Monadic and Diadic Image Processing

By:
Ahmad Nadeem Saigol

Contents

Note:

- The assignment is implemented in both languages - Python and MATLAB – using the same strategy discussed in this report. However, some values had to be readjusted slightly because of the way, the language handles the image underneath. (for example, MATLAB does not crop the image if rotated which is not the case with Python).
- The report provides results of MATLAB code only, but results of Python Code can be found in the respective folder, provided with this report.
- The report also provides complete codes for both languages. However, it is recommended to read them in their respective editor (.m and .ipynb files are provided)
- Python Code was implemented using Google Colab. To be able to run it on other environments, it may require some refactoring.
- Some Images have been omitted due to privacy reasons.

Part A:

Image Preprocessing:

Before applying the operation on the image of human, the picture is mapped to range [0,1] (in MATLAB only) and resized from [3024 4032 3] to [800 1200 3]. A grayscale image is also stored.

Histogram Equalization:

This operator basically transforms the image in such a way that the new image has almost flat histogram of number of pixel values. For color image, this operator was applied on each channel separately and was merged together afterwards.

MATLAB:

- Imhist() / plot() > for plotting histograms.
- Histeq() > Applies histogram equalization.

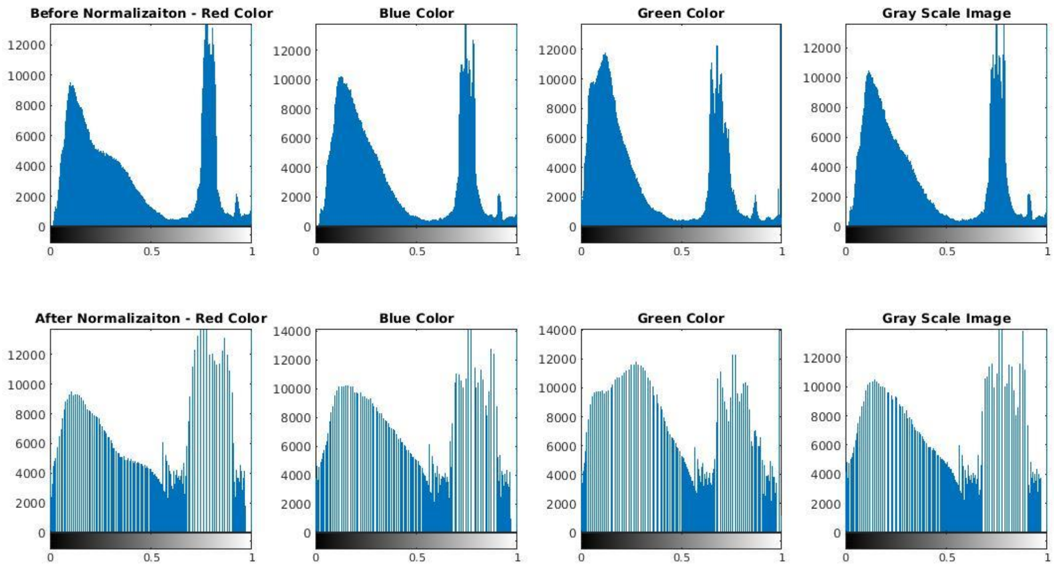
Python:

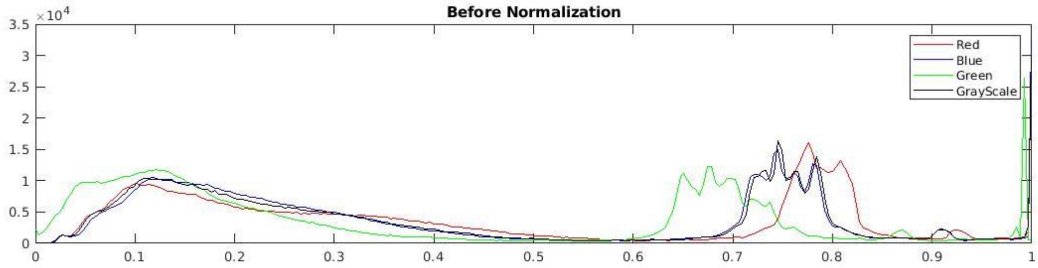
- calcHist() / plt.hist() > for calculating and plotting histograms
- equalizeHist > Applies histogram equalization

