Enhancement Techniques for Low-light and Hazy Images

Parth Shah, Suyash Bagad, Saurabh Kolambe
15D070004, 15D070007, 15D070011

***Abstract*— Images acquired under very low light conditions, where the image features are nearly invisible and the noise is significant, need to be filtered. Similarly, images of outdoor scenes degraded by haze, fog, and smoke due to atmospheric absorption and scattering, naturally need enhancement. In this project, we have implemented several fast enhancement techniques based on de-hazing for single low-light images. The first implementation uses the luminance map to estimate the global atmospheric light and the transmittance owing to the observed similarity between luminance map and dark channel prior (DCP). Another simple but effective method uses DCP to recover a high-quality haze-free image. The former method is shown to have two merits over the latter, firstly, the computational complexity is greatly reduced; and the problem of block artifacts is also addressed.**

I. Introduction

Hazy images are generally covered by mist or haze and thus are not very clear or well defined. Similarly Low light images have lesser dynamic range. Which degrades the image quality and makes it difficult to distinguish or to process such images for applications in computer vision such as object detection, object recognition. Removing haze and enhancing the Low light images can significantly increase the scene visibility and color contrast thus can be helpful in such applications

A. Low light Enhancement

Traditional algorithms for low light enhancement used alpha correction or histogram equalization techniques. Though This techniques is simple to implement but can cause saturation of color and thus sometimes loss of information. Other proposed solutions for this problem used CEM (color estimation model) or enhancement algorithms based on sparse representation of images.

B. Haze removal algorithms

Haze removal problem is ill posed problem if only a single hazy image is input. Many algorithms used multiple images based haze removal. Polarization based method uses multiple hazy input images with different degree of polarization. A Depth based methods used a predefined 3D depth model to filter out the haze. Thus given some prior with the input, Haze removal is comparatively easy. It is been observed that the hazy images have lesser contrast than well defined images. And thus a local contrast maximizing method can also be used.

To enhance single hazy image, a prior has to be derived from the same input image. Dark channel prior can be used for this case. Haze free images tends to have very low pixel value at least in one channel of the (RGB) image. These dark pixels can be used to directly to estimate the haze transmission in the image. inverted Low light images can show resemblance to a hazy image, thus above dehazing algorithm can be used to enhance the low light images as well. Luminance map is one of the alternative to the dark channel prior which is much faster in computation and gives lesser block artifacts caused by the local patch based algorithm used in Dark channel prior.

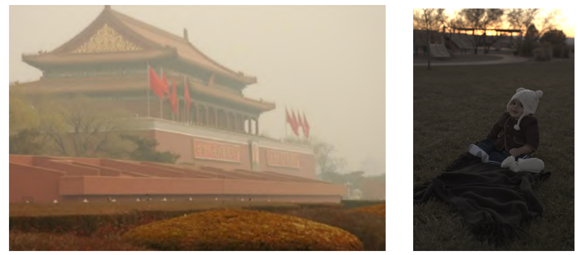


Fig. 1: Hazy and Low-light images

II. Background

The Goal of enhancing hazy and low light images is:

- Scene restoration
- Depth estimation

A hazy image can be modelled as,

$$I(m,n) = J(m,n).t(m,n) + A[1 - t(m,n)] \tag{1}$$

- I is the hazy image
- J is the underlying scene radiance
- t is the transmittance
- A is the global atmospheric light

Thus given a hazy input I, we can compute the enhanced image J if t, A are known. The first term $J(x)t(x)$ on the right-hand side is called direct attenuation, and the second term $A(1 - t(x))$ is called airlight. The direct attenuation describes the scene radiance and its decay in the medium, and the airlight results from previously scattered light and leads to the shift of the scene colors. While the direct attenuation is a multiplicative distortion of the scene radiance, the airlight is an additive one. When the atmosphere is homogenous, the transmission t can be expressed as

$$t(x) = 0.25 \times e^{\beta d(x)} \tag{2}$$

where b is the scattering coefficient of the atmosphere and d is the scene depth. This equation indicates that the scene radiance is attenuated exponentially with the depth. If we

(a) Hazefree image (b) DCP of Hazefree image

(c) Slightly hazy image (d) DCP of hazy image

Fig. 2: Dark Channel prior examples

can recover the transmission, we can also recover the depth up to an unknown scale.

The inverted low light images have high similarity with the hazy image and thus can be used modelled using a similar expression,

$$I_{inv}(m,n) = J_{inv}(m,n).t(m,n) + A[1 - t(m,n)] \quad (3)$$

Where the inverted image is computed as

$$I_{inv} = 255 - I \quad (4)$$

III. Methods

A. Haze removal Using DCP

Firstly, we will demonstrate the haze removal technique using dark channel prior method.

1) Computing DCP: The dark channel prior is based on the observation that the haze-free images has patches with at least one color channel with very low pixel value tending to zero. For a random image, the dark channel prior of the image is given by

$$D^{dark}(x) = min_{c\in\{R,G,B\}}(min_{y\in\Omega(x)}(I^c(y))) \quad (5)$$

Intuitively, for computing DCP, we take the minimum of the pixel intensities across all three regions, in a patch Ω. The example of DCP is shown in 2. Due to the additive airlight, a hazy image is brighter than its haze-free version where the transmission t is low. So, the dark channel of a hazy image will have higher intensity in regions with denser haze (as also demonstrated in fig. 2. Visually, the intensity of the dark channel is a rough approximation of the thickness of the haze.

