Instruction on run our program:

- To run the code please extract the zip file and find the code inside it as zip file is also containing other resources like pickle files and Images which are needed by the code.

- We are using the PKL file to store the Gaussian mask and Prewitt operators (provided inside the Pickle folder) which are again loaded with the help of PICKLE library. So, make sure to have Pickle folder uploaded by us along with the code script while running the code.

- We have split the Test and Training images into further 2 categories(directories) namely Test which contains Test Images (pos), Test Images (neg) and Train which contains Training Images (pos) and Training Images (neg).

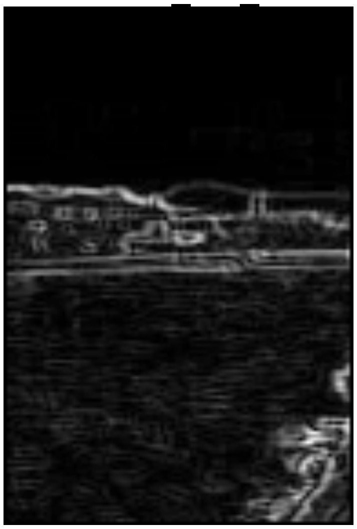
- The code is given in form of an IPYNB file and a PY file which both are having same code.

- We are using libraries like cv2, pickle, NumPy, math and Image.

Normalized Gradient Magnitude Images (Test Images (Negative)):

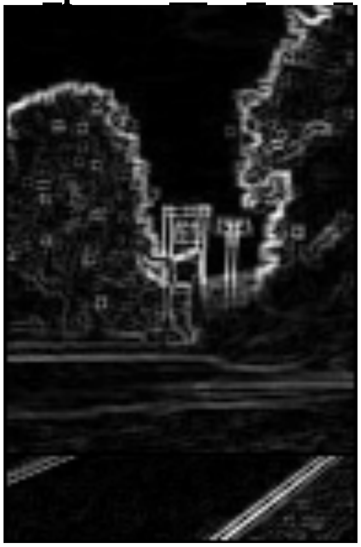
1. **00000003a_cut_Gred.bmp**

2. **00000090a_cut_Gred.bmp**

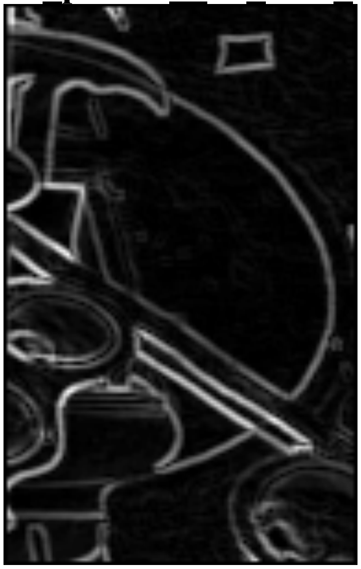


3. **00000118a_cut_Gred.bmp**

4. **no_person__no_bike_258_Cut_Gred.bmp**

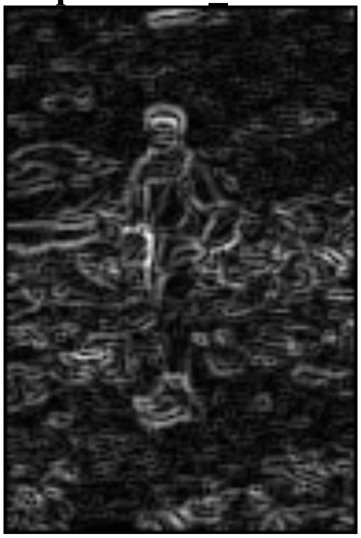


5. **no_person__no_bike_264_cut_Gred.bmp**



Normalized Gradient Magnitude Images (Test Images (Positive)):

6. **crop001034b_Gred.bmp**

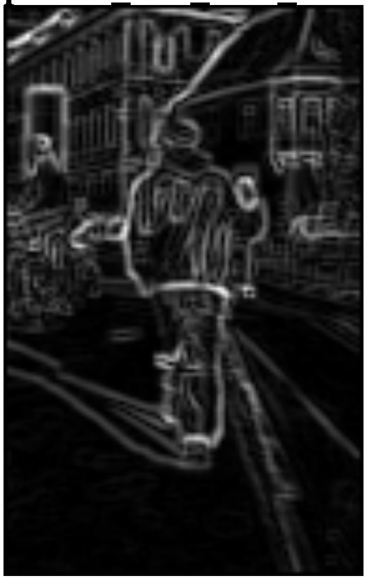


7. **crop001070a_Gred.bmp**

8. **crop001278a_Gred.bmp**

9. **crop001500b_Gred.bmp**

10.person_and_bike_151a_Gred.bmp



Classification results of Images:

Test image	Correct Classification	File name of 1st NN, Distance & Classification	File name of 2nd NN, Distance & Classification	File name of 3rd NN, Distance & Classification	Classification from 3-NN
crop001034b	Human	0.6601 & No-human	0.6441 & Human	0.6315 & Human	Human
crop001070a	Human	0.5021 & Human	0.5008 & Human	0.4982 & Human	Human
crop001278a	Human	0.5823 & Human	0.4732 & Human	0.4619 & Human	Human
crop001500b	Human	0.4928 & No-Human	0.4501 & Human	0.4206 & Human	Human
person_and_bike_151a	Human	0.5445 & Human	0.5256 & Human	0.5253 & Human	Human
00000003a_cut	No-human	0.5230 & No-human	0.5217 & No-human	0.5071 & Human	No-human
00000090a_cut	Ho-human	0.5406 & No-human	0.4341 & No-human	0.4232 & Human	No-human
00000118a_cut	No-human	0.5289 & No-human	0.5125 & No-human	0.5107 & No-human	No-human
no_person_no_bike_258_cut	No-human	0.4302 & Human	0.4237 & Human	0.4164 & Human	Human
no_person_no_bike_264_cut	No-human	0.4349 & No-human	0.4192 & No-human	0.4048 & No-human	No-human

In [1]:

```
import pickle as pkl
import numpy as np
import cv2
import math
import glob
from PIL import Image as im
```

In [2]:

```
# This function is used to load Masks from .pkl files
def loadMask(path,lable):
    with open(path, 'rb') as f:
        mask = pkl.load(f)
        return mask
```

In [3]:

```
def display_Img(img,lable="Img"): # function to Display an Image
    img=np.uint8(img)
    cv2.imshow(lable,img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    #data = im.fromarray(img) # uncomment to the save Images
    #data.save(lable+'.bmp')  # uncomment to the save Images
```

In [4]:

```
# This function loads image and converts 3 channel image to single channel grayscale image
def load_img(path):
    img=cv2.imread(path)
    h,w = img.shape[:2]
    gray = np.zeros((h,w), np.uint8)
    for i in range(h):
        for j in range(w):
            gray[i,j] = np.clip(0.114 * img[i,j,0]  + 0.587 * img[i,j,1] + 0.299 * img[i,j,2], 0, 255)
    return gray
```

In [5]:

```
# This function is used to apply operators like prewitt,Robert on an Image
def apply_mask(oldImg,mask,pedding=0):
    ( h , w ) = oldImg.shape
    newImg = np.zeros( ( h , w ), dtype = np.int32 )
    P = ( mask[0].size//2 )
    for j in range( (P + pedding) , h - (P + pedding) ):
        for i in range( (P + pedding) , w - (P + pedding) ):
            value = 0
            for k in range( -P , P+1 ):
                for l in range( -P , P+1 ):
                    value += mask[k+P][l+P] * oldImg[j+k][i+l]
            newImg[j][i] = value
    return ( newImg , P + pedding )
```

In [6]:

```
# In Normalization, pixel values are rescale to 0-255 range
def Normalization(img):
    img=np.absolute(img)
    img=img/img.max()*255
    return img
```

In [7]:

```
# Function to calculate gradient magnitude from horizontal and vertical gradient
def gradient_magnitude(gx,gy):
    ( h , w ) = gx.shape
    grad_mag = np.zeros( ( h , w ), dtype = np.uint32 )
    for j in range(h):
        for i in range(w):
            grad_mag[j][i]= abs(gx[j][i])+abs(gy[j][i])
    return grad_mag
```

In [8]:

```
# Function to calculate gradient angle from horizontal and vertical gradient
def gradient_angle(gx,gy):
    ( h , w ) = gx.shape
    grad_ang = np.zeros( ( h , w ), dtype = np.float32 )
    for j in range(h):
        for i in range(w):
            if gx[j][i] != 0:
                grad_ang[j][i]= math.degrees( math.atan(gy[j][i]/gx[j][i]) )
    return grad_ang
```