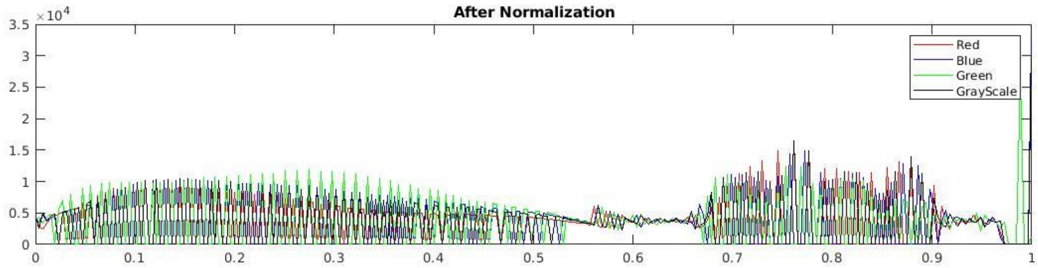
A comparison between before and after normalization is given below:

Result:

(image omitted)







Brightness:

It involves addition of a small number ‘b’ to each pixel value in each channel. On a scale of [0, 1], a value of 0.3, 0.5 and 0.2 was added to Blue, Green and Red channel, respectively and for grayscale, the value was 0.4.

$f(x) = x + b$

It was achieved through addition operator ‘+’ in both languages.

Result:

(image omitted)

Contrast:

It involves the multiplication of a scalar number ‘a’ with each pixel value in each channel. The value of ‘a’ for Red, Green, Blue and Grayscale channel was 1.2, 1.8, 2.1 and 1.5, respectively.

$f(x)=a*x$

It was achieved through multiplication operator ‘*’ in both languages.

Result:

(image omitted)

Gamma Correction:

It introduces nonlinearity in the voltage levels/output of ADC to mimic human eye functioning. The function applied is

f(x) = x^(gamma)

The value of gamma was 1/2.4 for color image and 1/2.2 for grayscale.
It was achieved through ‘np.power()’ in Python and operator ‘.^’ in MATLAB.

Result:

(image omitted)

Negative:

It involves inversion of the function i.e., 0 becomes white/color and 1 becomes black.

f(x) = 1- x

It was achieved through addition operator ‘-’ in both languages.

Result:

(image omitted)

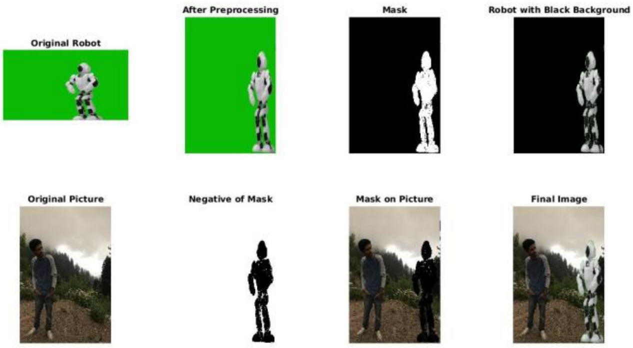
Part B:

Diadic Image Operation:

In this type of image processing, two images are operated with each other to create a new image. Following strategy was adopted to merge the robot with human:

- To make it look like as if the robot is moving towards the human, the robot was first moved towards right by 150 pixels using affine transformation and then afterwards projective transformation was applied with values [-0.0001, 0.00005, 1.3] in its last row.
- Then the image of the robot was resized to [800 1200 3] to meet the dimension of the image of the human.
- Afterwards, the mask of the robot was created as binary image. To do this, the robot is first converted to grayscale using ‘rgb2gray()’ in MATLAB and ‘cvtColor()’ in Python. Then, the mask was obtained from this image by setting a range for which the value assigned to the image was zero (removal of background) and outside that range, was one (retain robot).
- This binary image was then multiplied with original image of robot to obtain the image of the robot with black background.
- The mask created before was inverted and multiplied with the image of the human.
- The final image was obtained by adding image obtained in last step with robot with black background.