2) Estimating the Atmospheric light: The dark channel of a hazy image approximates the haze denseness (see Fig. 2). So we can use the dark channel to detect the most haze-opaque region and estimate the atmospheric light. We first pick the top 0.1 percent brightest pixels in the dark channel. These pixels are usually most haze-opaque. We first pick the top 0.1 percent brightest pixels in the dark channel. These pixels are usually most haze-opaque and note that these pixels may not be brightest ones in the whole input image. This simple method based on the dark channel prior is more robust than the ”brightest pixel” method. We use it to automatically estimate the atmospheric lights for all images shown in this work.

3) Estimating the Transmission : Now, given that we have the atmospheric light A, we first normalize the haze imaging equation 1 by A:

$$\frac{I^c(x)}{A^c} = t(x)\frac{J^c(x)}{A^c} + 1 - t(x) \quad (6)$$

where each color channel is normalized independently. We further assume that the transmission in a local patch x is constant. We denote this transmission as $\hat{t}(x)$. Now, calculating the dark channel prior on both sides, we have

$$\begin{aligned} & min_{c\in\{R,G,B\}}(min_{y\in\Omega(x)}(\frac{I^c(y)}{A^c})) \\ & = 1 - \hat{t}(x) + min_{c\in\{R,G,B\}}(min_{y\in\Omega(x)}(\frac{J^c(y)}{A^c}))\hat{t}(x) \end{aligned} \quad (7)$$

Since $\hat{t}(x)$ is a constant in the patch, it can be put on the outside of the min operators. As the scene radiance J is a haze-free image, the dark channel of J is close to zero due to the dark channel prior:

$$J^{dark}(x) = min_{c\in\{R,G,B\}}(min_{y\in\Omega(x)}(J^c(y))) = 0 \quad (8)$$

This leads to ($A^c > 0$)

$$min_{c\in\{R,G,B\}}(min_{y\in\Omega(x)}(\frac{J^c(y)}{A^c})) = 0 \quad (9)$$

Putting 9 in 7, we get the transmission map as

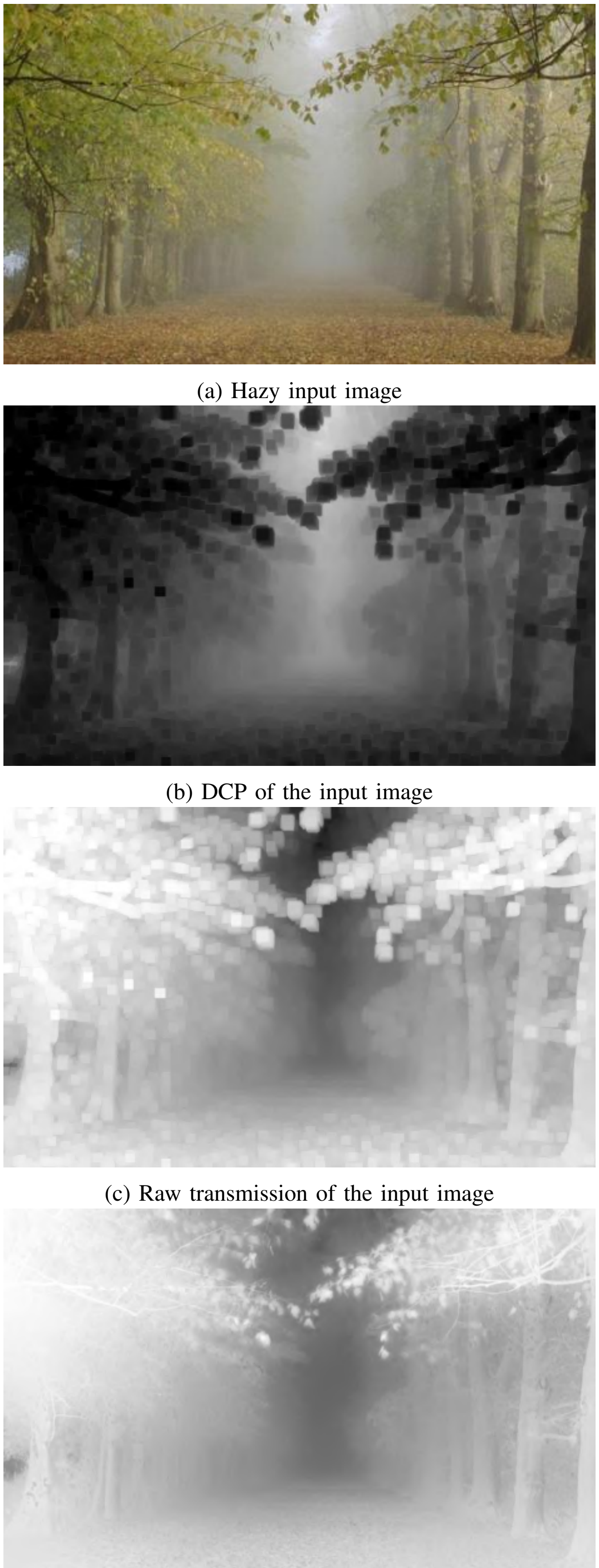
$$\hat{t}(x) = 1 - min_{c\in\{R,G,B\}}(min_{y\in\Omega(x)}(\frac{I^c(y)}{A^c})) \quad (10)$$

