Shri Ramdeobaba College of Engineering and Management, Nagpur

Department of Electronics Engineering

Digital Image Processing (ENT 355-3)

Name: Prajwal Pandurang Shette

Roll No: B1- 12

Experiment No: 08

Aim: Implementation of following White Balancing algorithms for Image Enhancement i) White Patch Algorithm

ii) Gray-world Algorithm

iii) Ground-truth Algorithm.

Theory: White Balancing

Firstly, what is white balancing (WB)? It is a color correcting process of removing unrealistic color casts, so that objects which appear white in person are correctly rendered white in your desired image. We will be implementing three white balancing techniques, these are:

1. White Patch Algorithm

This approach is typical of the Color Constancy adaptation where it searches for the lightest patch to use as a white reference similar to how the human visual system does. Note that for white to be observed in the image, each channel in you RGB color space should be at its maximum value.

2. Gray-world Algorithm

The gray-world algorithm is a white balance method that assumes that your image, on average, is a neutral gray. Gray-world assumption hold if we have a good distribution of colors in the image. Considering this assumption as true, the average reflected color is assumed to be the color of the light. Therefore, we can estimate the illumination color cast by looking at the average color and comparing it to gray.

3. Ground-truth Algorithm

So far, we have made assumptions on how the color spaces behave on our images. Now, instead of making assumptions for enhancing our images, we will select a patch (portion of an image) and use that patch to recreate our desired image.

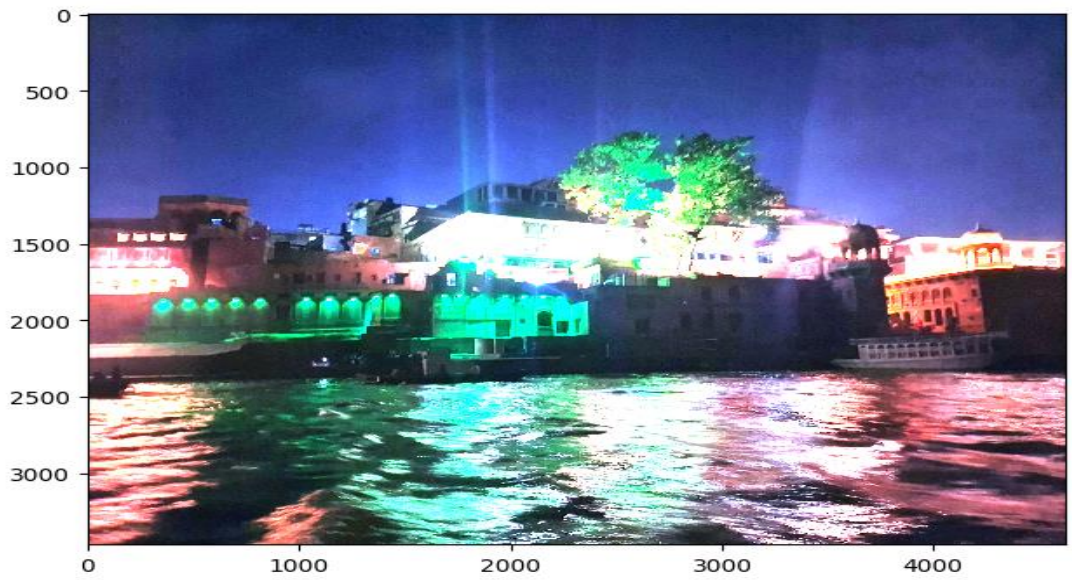
Code:

1. White Patch Algorithm

```
import numpy as np
import matplotlib.pyplot as plt
import skimage.io

lily = skimage.io.imread('farm.jpeg')

#White Patch
def white_patch(image, percentile=100):
    """
    White balance image using White patch algorithm
    Parameters
    ----------
    image : numpy array
            Image to white balance
    percentile : integer, optional
                 Percentile value to consider as channel maximum
    Returns
    -------
    image_wb : numpy array
```

```
                White-balanced image
    """
    white_patch_image = skimage.img_as_ubyte((image*1.0 /
                                    np.percentile(image,percentile,
                                    axis=(0, 1))).clip(0, 1))
    return white_patch_image
#call the function to implement white patch algorithm
skimage.io.imshow(white_patch(lily, 85))
```

Output:



