Project 2: Human Detection using HOG Feature Report

Cat Mai cm6226

05/20/2023

Requirements

1. Being able to compile Python code
2. Install Pillow and numpy, on some command line
    a. pip install Pillow
    b. pip install numpy
3. Change this section in main.py to appropriate parameters:

```python
# CHANGE THESE ######
train_positive_dir = "train_positive"
train_negative_dir = "train_negative"
test_img = "test_positive/T5.bmp"
distance = "hellinger"
# distance = "intersection"
#####################
```

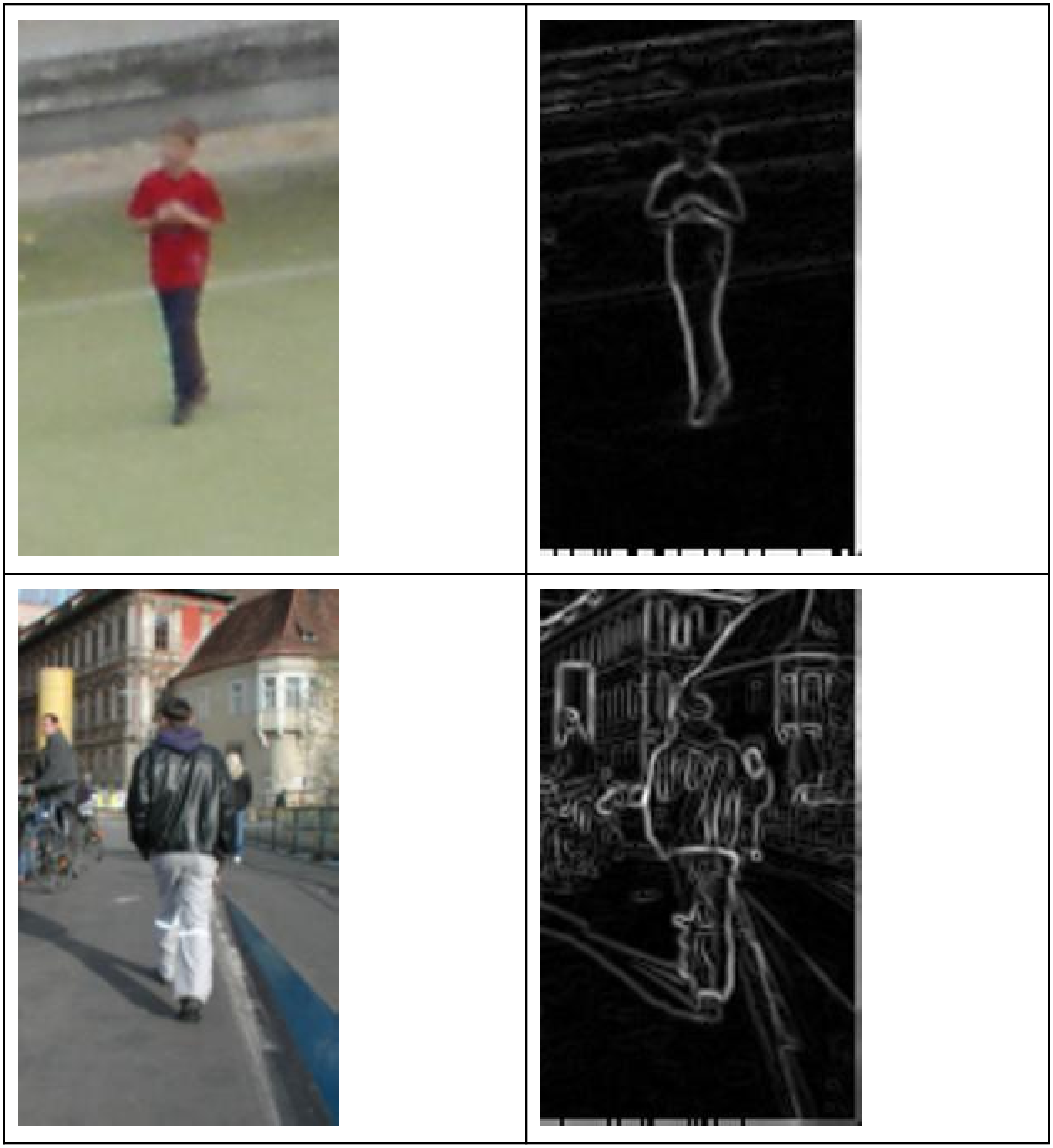
train_positive_dir is the directory name of positive training inputs. Similarly for train_negative_dir