This can be written as

$$\hat{t}(x) = 1 - DCP(\frac{I}{A}) \quad (11)$$

In practice, even on clear days the atmosphere is not absolutely free of any particle. So the haze still exists when we look at distant objects. So, we can optionally keep a very small amount of haze for the distant objects by introducing a constant parameter ω $(0 < \omega \leq 1)$ into 10 to finally get

$$\boxed{\hat{t}(x) = 1 - \omega\ min_{c\in\{R,G,B\}}(min_{y\in\Omega(x)}(\frac{I^c(y)}{A^c}))} \quad (12)$$



(a) Hazy input image

(b) DCP of the input image

(c) Raw transmission of the input image

(d) Refined transmission of the input image

Fig. 3: Transmission map estimation.

4) Tackling issues of Block artifacts and Patch size: As can be seen from fig. 3a, we could easily see some block artifacts in the transmission estimated directly from the equation 12. These block artifacts depend on the patch size while computing DCP. The dark channel prior becomes better for a larger patch size because the probability that a patch contains a dark pixel is increased, but the assumption that the transmission is constant in a patch becomes less appropriate. If the patch size is too large, halos near depth edges may become stronger. In order to solve this issue, we used used a *filter* to refine the raw transmission map. In our implementation, we give the user the choice to select which filter to use among *Guided, Bilateral, Mean, Gaussian*. However, the results for guided filter were by far the best. The reason is that, here we need to filter block artifacts without losing edge information. Thus we need a filter which filters out medeivally high frequency components but not the frequency components corresponding to *edges*. The raw and refined transmission map for the image in fig. 3 are shown in 3c, 3d.

5) Recovering Scene Radiance: With the atmospheric light and the transmission map, we can recover the scene radiance easily. But the direct attenuation term $J(x)t(x)$ can be very close to zero when the transmission $t(x)$ is close to zero. The directly recovered scene radiance J is prone to noise. Therefore, we restrict the transmission $t(x)$ by a lower bound t_0 , i.e., we preserve a small amount of haze in very dense haze regions. The final scene radiance $J(x)$ is recovered by

$$\boxed{J(x) = \frac{I(x) - A}{max\{t(x), t_0\}} + A} \tag{13}$$

A typical value of t_0 is 0.1. However, we have customized it to be given as an input by the user. Since the scene radiance is usually not as bright as the atmospheric light, the image after haze removal looks dim. So we increase the exposure of $J(x)$ for display after recovery.

6) A note on the patch size: A key parameter in this algorithm is the patch size in 12. As discussed already, 9: the larger the patch size, the darker the dark channel. Consequently, 9 is less accurate for a small patch, and the recovered scene radiance is oversaturated. On the other hand, the assumption that the transmission is constant in a patch becomes less appropriate. However, in the results in 4, we have used patch size $9 \times 9,\ 15 \times 15$, which shows that our implementation works for sufficiently large patch sizes. This is because the guided filtering technique is able to reduce the artifacts introduced by large patches.

7) Results of Haze removal: We have implemented this very simple but powerful haze removal technique with customization of parameters designed for the user. The results are shown in fig. 4.

Fig. 4: Results of Haze removal implemented on a variety of images (in order, *input image* followed by *recontructed radiance*.)

Fig. 5: Results of Haze removal (continued)

Fig. 6: Results of Haze removal (continued)

Fig. 7: Results of Haze removal (continued)

B. Low Light Image enhancement using Luminance map

In this work, a fast low-light enhancement algorithm is proposed based on de-hazing technique. In our proposed algorithm, luminance map of the inverted low-light image is used to estimate global atmospheric light and transmittance instead of DCP used in previously, based on the analysis of the similarity of the luminance map and the DCP. By doing so, not only the computation complexity can be reduced, but also the block artifacts could be avoided.

This method is inspired by the observation that luminance maps of the inverted low-light images have high similarity with the DCP as shown in Fig. 8. Therefore we will use the luminance map to estimate the global atmospheric light and transmittance instead of the DCP.

Using luminance over DCP has two merits in this proposition. On one hand, the minimum filtering is needed to compute the DCP, which is quite time-consuming; on the other hand, the transmittance in a local patch is assumed to be constant and it is not refined by soft matting because of compu- tation complexity. Thus severe block artifacts would be introduced in the places where transmittance is not continuous. Both of these drawbacks are overcome by this method.

1) Computing Luminance: Instead of DCP, for increasing the computational complexity, luminance map of the inverted low-light image is used to estimate global atmospheric light and transmittance, based on the analysis of the similarity of the luminance map and the DCP. Three color channels of the inverted-low light are weighted summed to computed the luminance map $L(x)$

$$L(x) = 0.299 \times I^R(x) + 0.587 \times I^G(x) + 0.114 \times I^B(x) \quad (14)$$

2) Estimating Atmospheric Light: Considering the similarity between the luminance map and the DCP, we substitute the luminance map for the DCP to estimate the global atmospheric light. The pixel with the highest intensity in the inverted low-light image is selected as the global atmospheric light from the 0.1 % pixels with the highest intensity in the luminance map.