In [9]:

```
def Gradient_Operation( Image , pedding=0 ):
    # loading Prewitt X derivative from .pkl file
    Prewitt_X = loadMask('Pickle/prewitt-x.pkl','Prewitt X derivative')
    # loading Prewitt Y derivative from .pkl file
    Prewitt_Y = loadMask('Pickle/prewitt-y.pkl','Prewitt Y derivative')
    (Gx , _ ) = apply_mask(Image,Prewitt_X,pedding) # Calculating horizontal gradient
    Gx=Normalization(Gx)  # Normalizing horizontal gradient image
    (Gy , pedding ) = apply_mask(Image,Prewitt_Y,pedding) #Calculating vertical gradient
    Gy=Normalization(Gy)  # Normalizing vertical gradient image
    Grad_Mag=gradient_magnitude(Gx,Gy) # Calculating gradient magnitude
    Grad_Mag=Normalization(Grad_Mag) # Normalizing gradient magnitude image
    Grad_Ang = gradient_angle(Gx,Gy) # Calculating gradient angle
    return ( Grad_Mag , Grad_Ang )
```

In [10]:

```
def extract_features(training_image_path):
    list_vec=np.array([])
    list_lable=np.array([])
    lable=0
    for dir in glob.glob(training_image_path): # iterating class directory
        for file_name in glob.glob(dir+"\\*"): # iterating image in the diractory
            Img=load_img(file_name) #Loading Image
            (Grad_Mag,Grad_Ang)=Gradient_Operation(Img) #Gradient operation on image
            vec=np.array([])
            h,w = Grad_Mag.shape[:2]
            for i in range(0,h-8,8):
                for j in range(0,w-8,8):
                    for k in range(i,i+16,8):
                        for l in range(j,j+16,8):
                            Bin = np.zeros(9)
                            for m in range (k,k+8):
                                for n in range(l,l+8):
                                    ang=Grad_Ang[m][n]%180 # if angle is not in [0,180) then converting it to [0,180)
                                    bin_index1=int(ang/20)%9 # calculating index1 of bin
                                    bin_index2=(bin_index1+1)%9 # calculating index2 of bin
                                    Bin[bin_index1]=Bin[bin_index1]+Grad_Mag[m][n]*(20-(ang%20))/20 # adding gradient value to bin
                                    Bin[bin_index2]=Bin[bin_index2]+Grad_Mag[m][n]*(ang%20)/20 # adding gradient to bin
                            vec=np.append(vec,Bin) # appending current 9 histogram channels to previous channels
            vec=vec/math.sqrt(sum(p*p for p in vec)) # Normalization
            list_lable=np.append(list_lable,[lable]) # labeling according to the class

            '''a_file = open(file_name[:-3]+"txt", "w")
            for row in vec:
                np.savetxt(a_file, [row],fmt="%f")
            a_file.close()''' #uncomment this to create .txt files of the image feature vector

            if list_vec.size == 0: # creating list of features of each image
                list_vec=np.array([vec])
            else:
                list_vec=np.concatenate((list_vec,[vec]),axis=0)
        lable=lable+1
    list_lable=np.uint8(list_lable)
    return list_vec,list_lable
```

In [11]:

```
list_vec,list_lable=extract_features("Image Data\\Training\\*")
```