Result:



Code:

MATLAB:

```
clear all

f1=figure;
f2=figure;
f3=figure;
f4=figure;
f5=figure;
f6=figure;
f7=figure;
f8=figure;

% ------Read and Display Image -----

% read the picture
mypicture=imread('CV_Assign2/mypicture.jpg');

%map the intensity values in range [0 1]
mypicture=im2double(mypicture);

%Resizing the image
mypicture=imresize(mypicture,[1200, 800]);

%convert image to grayscale
mypicture_gray=rgb2gray(mypicture);

%display both images (RGB and Grayscale)
figure(f1)

subplot(1,4,1)
imshow(mypicture)
title('Colored Image - Original')

subplot(1,4,3)
imshow(mypicture_gray)
title('Gray Scaled Image - Original')

% ------ Histogram Normalization -----
```

```
figure(f2)

subplot(2,4,1)
imhist(mypicture(:,:,1))
title('Before Normalizaiton - Red Color')

subplot(2,4,2)
imhist(mypicture(:,:,2))
title('Blue Color')

subplot(2,4,3)
imhist(mypicture(:,:,3))
title('Green Color')

subplot(2,4,4)
imhist(mypicture_gray)
title('Gray Scale Image')


%another way to visualise distributions

[Rcount, Rloc]=imhist(mypicture(:,:,1));
[Bcount, Bloc]=imhist(mypicture(:,:,2));
[Gcount, Gloc]=imhist(mypicture(:,:,3));
[Blcount, Blloc]=imhist(mypicture_gray);

figure(f3)
subplot(2,1,1)
plot(Rloc, Rcount, 'r',Bloc, Bcount, 'b', Gloc, Gcount, 'g', Blloc, Blcount, 'k');
title('Before Normalization')
legend('Red', 'Blue','Green','GrayScale')

% Applying histogram equalization
%colored image
R=histeq(mypicture(:,:,1),256);
B=histeq(mypicture(:,:,2),256);
G=histeq(mypicture(:,:,3),256);
picture_hist=cat(3,R,G,B);

%grayscale image
picture_hist_gray=histeq(mypicture_gray,256);

%display both images (RGB and Grayscale)
figure(f1)

subplot(1,4,2)
imshow(picture_hist)
title('Normalized')

subplot(1,4,4)
imshow(picture_hist_gray)
title('Normalized')

figure(f2)

subplot(2,4,5)
imhist(R)
title('After Normalizaiton - Red Color')
```

```
subplot(2,4,6)
imhist(B)
title('Blue Color')

subplot(2,4,7)
imhist(G)
title('Green Color')

subplot(2,4,8)
imhist(picture_hist_gray)
title('Gray Scale Image')


%another way to visualise distributions

[Rcount, Rloc]=imhist(R);
[Bcount, Bloc]=imhist(B);
[Gcount, Gloc]=imhist(G);
[Blcount, Blloc]=imhist(picture_hist_gray);

figure(f3)
subplot(2,1,2)
plot(Rloc, Rcount, 'r',Bloc, Bcount, 'b', Gloc, Gcount, 'g', Blloc, Blcount, 'k');
title('After Normalization')
legend('Red', 'Blue','Green','GrayScale')

% ------ Brightness ------
%colored
picture_bright=mypicture+reshape([0.2, 0.5, 0.3], [1 1 3]);

%grayscale
picture_bright_gray=mypicture_gray+0.4;

%display results

figure(f4)

subplot(1,4,1)
imshow(mypicture)
title('Colored Image - Original')

subplot(1,4,3)
imshow(mypicture_gray)
title('Gray Scaled Image - Original')

subplot(1,4,2)
imshow(picture_bright)
title('Increased Brightness (0.2,0.5,0.3)')

subplot(1,4,4)
imshow(picture_bright_gray)
title('Increased Brightness (0.4)')

% ------- Contrast ----
%colored
picture_con=mypicture.*reshape([1.2, 1.8, 2.1], [1 1 3]);

%grayscale
picture_con_gray=mypicture_gray.*1.5;