3) Estimating Transmission: Though the inverted low-light image is not a real haze one, its transmittance is closely connected to luminance. The darker the scene is, the denser the corresponding haze of the inverted low-light image. Therefore it is more reasonable to estimate the transmittance using the luminance map. The initial transmittance map $\hat{t}(x))$ is estimated using the luminance map as:

$$\boxed{\hat{t}(x) = 1 - \omega \ L(x)} \quad (15)$$

where ω is a parameter. The smoother transmittance may allow the underlying scene radiance map to contain more details, so a mean filter is used to obtain the final transmittance $t(x)$:

$$t(x) = meanfilter(\hat{t}(x)) \quad (16)$$

(a) Low-light image

(b) DCP of inverted image

(c) Luminance map of original image

(d) Luminance map of inverted image

Fig. 8: Example of DCP and Luminance for a low-light image

The transmittance $t(x)$ tends to be very low, thus the lower bound of the transmittance is limited to $t_0 = 0.01$ in order to avoid being zero as the denominator in the step of recovering the scene radiance. (This too has been implemented in a way that it is user customizable).

4) Recovering Scene Radiance: After obtaining the estimation values of the global atmospheric light and transmittance, the inverted underlying scene radiance $J_{inv}(x)$ can be recovered as

$$\boxed{J_{inv}(x) = \frac{I_{inv}(x) - A}{max\{t(x), t_0\}} + A} \quad (17)$$

5) Results of Low-light image enhancement: We have implemented this very effective low-light enhancement technique with customization of parameters designed for the user. The results are shown in fig. 9.

Fig. 9: Results of low-light image enhacement implemented on a variety of images (in order, *input image* followed by *recontructed radiance*.)

Fig. 10: Results of low-light enhacement (continued).

Fig. 11: Results of low-light enhacement (continued).

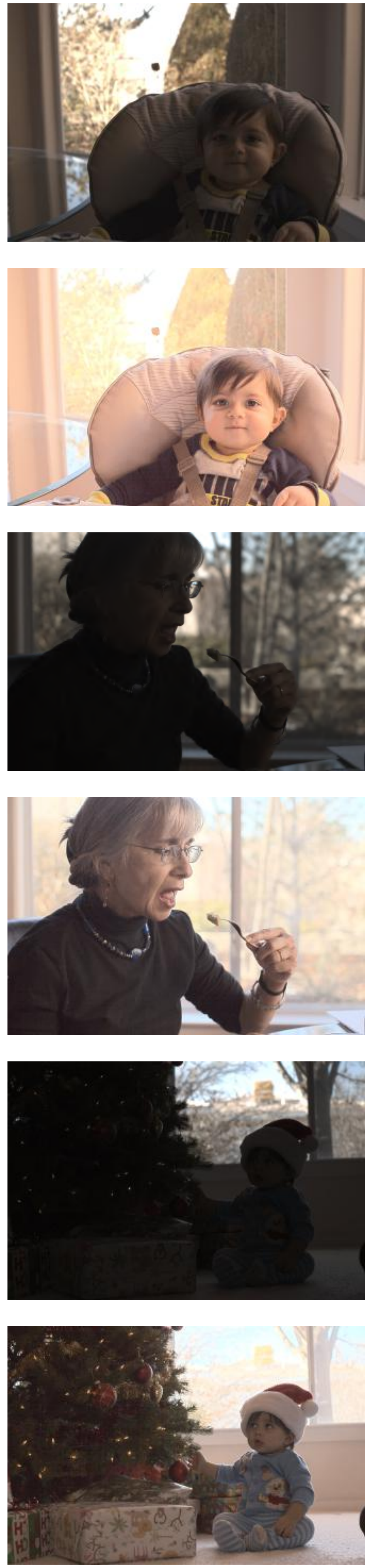


Fig. 12: Results of low-light enhacement (continued).