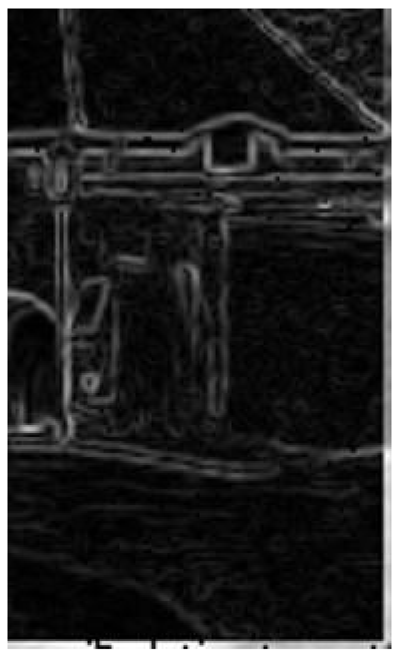
test_img is full path to test input file

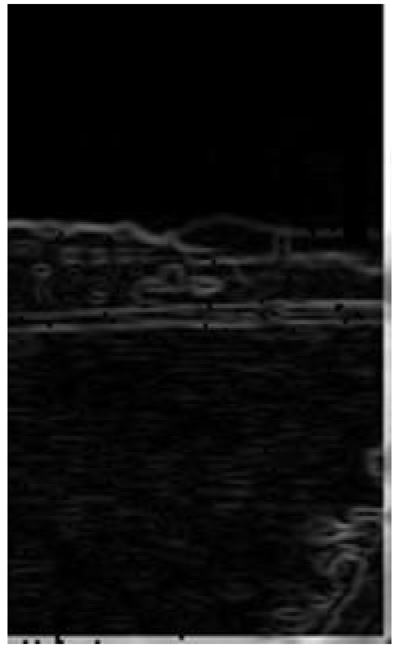
distance is either “hellinger” or “intersection”