```

```
%display results

figure(f5)

subplot(1,4,1)
imshow(mypicture)
title('Colored Image - Original')

subplot(1,4,3)
imshow(mypicture_gray)
title('Gray Scaled Image - Original')

subplot(1,4,2)
imshow(picture_con)
title('Contrast (1.2,1.8,2.1)')

subplot(1,4,4)
imshow(picture_con_gray)
title('Contrast (1.5)')

% -------Gamma Correction -----
%colored
picture_gam=mypicture.^(1/2.4);

%grayscale
picture_gam_gray=mypicture_gray.^(1/2.2);

%display results

figure(f6)

subplot(1,4,1)
imshow(mypicture)
title('Colored Image - Original')

subplot(1,4,3)
imshow(mypicture_gray)
title('Gray Scaled Image - Original')

subplot(1,4,2)
imshow(picture_gam)
title('Gamma Correction (1/2.4)')

subplot(1,4,4)
imshow(picture_gam_gray)
title('Gamma Correction (1/2.2)')

% ----- Negative ----
%colored
picture_neg=1-mypicture;

%grayscale
picture_neg_gray=1-mypicture_gray;

%display results

figure(f7)
```

```
subplot(1,4,1)
imshow(mypicture)
title('Colored Image - Original')

subplot(1,4,3)
imshow(mypicture_gray)
title('Gray Scaled Image - Original')

subplot(1,4,2)
imshow(picture_neg)
title('Negative')

subplot(1,4,4)
imshow(picture_neg_gray)
title('Negative')

% ------ Diadic Image Operation -----

%read the robot image
robot=imread('CV_Assign2/robot.jpg');

%map the image to range [0, 1]
robot=im2double(robot);

figure(f8)
subplot(2,4,1)
imshow(robot)
title('Original Robot')

%temporary variable used for translating the robot
temp=zeros(size(robot));
temp=temp+robot(1,1,:);

%translation of robot
transl_transf=affine2d([1 0 0; 0 1 0; 150 0 1]);
temp(70:end,350+150:560+150,:)=imwarp(robot(70:end,350:560,:),transl_transf);
robot=temp;

%projective transformation
rotate_transf_robot=projective2d([1 0 -0.0001; 0 1 0.00005; 0 0 1.3]);
robot=imwarp(robot, rotate_transf_robot, 'FillValues', reshape(robot(1,1,:),[1 3]));

%resize the image
robot=imresize([zeros(30,578,3)+robot(1,1,:); robot], [1200 800]);

subplot(2,4,2)
imshow(robot)
title('After Preprocessing')

%RGB to gray conversion
gray_robot=rgb2gray(robot);

subplot(2,4,3)
imshow(gray_robot)
title('Grayscale Robot')

%dims
s=size(gray_robot);
```

```
%mask creation
mask=gray_robot;


for j=1:s(1)
   for k=1:s(2)
      if ((gray_robot(j,k)<=(0.47)) && (gray_robot(j,k)>=(0.4)))
        mask(j,k)=0;
      else
        mask(j,k)=1;
      end
   end
end

mask(:,:,2)=mask;
mask(:,:,3)=mask(:,:,1);

subplot(2,4,3)
imshow(mask)
title('Mask')

%creation of robot with black background
robot_black_bg=(robot.*mask);

subplot(2,4,4)
imshow(robot_black_bg)
title('Robot with Black Background')

%negative of mask
maskforImg=1-mask;

subplot(2,4,5)
imshow(mypicture)
title('Original Picture')

subplot(2,4,6)
imshow(maskforImg)
title('Negative of Mask')

%Mask imprinting on the picture
imgWithMask=mypicture.*maskforImg;

subplot(2,4,7)
imshow(imgWithMask)
title('Mask on Picture')

%Finalizing image
finalImg=imgWithMask+robot_black_bg;

subplot(2,4,8)
imshow(finalImg)
title('Final Image')
```