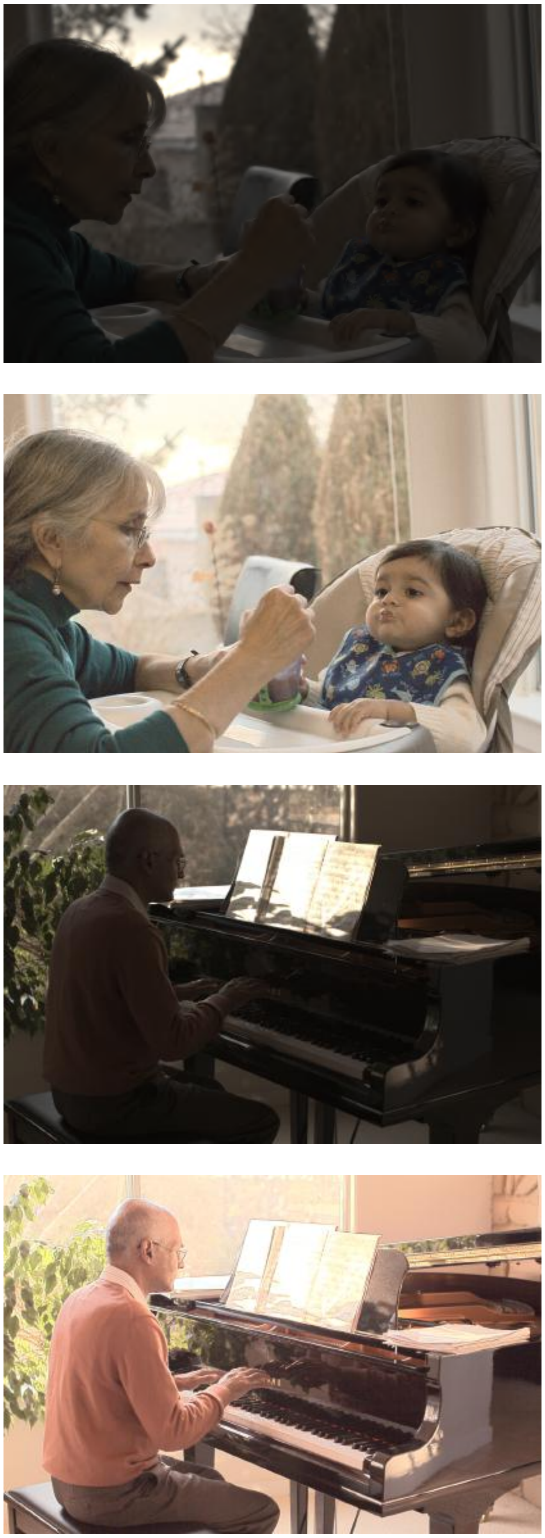


Fig. 13: Results of low-light enhacement (continued).

Fig. 14: Results of low-light enhacement, here the transmission map is computed using the luminance of inverted image, but A computed using luminance of original image, leading to superior results.

IV. Results and Conclusion

This work successfully solves the problem of low-light image enhancement and haze removal using very simple but effective techniques.

In the first method, a very simple but powerful prior, called the dark channel prior, is used for single image haze removal. The dark channel prior is based on the statistics of outdoor haze-free images. Combining the prior with the haze imaging model, single image haze removal becomes simpler and more effective.

In the second method, the luminance map of the inverted low-light image is used to estimate the global atmospheric light and transmittance instead of the DCP according to the high similarity between the luminance map and the DCP. Experimental results demonstrate that our proposed algorithm achieves excellent enhancement results in terms of subjective and objective quality. In addition, our proposed algorithm can meet the real-time requirements in practical applications since the computation complexity is greatly reduced without computing DCP using minimum filtering.

The low-light image enhancement method can be used in real-time surveillance applications as the computational speed is much high and could match real-time specifications.

Acknowledgement

We would personally like to thank Prof. Amit Sethi for designing this project since it really contributed a lot in my learning of this sub-section of the subject. Although it took quite a bit of time to code, but all of it was worth it!

References

[1] Pizer, S.M., Amburn, E.P., Austin, J.D., Cromartie, R., Geselowitz, A., Greer, T., Romeny, B.H., Zimmerman, J.B., Zuiderveld, K.: Adaptive histogram equalization and its variations. Comput. Vision Graph. Image Process. 39(3), 355368 (1987)

[2] Malm, H., Oskarsson, M., Warrant, E., Clarberg, P., Hasselgren, J., Lejdfors, C.: Adaptive enhancement and noise reduction in very low light-level video. In: Proceedings of the 11st IEEE International Conference on Computer Vision, Rio de Janeiro, Brazil, pp. 18, 1421 October 2007

[3] R. Fattal, Single Image Dehazing, Proc. ACM SIGGRAPH 08, 2008.

[4] R. Tan, Visibility in Bad Weather from a Single Image, Proc. IEEE Conf. Computer Vision and Pattern Recognition, June 2008.

[5] M. van Herk, A Fast Algorithm for Local Minimum and Maximum Filters on Rectangular and Octagonal Kernels, Pattern Recognition Letters, vol. 13, pp. 517-521, 1992.

[6] He, K., Sun, J., Tang, X.: Single image haze removal using dark channel prior. IEEE Trans. Pattern Anal. Mach. Intell. 33(12), 23412353 (2011)

[7] Fu, H., Ma, H., Wu, S.: Night removal by color estimation and sparse representa- tion. In: Proceedings of the 21st IEEE International Conference on Pattern Recog- nition, Tsukuba, Japan, pp. 36563659, 1115 November 2012

[8] Sen, P., Kalantari, N.K., Yaesoubi, M.: Robust patch-based HDR reconstruction of dynamic scenes. ACM Trans. Graph. 31(6), 439445 (2012)

[9] Goldstein, Amit and Fattal, Raanan, Blur-Kernel Estimation from Spectral Irregularities, European Conference on Computer Vision (ECCV), 2012 622–635.

[10] A. Hore and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," 2010 20th International Conference on Pattern Recognition, Istanbul, 2010, pp. 2366-2369.

[11] H. Zhang, W. He, L. Zhang, H. Shen and Q. Yuan, "Hyperspectral Image Restoration Using Low-Rank Matrix Recovery," in IEEE Transactions on Geoscience and Remote Sensing, vol. 52, no. 8, pp. 4729-4743, Aug. 2014.

[12] Juan Song, Liang Zhang, Peiyi Shen, Xilu Peng, and Guangming Zhu, Single Low-Light Image Enhancement Using Luminance Map, CCPR 2016, Part II, CCIS 663, pp. 101110, 2016.