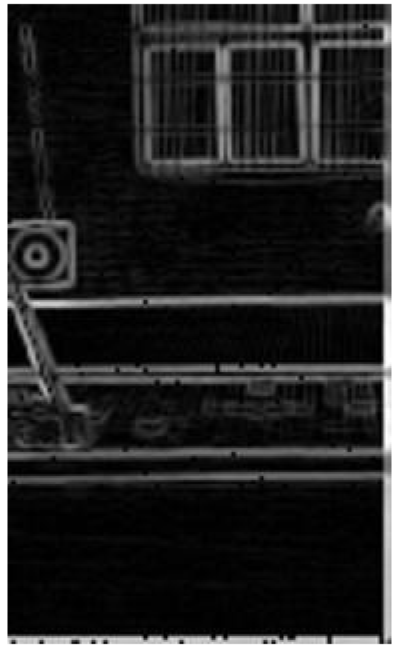
Normalized Gradient Magnitude images



Source Code

```
import PIL
from PIL import Image
import math
import numpy as np
import os


def to_grayscale(img):

	# Grayscale formula = Round(0.299R + 0.587G + 0.114B)
	img = np.asarray(Image.open(img))
	gs_img = np.round(np.dot(img[...,:3], [0.299, 0.587, 0.114]))

	return gs_img


def get_gradients(img):

	height, width = img.shape
	grad_x = np.zeros(img.shape)
	grad_y = np.zeros(img.shape)
	angles = np.zeros(img.shape)
	magnitude = np.zeros(img.shape)

	# Sobel operators
	Gx = [
		[-1, 0, 1],
		[-2, 0, 2],
		[-1, 0, 1]
	]

	Gy = [
		[1, 2, 1],
		[0, 0, 0],
		[-1, -2, -1]
	]

	sobel_height, sobel_width = 3, 3

	for row in range(height):
		for col in range(width):
```

```
                        # calculate gradients
                        for i in range(sobel_height):
                                for j in range(sobel_width):
                                        try:
                                                grad_x[row][col] += Gx[i][j] * img[row+i][col+j]
                                                grad_y[row][col] += Gy[i][j] * img[row+i][col+j]
                                        except IndexError:
                                                # mask goes out of frame, do nothing
                                                pass


        # calculate gradient angles and gradient magnitude
        for row in range(height):
                for col in range(width):

                        if grad_x[row][col] != 0:
                                angles[row][col] =
math.degrees(math.atan(grad_y[row][col]/grad_x[row][col]))
                                magnitude[row][col] = math.sqrt(grad_x[row][col]**2 +
grad_y[row][col]**2)

                        if 180 <= angles[row][col] < 360:
                                angles[row][col] = angles[row][col] - 180


        # normalization to range [0, 255]
        magnitude = magnitude / (np.max(magnitude)/255)

        return angles, magnitude


def get_hog_features(gradient_angles, gradient_magnitude):

        # 20 X 12 cells, 9 bins
        block_3d = np.zeros((20,12,9))
        normalized_hog_feature = np.zeros(7524)
        block = np.zeros(36)

   # distance between bin centers
        step = 20

   # Calculating the feature vector (block_3d)
        for row in range(19):
```

```
        for col in range(11):

            histogram = np.zeros(9)

            for row_block in range(8*row, 8*(row+1)):
                for col_block in range(8*col, 8*(col+1)):

                    angle = gradient_angles[row_block][col_block]
                    angle = angle if angle <= 180 else angle - 180
                    mag = gradient_magnitude[row_block][col_block]

                    # Determine weight for two bins based on distance from
bin center
                    bin1 = math.floor(angle/step)
                    bin2 = (bin1+1) % 9
                    center = step*bin1 + 10
                    histogram[bin1] += (1 - (angle-center)/step) * mag
                    histogram[bin2] += (angle-center)/step * mag

            for bin in range(9):
                block_3d[row][col][bin] = histogram[bin]


    # flatten the block vector
    index = 0
    for row in range(19):
        for col in range(11):
            sum_squared = 0

            # For each cell (4 per block)
            for cell in range(4):
                for i in range(9):
                    block[9*cell + i] = block_3d[row][col][i]
                    sum_squared += block[9*cell + i]**2

            # block normalization
            if sum_squared != 0:
                block_length = math.sqrt(sum_squared)
                block = block/block_length

            for i in range(index, index+36):
                normalized_hog_feature[i] = block[i%36]

            index += 36
```

```
        # normalize vector by sum of components
        return normalized_hog_feature/np.sum(normalized_hog_feature)


def get_similarity(training_vectors, input_vector, distance='intersection'):
        """
                Calculate similarity between test input and each training input.
                Distance options: 'intersection', 'hellinger'
                Returns a vector of size = number of training inputs
        """

        n_training, _ = training_vectors.shape
        sim_vector = np.zeros(n_training)

        for i in range(n_training):
                if distance == 'intersection':
                        sim_vector[i] = 1 - np.sum(np.minimum(input_vector, training_vectors[i]))

                if distance == 'hellinger':
                        product = np.abs(np.multiply(input_vector, training_vectors[i]))
                        sim_vector[i] = 1 - np.sum(np.sqrt(product))


        return sim_vector


def classifyThreeNN(sim_vector, label_vector):

        # np.argsort returns indices of sorted values (sorted by ascending)
        # get first three i.e. closest three
        indices_of_closest_neighbors = np.argsort(sim_vector)[:3]

        score = sum([label_vector[i] for i in indices_of_closest_neighbors])
        if score < 2:
                return 'No-human'
        return 'Human'



if __name__ == "__main__":

    # CHANGE THESE ######
        train_positive_dir = "train_positive"
```

```
train_negative_dir = "train_negative"
test_img = "test_negative/T10.bmp"
distance = "hellinger"
# distance = "intersection"
#####################


# File processing
train_positive = [f"{train_positive_dir}/{f}" for f in os.listdir(train_positive_dir)]
train_negative = [f"{train_negative_dir}/{f}" for f in os.listdir(train_negative_dir)]
train_files = train_positive + train_negative
train_labels = [1 for i in train_positive] + [0 for i in train_negative]

n_training = len(train_files)
training_vectors = np.zeros((n_training,7524))

# Training
for i in range(n_training):
        img = to_grayscale(train_files[i])
        angles, magnitude = get_gradients(img)
        training_vectors[i] = get_hog_features(angles,magnitude)

# For input image
img = to_grayscale(test_img)
angles, magnitude = get_gradients(img)
test_vector = get_hog_features(angles, magnitude)
sim_vector = get_similarity(training_vectors, test_vector, distance=distance)
classification = classifyThreeNN(sim_vector, train_labels)
print(f"Test img class: {classification}")
```