In [12]:

```
NN_lables=["1st NN: ","2nd NN: ","3rd NN: "]
is_person=["No Human","Human"]
def detect_Image(testing_image_path,list_vec,list_lable):
    for dir in glob.glob(testing_image_path): # iterating class directory
        for file_name in glob.glob(dir+"\\*"): # iterating image in the diractory
            Img=load_img(file_name) #Loading Image
            (Grad_Mag,Grad_Ang)=Gradient_Operation(Img) #Gradient operation on image
            #data = im.fromarray(np.uint8(Grad_Mag)) # uncomment to the save Images
            #data.save("Gred "+file_name[:-4]+"_Gred.bmp")  # uncomment to the save Images
            input_image_vec=np.array([])
            h,w = Grad_Mag.shape[:2]
            for i in range(0,h-8,8):
                for j in range(0,w-8,8):
                    for k in range(i,i+16,8):
                        for l in range(j,j+16,8):
                            Bin = np.zeros(9)
                            for m in range (k,k+8):
                                for n in range(l,l+8):
                                    ang=Grad_Ang[m][n]%180 # if angle is not in [0,180) then converting it to [0,180)
                                    bin_index1=int(ang/20)%9 # calculating index1 of bin
                                    bin_index2=(bin_index1+1)%9 # calculating index2 of bin
                                    Bin[bin_index1]=Bin[bin_index1]+Grad_Mag[m][n]*(20-(ang%20))/20 # adding gradient value to bin
                                    Bin[bin_index2]=Bin[bin_index2]+Grad_Mag[m][n]*(ang%20)/20 # adding gradient value to bin
                            input_image_vec=np.append(input_image_vec,Bin) # appending current 9 histogram channels to previous ch
            input_image_vec=input_image_vec/math.sqrt(sum(p*p for p in input_image_vec)) # Normalizing

            '''a_file = open(file_name[:-3]+"txt", "w")
            for row in input_image_vec:
                np.savetxt(a_file, [row],fmt="%f")
            a_file.close()''' #uncomment this to create .txt files of the image feature vector

            diff_vec_list=np.array([])
            for training_image_vec in list_vec:
                diff_vec=sum(np.minimum(input_image_vec,training_image_vec))/sum(training_image_vec) # Histogram intersection form
                if list_vec.size == 0:
                    diff_vec_list=np.array([diff_vec])
                else:
                    diff_vec_list=np.concatenate((diff_vec_list,[diff_vec]),axis=0)
            print(file_name+": ")
            r=0
            s=0
            for min_value in np.flip(sorted(diff_vec_list)[-3:]): # Finding 3NNs
                index=list_lable[np.where(diff_vec_list==min_value)][0]
                print(NN_lables[r]+is_person[index]+", Distance: "+str(diff_vec_list[np.where(diff_vec_list==min_value)][0]))
                s=s+index
                r=r+1
            if s>1:
                print("Final class: Human")
            else:
                print("Final class: No Human")
            print("\n")
```

In [13]:

```
detect_Image("Image Data\\Test\\*",list_vec,list_lable)
```

```
Image Data\Test\Test images (Neg)\00000003a_cut.bmp:
1st NN: No Human, Distance: 0.5230812253814514
2nd NN: No Human, Distance: 0.5217387278946161
3rd NN: Human, Distance: 0.5071111274573443
Final class: No Human


Image Data\Test\Test images (Neg)\00000090a_cut.bmp:
1st NN: No Human, Distance: 0.5406897696496992
2nd NN: No Human, Distance: 0.4341612520764696
3rd NN: Human, Distance: 0.4232822065719097
Final class: No Human


Image Data\Test\Test images (Neg)\00000118a_cut.bmp:
1st NN: No Human, Distance: 0.528915698991295
2nd NN: No Human, Distance: 0.5125348409005692
3rd NN: No Human, Distance: 0.51070261228263
Final class: No Human


Image Data\Test\Test images (Neg)\no_person__no_bike_258_Cut.bmp:
1st NN: Human, Distance: 0.43023391372834235
2nd NN: Human, Distance: 0.4237153467931112
3rd NN: Human, Distance: 0.4164266928503956
Final class: Human


Image Data\Test\Test images (Neg)\no_person__no_bike_264_cut.bmp:
1st NN: No Human, Distance: 0.4349281352845946
2nd NN: No Human, Distance: 0.4192476162624286
3rd NN: No Human, Distance: 0.40487694865480367
Final class: No Human


Image Data\Test\Test images (Pos)\crop001034b.bmp:
1st NN: No Human, Distance: 0.6601720964411899
2nd NN: Human, Distance: 0.6441448752593412
3rd NN: Human, Distance: 0.631563307801296
Final class: Human


Image Data\Test\Test images (Pos)\crop001070a.bmp:
1st NN: Human, Distance: 0.5021594008239672
2nd NN: Human, Distance: 0.5008061256348748
3rd NN: Human, Distance: 0.4982364413088014
Final class: Human


Image Data\Test\Test images (Pos)\crop001278a.bmp:
1st NN: Human, Distance: 0.5823198128036237
2nd NN: Human, Distance: 0.4732859645094431
3rd NN: Human, Distance: 0.46199295134623974
Final class: Human


Image Data\Test\Test images (Pos)\crop001500b.bmp:
1st NN: No Human, Distance: 0.49283979086006785
2nd NN: Human, Distance: 0.45016034444018227
3rd NN: Human, Distance: 0.420667177389211
Final class: Human


Image Data\Test\Test images (Pos)\person_and_bike_151a.bmp:
1st NN: Human, Distance: 0.544543240800552
2nd NN: Human, Distance: 0.525673758374802
3rd NN: Human, Distance: 0.5253446888251687
Final class: Human
```