[13] Guo, F., Cai, Z., Xie, B., Tang, J.: Automatic image haze removal based on lumi- nance component. In: Proceedings of the Sixth IEEE International Conference on Wireless Communications Networking and Mobile Computing, Chengdu, China, pp. 14, 2325 September 2010

Appendix

The folder of the code and results can be found here. The code can be run on any system with `Python 3.3` or newer, along with `openCV`, `numpy`, `tkinter`, `PIL`, `matplotlib` packages installed. The implementation is done using a terminal-based GUI, so the steps after running the code `main.py` are self-explanatory. The code of the implementation is:

```python
# ##################################################
# Enhancement Techniques for Low-Light and Hazy Images
#
#     A fast enhancement method based on de-hazing
#     is implemented for single low-light images.
#
#     The dark channel prior (DCP: a statistic for
#     haze-free atmospheric images) is
#     used in this implementation to estimate the global
#     atmospheric light and the transmittance.
#
#     Instead of dark channel prior (DCP) used n the
#     de-hazing related literature, the luminance map is
#     used in this implementation to estimate the global
#     atmospheric light and the transmittance.
#
# Suyash Bagad | Saurabh Kolambe | Parth Shah
# 15D070007    | 15D070011       | 15D070004
#
#
# Project: Enhancement Techniques for Low-Light and Hazy Images
# EE 610: Image Processing
# Autumn Semester, 2018
#
# Department of Electrical Engineering,
# IIT Bombay
#
# Copyrights reserved @ Suyash Bagad
# ##################################################

# Import the required modules in Python
import numpy as np
from PIL import Image
from tkinter import *
from tkinter.filedialog import askopenfilename
import sys
from PIL import Image
import cv2
import matplotlib.pyplot as plt
from PIL import Image
from guided_filter import guided_filter
import shutil, os

def luminance(I):
    """
    Computes luminance of an Image I (RGB).

    Parameters
    -----------
    I:  an M * N * 3 numpy array containing data ([0, L-1]) in the image where
        M is the height, N is the width, 3 represents R/G/B channels.

    Return
    -----------
    A scalar 'luminance' value of input image.
    """

    I_b, I_g, I_r = cv2.split(I)
    return (np.array( 0.299 * I_r  +  0.587 * I_g  +  0.114 * I_b )).astype('int')

def get_dcp(I, h, w):
    """
    Get the dark channel prior in the (RGB) image data.

    Parameters
    -----------
    I:  an M * N * 3 numpy array containing data ([0, L-1]) in the image where
        M is the height, N is the width, 3 represents R/G/B channels.
    h:  window height
    w:  window width

    Return
    -----------
    An M * N array for the dark channel prior ([0, L-1]).
    """

    M, N, _ = I.shape
    padded = np.pad(I, ((w // 2, w // 2), (h // 2, h // 2 + (h and 1)), (0, 0)), 'edge')
    dark_channel = np.zeros((M, N))
    for i, j in np.ndindex(dark_channel.shape):
        dark_channel[i, j] = np.min(padded[i:i + h, j:j + w, :])
    return dark_channel

def atmospheric_light(I, prior, f):
    """
    Computes the Global atmospheric light (A) for all three channels of image I.

    Parameters
    -----------
    I:      the M * N * 3 RGB image data ([0, L-1]) as numpy array
    prior:  the prior of the image as an M * N numpy array
    f:      fraction of pixels for estimating the atmosphere light

    Return
    -----------
    A 3-element array containing atmosphere light ([0, L-1]) for each channel
    """

    M, N = prior.shape
    flatI = I.reshape(M * N , 3)
    flatdark = prior.ravel()

    # Find top M * N * f indices
    searchidx = (-flatdark).argsort()[:int(M * N * f)]
    # print('Atmosphere light region:', [(i // N, i % N) for i in searchidx])

    # Return maximum intesities in each channel
    return np.max(flatI.take(searchidx, axis=0), axis=0)

def transmittance(I, A, h, w, omega=0.95, prior='dcp', filter_t='guided', tmin=0.15, r=40, eps=1e-3):
```

```python
    """
    Get the transmission esitmate in the (RGB) image data.

    Parameters
    ----------
    I:        the M * N * 3 RGB image data ([0, L-1]) as numpy array
    A:        a 3-element array containing atmosphere light
              ([0, L-1]) for each channel
    h:        window height for the estimate
    w:        window width for the estimate
    omega:    bias for the estimate
    prior:    the statistical prior of the image as an M * N numpy array
    filter:   the type of filter to be used for refining raw transmittance
    tmin:     threshold of transmittance

    eps:      epsilon for the guided filter

    Return
    ----------
    An M * N array containing the transmission rate (transmittance) ([0.0, 1.0])
    """
    # Transmittance for DCP based approach
    if prior == 'dcp':
        raw_tx = 1.0 - omega * get_dcp(I / A, h, w)

    # Transmittance for luminance based approach
    elif prior == 'luminance':
        raw_tx = 1.0 - omega * luminance(I) / 255.0

    # Throw error in any other case
    else:
        raise ValueError("The 'prior' argument must either be 'luminance' or 'dcp'.")

    # Refined transmittance based on guided filter
    if filter_t == 'guided':
        refined_tx = np.maximum(raw_tx, tmin)
        normI = (I - I.min()) / (I.max() - I.min())
        refined_tx = guided_filter(normI, refined_tx, r, eps)

    # Refined transmittance based on bilateral filter
