BME 646/ ECE695DL: Homework 2

Shaswata Roy

24 January 2022

1 Introduction

This assignment mostly deals with image manipulation using torch and PIL.

2 Methodology

2 pictures were taken of a glass from separate viewpoints. The images were then manipulated using package functions and then the histograms were compared.

3 Implementation and Results

```
from PIL import Image
import PIL
import torch
import matplotlib.pyplot as plt
import numpy as np
from typing import List
from torchvision import transforms
from scipy.stats import wasserstein_distance
import torchvision.transforms.functional as TF

#Answer 2
im = Image.open("1.jpg")
plt.imshow(im)

#Answer 3
convert = transforms.Compose([ transforms.Normalize([0.5,0.5,0.5],[0.5,0.5,0.5]) ])

def RGB_hist(rgb_tensor: torch.Tensor)-> List[torch.Tensor]:
  r_tensor = rgb_tensor[0]
  g_tensor = rgb_tensor[1]
  b_tensor = rgb_tensor[2]
  hist_r = torch.histc(r_tensor, bins = 10, min = -1.0, max = 1.0)
  hist_g = torch.histc(g_tensor, bins = 10, min = -1.0, max = 1.0)
  hist_b = torch.histc(b_tensor, bins = 10, min = -1.0, max = 1.0)

  hist_r = hist_r.div(hist_r.sum())
  hist_g = hist_g.div(hist_g.sum())
  hist_b = hist_b.div(hist_b.sum())

  return [hist_r,hist_g,hist_b]


x = np.linspace(-1,0.8,10)
plt.figure(figsize=(25,20))

tensor_imgs = []
histTensor = []
for i in range(1,3):
  im = Image.open(str(i)+".jpg")
  tensor_im = transforms.ToTensor()(im)
  tensor_im = convert(tensor_im)
  tensor_imgs.append(tensor_im)

  [hist_r,hist_g,hist_b] = RGB_hist(tensor_im)
  histTensor.append([hist_r,hist_g,hist_b])

  plt.subplot(4,4,4*i+1)
  plt.imshow(im)

  plt.subplot(4,4,4*i+2)
  plt.bar(x,hist_r,width=0.2,align='edge',color='r',alpha=0.3)

  plt.subplot(4,4,4*i+3)

  plt.bar(x,hist_g,width=0.2,align='edge',color='g',alpha=0.3)

  plt.subplot(4,4,4*i+4)

  plt.bar(x,hist_b,width=0.2,align='edge',color='b',alpha=0.3)

#Answer 4
for ch in range(3):
  dist = wasserstein_distance( torch.squeeze( histTensor[0][ch] ).cpu().numpy(),torch.squeez
  print("\n Wasserstein distance for channel: ", dist)

#Answer 5
import torchvision.transforms.functional as TF
```

```
plt.figure(figsize=(8,6))

affine_transformer = TF.affine(tensor_imgs[1],angle=-25,translate=(-50.,200.),scale=1.,shear
cropped = TF.crop(tensor_imgs[0],800,500,2000,3000)
affine_transformer = TF.crop(affine_transformer,800,500,2000,3000)
for i in range(-40,0,5):
  affine_transformer = TF.affine(tensor_imgs[1],angle=i,translate=(-50.,200.),scale=1.,shear
  cropped = TF.crop(tensor_imgs[0],800,500,2000,3000)
  affine_transformer = TF.crop(affine_transformer,800,500,2000,3000)
  histTensor[0] = RGB_hist(cropped)

  histTensor[1] = RGB_hist(affine_transformer)

  dist = 0

  for ch in range(3):
    dist = dist+wasserstein_distance( torch.squeeze( histTensor[0][ch] ).cpu().numpy(),torch
  print("\n Angle ", i," Total Wasserstein distance ",dist)

affine_transformer = TF.affine(tensor_imgs[1],angle=-25,translate=(-50.,200.),scale=1.,shear
cropped = TF.crop(tensor_imgs[0],800,500,2000,3000)
affine_transformer = TF.crop(affine_transformer,800,500,2000,3000)
plt.imshow(cropped[0],alpha=0.5,cmap='gray')
plt.imshow(affine_transformer[0],alpha=0.3,cmap='gray')

#Answer 6
for x in range(-50,200,50):
  affine_transformer = TF.affine(tensor_imgs[1],angle=-25,translate=(-50.,200.),scale=1.,she
  perspective_transformer = TF.perspective(affine_transformer,startpoints=[[500,500],[500,25
  perspective_transformer= TF.crop(perspective_transformer,800,500,2000,3000)

  cropped = TF.crop(tensor_imgs[0],800,500,2000,3000)
  #plt.imshow(cropped[0],alpha=0.5,cmap='gray')
  #plt.imshow(perspective_transformer[0],alpha=0.3,cmap='gray')

  histTensor[0] = RGB_hist(cropped)

  histTensor[1] = RGB_hist(perspective_transformer)

  dist = 0

  for ch in range(3):
    dist = dist+wasserstein_distance( torch.squeeze( histTensor[0][ch] ).cpu().numpy(),torch
  print("\n Perpective Change", x," Total Wasserstein distance ",dist)
```

```
x=50
affine_transformer = TF.affine(tensor_imgs[1],angle=-25,translate=(-50.,200.),scale=1.,shear
perspective_transformer = TF.perspective(affine_transformer,startpoints=[[500,500],[500,250
perspective_transformer= TF.crop(perspective_transformer,800,500,2000,3000)

cropped = TF.crop(tensor_imgs[0],800,500,2000,3000)
plt.imshow(cropped[0],alpha=0.5,cmap='gray')
plt.imshow(perspective_transformer[0],alpha=0.3,cmap='gray')
```

While changing the affine parameters the translation parameter was fine tuned manually while the angle parameter was setup to different values using a loop. The best angle was found out using the total Wasserstein distance (sum over RGB). The angle found through this method was 30 degrees while just by sight it seemed to be more like 25 degrees which was close.

Note: While comparing the 2 images had to be cropped to make sure we don't compare the dark pixels which appear when we rotate an image.

Finally the perspective parameter was changed. The left side was increased by a specific amount using a loop and this increase was evaluated using the total Wasserstein distance. The best increase was found to be 50px which coincided with what was seen.

In both the cases the best fit parameters have been used to visualize the 2 images on top of each other.(It is black and white as only the green color has been used to compare the 2 images).

4 Lessons Learned

- How to use functions from PIL to import images
- Convert PIL images to torch tensors
- The use of Wasserstein distance in comparing the histogram of 2 images
- How to manipulate images using affine and perspective
- Learned to use crop on top of all these methods as the rotation was introducing black pixels which was affecting the Wasserstein distance

5 Suggested Enhancements

The assignment could have been a bit more specific and detailed.