Python:

```
import cv2
```

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from google.colab.patches import cv2_imshow

#-------Read and Display Images -----

human=cv2.imread('/content/mypicture.jpg')
human=cv2.resize(human, (800,1200))
cv2_imshow(human)

robot=cv2.imread('/content/robot.jpg')
cv2_imshow(robot)

human_gray=cv2.cvtColor(human, cv2.COLOR_BGR2GRAY)
cv2_imshow(human_gray)

robot_gray=cv2.cvtColor(robot, cv2.COLOR_BGR2GRAY)
cv2_imshow(robot_gray)

#-------Histograms-----

#Colored
color = ('b','g','r')
for i,col in enumerate(color):
  histr = cv2.calcHist([human],[i],None,[256],[0,256])
  plt.plot(histr,color = col)
  plt.xlim([0,256])
plt.show()

#Colored
for i,col in enumerate(color):
  plt.hist(human[:,:,i].ravel(),256,[0,256]);
  plt.title(col)
plt.show()

#Grayscale
hist_gray = cv2.calcHist([human_gray],[0],None,[256],[0,256])
plt.plot(histr,color = 'k')
plt.xlim([0,256])
plt.title('gray')
plt.show()

#GrayScale
plt.hist(human_gray.ravel(),256,[0,256]);
plt.title('gray')
plt.show()

#---------Histogram Equalization-------

#Colored
```

```
human_hist_norm=np.zeros_like(human)
for i,col in enumerate(color):
  human_hist_norm[:,:,i]=cv2.equalizeHist(human[:,:,i])
  histr = cv2.calcHist([human_hist_norm],[i],None,[256],[0,256])
  plt.plot(histr,color = col)
  plt.xlim([0,256])
plt.show()

#Colored
for i,col in enumerate(color):
  plt.hist(human_hist_norm[:,:,i].ravel(),256,[0,256]);
  plt.title(col)
plt.show()

#Colored Image Obtained after normalization
cv2_imshow(human_hist_norm)

#GrayScale
human_gray_hist_norm=cv2.equalizeHist(human_gray)
hist_gray = cv2.calcHist([human_gray_hist_norm],[0],None,[256],[0,256])
plt.plot(histr,color = 'k')
plt.xlim([0,256])
plt.title('gray')
plt.show()

#GrayScale
plt.hist(human_gray_hist_norm.ravel(),256,[0,256]);
plt.title('gray')
plt.show()

#Gray Scale Image Obtained after normalization
cv2_imshow(human_gray_hist_norm)

#--------Brightness--------

#Colored
human_bright = human + np.array([0.3*255, 0.5*255, 0.2*255]).reshape(1,1,3)
cv2_imshow(human_bright)

#GrayScale
human_gray_bright = human_gray + 0.4*255
cv2_imshow(human_gray_bright)

#----------Contrast-------

#Colored
human_con = human * np.array([2.1, 1.8, 1.2]).reshape(1,1,3)
cv2_imshow(human_con)

#GrayScale
human_gray_con = human_gray *1.5
```

```python
cv2_imshow(human_gray_con)

#-------- Gamma Correction -----

#Colored
human_gamma = np.power(human/255, (1/2.4))*255
cv2_imshow(human_gamma)

#GrayScale
human_gray_gamma = np.power(human_gray/255, (1/2.2))*255
cv2_imshow(human_gray_gamma)

#----------Negative-------

#Colored
human_neg = 1 - human
cv2_imshow(human_neg)

#GrayScale
human_gray_neg = 1 - human_gray
cv2_imshow(human_gray_neg)

#----------Diadic Image Processing-------

#Translation of Robot
height, width=robot.shape[:2]
b,g,r=robot[0,0,:]
T=np.float32([[1, 0, 150], [0, 1, 0]])
trans_robot=cv2.warpAffine(robot, T, (width, height), borderValue=(b.item(),
 g.item(), r.item()))

cv2_imshow(trans_robot)

#Projective Transforamtion
T=np.float32([[1, 0, 150], [0, 1, 90],[-0.0001, 0, 1.3]])
presp_robot=cv2.warpPerspective(trans_robot, T, (width, height), borderValue
=(int(b),int(g),int(r)))

cv2_imshow(presp_robot)

#Resize the image
proccessed_robot=cv2.resize(presp_robot, (800,1200))

cv2_imshow(proccessed_robot)

#Mask Creation
robot_gray=cv2.cvtColor(proccessed_robot, cv2.COLOR_BGR2GRAY)
height, width=robot_gray.shape
mask=robot_gray

for k in range(1200):
```

```
  for l in range(800):
    value = robot_gray.item(k,l)
    #'mask' contains only zero and one. This is done for its easier multipli
cation with robot image
    if value >= 109 and value <= 125:
      mask[k,l]=0
    else:
      mask[k,l]=1

cv2_imshow(mask*255) #upscaling for visualization

#Robot with Black Background
mask_merged=cv2.merge((mask,mask,mask))
robotBlackbg=np.multiply(proccessed_robot, mask_merged)

cv2_imshow(robotBlackbg)

#Negative of Mask
negMask=1-mask_merged
cv2_imshow(negMask*255) #upscaling for visualization

#Mask imprinting on picture
imgwithMask=human*negMask
cv2_imshow(imgwithMask)

#Final Image
final = imgwithMask + robotBlackbg

cv2_imshow(final)
```