    elif filter_t == 'bilateral':
        refined_tx = cv2.bilateralFilter(raw_tx, 9, 75, 75)
        refined_tx = np.maximum(refined_tx, tmin)

    # Refined transmittance based on gaussian filter
    elif filter_t == 'gaussian':
        refined_tx = cv2.GaussianBlur(raw_tx, (5, 5), 0)
        refined_tx = np.maximum(refined_tx, tmin)

    # Refined transmittance based on mean filter
    elif filter_t == 'mean':
        refined_tx = cv2.blur(raw_tx, (5, 5))
        refined_tx = np.maximum(refined_tx, tmin)

    else:
        refined_tx = raw_tx

    return raw_tx, refined_tx

def get_radiance(I, A, t):
    """
    Recover the radiance from raw image data with atmosphere light
    and transmission rate estimate.

    Parameters
    ----------
    I:      M * N * 3 data as numpy array for the hazy image
    A:      a 3-element array containing atmosphere light
            ([0, L-1]) for each channel
    t:      estimate of the transmission rate

    Return
    ----------
    M * N * 3 numpy array for the recovered radiance
    """
    tile_t = np.zeros_like(I)
    tile_t[:, :, 0] = tile_t[:, :, 1] = tile_t[:, :, 2] = t
    return ((I - A) / tile_t + A)

def to_img(raw):
    """
    Convert M * N * 3 matrix to 256-bit image data.
    """
    # Threshold to [0, 255]
    cut = np.maximum(np.minimum(raw, 255), 0).astype(np.uint8)

    if len(raw.shape) == 3:
        b, g, r = cut[:,:,0], cut[:,:,1], cut[:,:,2]
        R = Image.fromarray(r)
        G = Image.fromarray(g)
        B = Image.fromarray(b)
        cut = Image.merge("RGB", (R, G, B))
        return cut
    else:
        return Image.fromarray(cut)

def dehaze(image, tmin=0.2, Amax=220, h=15, w=15, f=0.0001, omega=0.95, prior='dcp', filter1='guided', r=40, eps=1e-3):
    """
    Dehaze the given RGB image.

    Parameters
    ----------
    image:      the Image object of the RGB image
    Amax:       upper bound of atmospheric light
    Other parameters same as that of 'transmittance' function.

    Return
    ----------
    (dark, rawt, refinedt, rawrad, rerad)
    Images for dark channel prior, raw transmission estimate,
    refiend transmission estimate, recovered radiance with raw t,
    recovered radiance with refined t.
    """
    I = np.asarray(image, dtype=np.float64)
    if prior == 'dcp':
        Idark = get_dcp(I, h, w)
    elif prior == 'luminance':
        Idark = luminance(I)
    else:
        raise ValueError("The 'prior' argument must either be 'luminance' or 'dcp'.")

    A = atmospheric_light(I, Idark, f)
    print(A)
```

```python
    raw_t, refined_t = transmittance(I, A, h, w, omega=omega, prior=prior, filter_t = filter1, tmin=tmin, eps=eps)
    white = np.full_like(Idark, 255)

    return [to_img(raw) for raw in (Idark, white * raw_t, white * refined_t, get_radiance(I, A, raw_t), get_radiance(I, A, refined_t))]

def bgr_rgb(Ibgr):
    """
    Converts BGR image to RGB using Pillow.
    """
    b, g, r = Ibgr.split()
    return Image.merge("RGB", (r, g, b))


# Main function
if __name__ == "__main__":
    root = Tk()
    root.withdraw()
    initialdir="/home/saurabh/Desktop/SEM_7/IP/Project"
    filename = askopenfilename(initialdir = "/home/saurabh/Desktop/SEM_7/IP/Project", title = "Choose a degraded Image")
    image = cv2.imread(filename)


#    shutil.rmtree(initialdir+'/output')
    out_dir = input("Name of the Output Folder :  ")
    os.mkdir(initialdir+"/Output/" + out_dir)

    dest = initialdir+"/Output/"+ out_dir + "/"

    image1 = Image.fromarray(image)
    image1 = bgr_rgb(image1)
    image1.save(dest + 'input.png')

    print("Input image read from " + filename)
    dict_algo = {
    "d" : "DCP-based",
    "l" : "Luminance-based",
  }

    while(True):
        algo = input("Image restoration method: Press d for DCP-based and l for luminance-based.\n")
        print("Using {0} algorithm".format(dict_algo[algo]))
        if algo == 'd':
            # Dehazing using Dark channel prior
            dark, raw_t, refined_t, raw_rad, refined_rad = dehaze(image, 0.4, 220, 15, 15, 0.0001, 0.98, prior='dcp', filter1='guided')

            # Save results in the output directory
            dark.save(dest + 'dark_dcp.png')
            raw_t.save(dest + 'rawt_dcp.png')
            refined_t.save(dest + 'refinedt_dcp.png')
            raw_rad.save(dest + 'radiance_rawt_dcp.png')
            refined_rad.save(dest + 'output.png')

            break
        elif algo == 'l':
            # Dehazing using luminance map
            image_inv = 255 - image
            Idark, raw_t, refined_t, raw_rad_inv, refined_rad_inv = dehaze(image_inv, 0.2, 220, 15, 15, 0.001, 0.95, 'luminance', 'mean')

            # Save results in the output directory
            white = np.full_like(raw_rad_inv, 255)
            raw_rad = to_img(white - raw_rad_inv)
            refined_rad = to_img(white - refined_rad_inv)

            b, g, r = raw_rad.split()
            raw_rad = Image.merge("RGB", (r, g, b))

            b, g, r = refined_rad.split()
            refined_rad = Image.merge("RGB", (r, g, b))

            Idark.save(dest + 'dark_lum.png')
            raw_t.save(dest + 'rawt_lum.png')
            refined_t.save(dest + 'refinedt_lum.png')
            raw_rad.save(dest + 'radiance_rawt_lum.png')
            refined_rad.save(dest + 'output.png')

            break
        else:
            raise ValueError("Invalid input! Try again.\n")
```