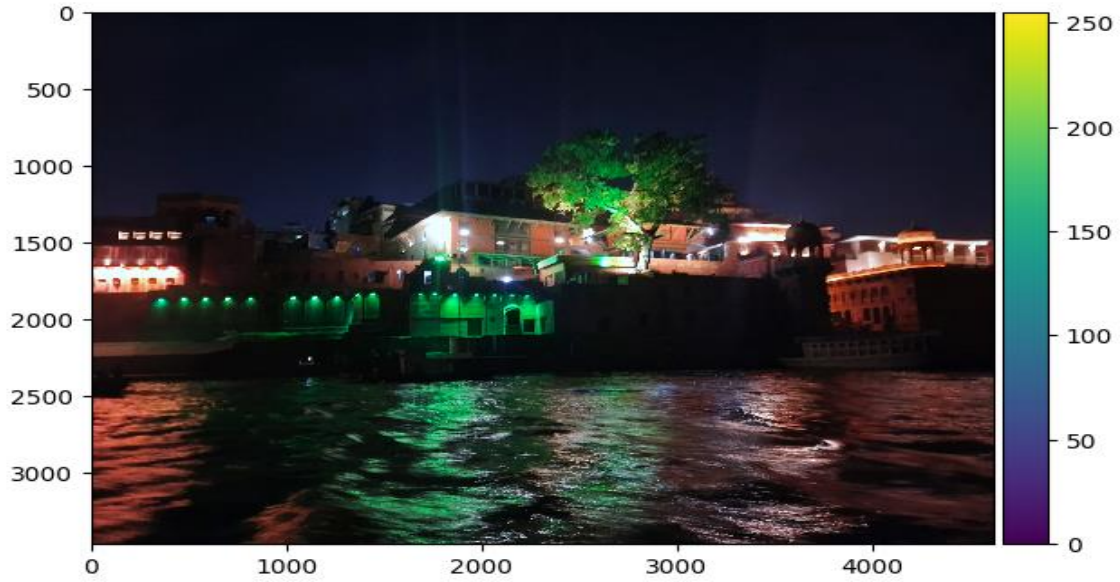
2. Gray-world Algorithm

Code

```
# 1.White Patch Algorithm
import skimage as sk
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
# from skimage import img_as_ubyte
from skimage.util import img_as_uint
from skimage.util import img_as_float
```

```
# lily = img = imread('lily.jpg')
lily = sk.io.imread('lily.jpg')
def gray_world(image):
    """
    White balance image using Gray-world algorithm
    Parameters
    ----------
    image : numpy array
            Image to white balance

    Returns
    -------
    image_wb : numpy array
               White-balanced image
    """
    image_grayworld = ((image * (sk.access.mean() /
                      image.mean(axis=(0,1)))).
                      clip(0,255).astype(int))
    # for images having a transparency channel

    if image.shape[2] == 4:
        image_grayworld[:,:,3] = 255
    return image_grayworld
#call the function to implement gray world algorithm
sk.io.imshow(gray_world(lily))
```

Output:



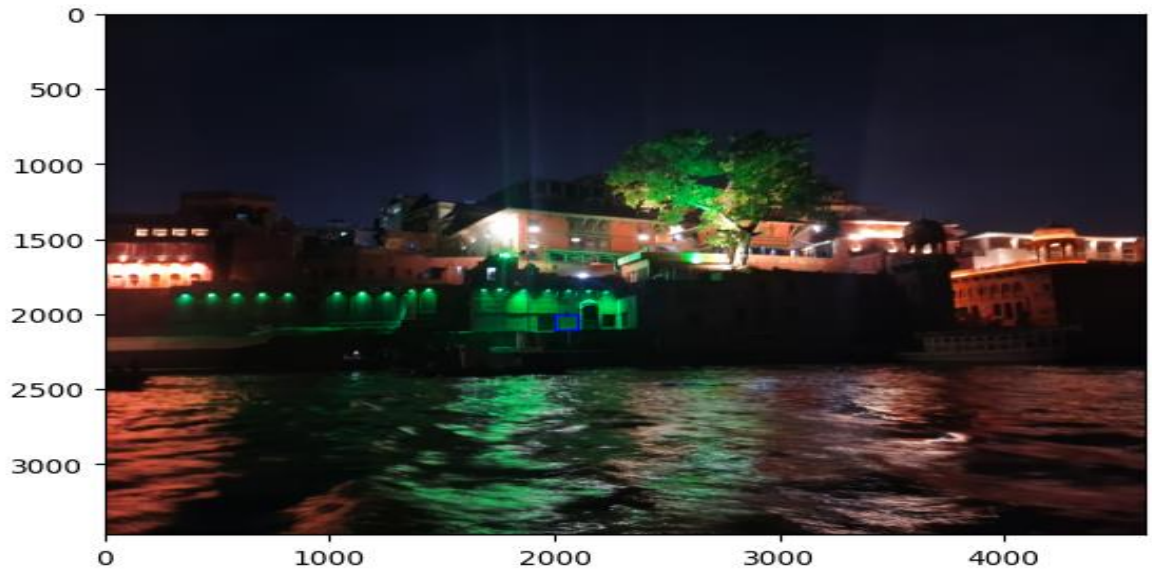
3. Ground-truth Algorithm

Code:

```
import skimage as sk
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
from skimage import img_as_ubyte

# lily = img = imread('lily.jpg')
lily = sk.io.imread('lily.jpg')
from matplotlib.patches import Rectangle
fig, ax = plt.subplots()
ax.imshow(lily)
ax.add_patch(Rectangle((650, 550), 100, 100, edgecolor='b',
facecolor='none'));


def ground_truth(image, patch, mode='mean'):
    """
    White balance image using Ground-truth algorithm
    Parameters
    ----------
    image : numpy array
            Image to white balancr
    patch : numpy array
            Patch of "true" white
    mode : mean or max, optional
           Adjust mean or max of each channel to match patch

    Returns
    -------

    image_wb : numpy array
               White-balanced image
    """
    image_patch = img_patch
    if mode == 'mean':
        image_gt = ((image * (image_patch.mean() / \
                    image.mean(axis=(0, 1))))\
                    .clip(0, 255)\
                    .astype(int))
    if mode == 'max':
        image_gt = ((image * 1.0 / image_patch.max(axis(0,1))).clip(0, 1))
    #transparency channel
```

```
    if image.shape[2] == 4:
        image_gt[:,:,3] = 255
    return image_gt

sk.io.imshow(ground_truth(lily, img_patch, 'max'))
```

Output:



Observation & Conclusion:

White Patch: As observed, it can be seen that the image became relatively brighter with the lily at the middle becoming very vibrant. This is how the white patch algorithm enhanced our image. Now, let us see the next algorithm.

Gray World: Seeing the image, it can be observed that it did not deviate that much from the original one. One reason for this might be that the average color and its comparison to gray is not that significant. Then let us see the last algorithm.

Ground truth: The output is slightly closer to the white patch output but the latter is brighter. It also emphasized the color of the lily but instead of highlighting the color of the pads, it only brightened it.For the ground truth algorithm, the output image depends greatly on the choice of the patch image. So, choose the patch wisely by visualizing what kind of enhanced image you would want to arrive at.