Tables

Using **histogram intersection** as distance

Test input 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**
Image 1	Human	DB4.bmp 0.4915 Human	DB9.bmp 0.5025 Human	DB2.bmp 0.5029 Human	Human
Image 2	Human	DB2.bmp 0.3973 Human	DB8.bmp 0.4028 Human	DB9.bmp 0.4264 Human	Human
Image 3	Human	DB19.bmp 0.4244 No-human	DB4.bmp 0.4494 Human	DB9.bmp 0.4585 Human	Human
Image 4	Human	DB4.bmp 0.4582 Human	DB2.bmp 0.4834 Human	DB17.bmp 0.4958 No-human	Human
Image 5	Human	DB9.bmp 0.3975 Human	DB8.bmp 0.4021 Human	DB17.bmp 0.4028 No-human	Human
Image 6	No-human	DB4.bmp 0.4228 Human	DB9.bmp 0.4347 Human	DB18.bmp 0.4389 No-human	Human
Image 7	No-human	DB16.bmp 0.4792 No-human	DB11.bmp 0.5218 No-human	DB15.bmp 0.5507 No-human	No-human
Image 8	No-human	DB17.bmp 0.4337 No-human	DB18.bmp 0.4364 No-human	DB2.bmp 0.4458 Human	No-human
Image 9	No-human	DB15.bmp 0.4405 No-human	DB16.bmp 0.4633 No-human	DB9.bmp 0.4843 Human	No-human
Image 10	No-human	DB18.bmp 0.4417 No-human	DB8.bmp 0.4511 Human	DB9.bmp 0.4529 Human	Human

Using **Hellinger distance** as distance

Test input 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
Image 1	Human	DB4.bmp 0.1878 Human	DB17.bmp 0.1996 No-human	DB19.bmp 0.2027 No-human	No-human
Image 2	Human	DB2.bmp 0.1436 Human	DB8.bmp 0.1462 Human	DB4.bmp 0.1591 Human	Human
Image 3	Human	DB19.bmp 0.1564 No-human	DB4.bmp 0.1677 Human	DB17.bmp 0.1741 No-human	No-human
Image 4	Human	DB4.bmp 0.1666 Human	DB2.bmp 0.1899 Human	DB17.bmp 0.1947 No-human	Human
Image 5	Human	DB9.bmp 0.1365 Human	DB17.bmp 0.1407 No-human	DB8.bmp 0.1411 Human	Human
Image 6	No-human	DB4.bmp 0.1539 Human	DB17.bmp 0.1632 No-human	DB9.bmp 0.1702 Human	Human
Image 7	No-human	DB11.bmp 0.2340 No-human	DB16.bmp 0.2530 No-human	DB15.bmp 0.2779 No-human	No-human
Image 8	No-human	DB17.bmp 0.1466 No-human	DB18.bmp 0.1519 No-human	DB4.bmp 0.1532 Human	No-human
Image 9	No-human	DB15.bmp 0.1520 No-human	DB9.bmp 0.1939 Human	DB18.bmp 0.1977 No-human	No-human
Image 10	No-human	DB18.bmp 0.1678 No-human	DB9.bmp 0.1702 Human	DB8.bmp 0.1725 Human	Human