The code for guided filter is:

```python
# ##########################################################################
# Enhancement Techniques for Low-Light and Hazy Images
#
#    Guided filter implementation as a supplement to
#    original problem of dehazing using DCP and luminance.
#
# Suyash Bagad | Saurabh Kolambe | Parth Shah
# 15D070007    | 15D070011       | 15D070004
#
#
# Project: Enhancement Techniques for Low-Light and Hazy Images
# EE 610: Image Processing
# Autumn Semester, 2018
#
# Department of Electrical Engineering,
# IIT Bombay
#
# Copyrights reserved @ Suyash Bagad
# ##########################################################################


from itertools import combinations_with_replacement
from collections import defaultdict
import numpy as np
from numpy.linalg import inv

R, G, B = 0, 1, 2


def boxfilter(I, r):
    """
    Fast box filter implementation.

    Parameters
    ----------
    I:  a single channel/gray image data normalized to [0.0, 1.0]
    r:  window radius
```

```python

    Return
    ----------
    The filtered image data.
    """
    M, N = I.shape
    dest = np.zeros((M, N))

    # cumulative sum over Y axis
    sumY = np.cumsum(I, axis=0)
    # difference over Y axis
    dest[:r + 1] = sumY[r: 2 * r + 1]
    dest[r + 1:M - r] = sumY[2 * r + 1:] - sumY[:M - 2 * r - 1]
    dest[-r:] = np.tile(sumY[-1], (r, 1)) - sumY[M - 2 * r - 1:M - r - 1]

    # cumulative sum over X axis
    sumX = np.cumsum(dest, axis=1)
    # difference over Y axis
    dest[:, :r + 1] = sumX[:, r:2 * r + 1]
    dest[:, r + 1:N - r] = sumX[:, 2 * r + 1:] - sumX[:, :N - 2 * r - 1]
    dest[:, -r:] = np.tile(sumX[:, -1][:, None], (1, r)) - \
        sumX[:, N - 2 * r - 1:N - r - 1]

    return dest


def guided_filter(I, p, r=40, eps=1e-3):
    """
    Refine a filter under the guidance of another (RGB) image.

    Parameters
    ----------
    I:   an M * N * 3 RGB image for guidance.
    p:   the M * N filter to be guided
    r:   the radius of the guidance
    eps: epsilon for the guided filter

    Return
    ----------
    The guided filter.
    """
    M, N = p.shape
    base = boxfilter(np.ones((M, N)), r)

    # each channel of I filtered with the mean filter
    means = [boxfilter(I[:, :, i], r) / base for i in range(3)]
    # p filtered with the mean filter
    mean_p = boxfilter(p, r) / base
    # filter I with p then filter it with the mean filter
    means_IP = [boxfilter(I[:, :, i] * p, r) / base for i in range(3)]
    # covariance of (I, p) in each local patch
    covIP = [means_IP[i] - means[i] * mean_p for i in range(3)]

    # variance of I in each local patch: the matrix Sigma in ECCV10 eq.14
    var = defaultdict(dict)
    for i, j in combinations_with_replacement(range(3), 2):
        var[i][j] = boxfilter(
            I[:, :, i] * I[:, :, j], r) / base - means[i] * means[j]

    a = np.zeros((M, N, 3))
    for y, x in np.ndindex(M, N):
        #         rr, rg, rb
        # Sigma = rg, gg, gb
        #         rb, gb, bb
        Sigma = np.array([[var[R][R][y, x], var[R][G][y, x], var[R][B][y, x]],
                          [var[R][G][y, x], var[G][G][y, x], var[G][B][y, x]],
                          [var[R][B][y, x], var[G][B][y, x], var[B][B][y, x]]])
        cov = np.array([c[y, x] for c in covIP])
        a[y, x] = np.dot(cov, inv(Sigma + eps * np.eye(3)))

    b = mean_p - a[:, :, R] * means[R] - \
        a[:, :, G] * means[G] - a[:, :, B] * means[B]

    q = (boxfilter(a[:, :, R], r) * I[:, :, R] + boxfilter(a[:, :, G], r) *
         I[:, :, G] + boxfilter(a[:, :, B], r) * I[:, :, B] + boxfilter(b, r)) / base

    return q
```