times col + c$ to model the transmission map within each block. If the block has less than 20 pixels, we fit the block with the smallest rectangle and use its perimeter pixels to fit the transmission model. After we computed a transmission map based on these transmission models, we employed a guided image filter [4] to improve the accuracy of the segmented transmission map. We used *fitlm* and *fitLine* functions in MatLab and C++/OpenCV implementations respectively.

G. *Recovering the Scene Radiance (Dehazed Image)*

The final step is recovering the radiance from $I(x)$. From equation (1), we can get the expression of $R(x)$ as a function of airlight, A, and transmission function, $t(x)$, as below.

$$R(x) = A - (A - I(x)) / t(x) \tag{7}$$

In the case when $t(x)$ is close to zero, $R(x)$ will approach infinity. So based on Yu *et al.*'s approach, we set the largest $t(x)$ in the haze-opaque region, $t0$, as the lower bound of $t(x)$. Finally, we have the final radiance function as

$$R(x) = A - (A - I(x)) / \max\{t(x), t0\} \tag{8}$$

III. Android Implementation

After achieving a working MatLab implementation, we rewrote the code in C++ with the OpenCV class, and ran our program on an Android phone. We were not able to get code running on the provided Droid phones, but were able to port it onto an Android phone that could download OpenCV Manager.

IV. Discussion

Our implementation of Yu *et al.*'s algorithm is able to successfully dehaze most images. In Fig. 1, we compare our result (g) to that of Yu et al (h). Due to the particularities with which we implemented the algorithm, our dehazed image suffers from less oversaturation than the result given in the paper. We tested our program on various foggy or smog filled pictures, and found that in most cases our algorithm successfully dehazes the image. However, our implementation does not perform well on images that do not segment into many pieces and ends up oversaturating the image.

The performance of the algorithm is quite fast. In MatLab, dehazed images are generated nearly instantly. On a mobile phone, images are usually dehazed in under 5 seconds. Images that do not contain as many dark pixels take longer to run as the algorithm iteratively searches until it finds enough dark pixels.

V. Conclusion

We were able to successfully implement an algorithm to dehaze images on MatLab and for Android. While following the general steps outlined by Yu *et al.*, our specific implementation achieved a result that in some cases produces a better result, suggesting that the implementation of the algorithm can be even further refined. As our implementation can be run on both computing and mobile platforms, our dehazing program can be easily run and show photographers what their pictures would look like in the absence of factors like smog or fog.

Acknowledgements

We would like to give special thanks to the teaching staff of EE 368 for the hard work in this quarter. We would like to acknowledge Kaiming He for the use of his guided filter implementation, P. Felzenszwalb and D. Huttenlocher for the use of the Efficient Graph-Based Image Segmentation implementation.

References

[1] He, K., Sun, J., & Tang, X. (2011). Single image haze removal using dark channel prior. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 33(12), 2341-2353.

[2] Yu, Q., Ding, Z., Rong, R., Zhang, Z., & Wang, D. (2011). Dark Pixel Detection: A Novel Single Image Dehaze Approach. IVCNZ.

[3] Felzenszwalb, P., Huttenlocher, D., Efficient graph-based image segmentation. International Journal of Computer Vision,59(2), 167-181, 2004.

[4] He, K., Sun, J., Tang, X, Guided image filtering. ECCV, 2010.