Computer Vision Project 2: Human Detection

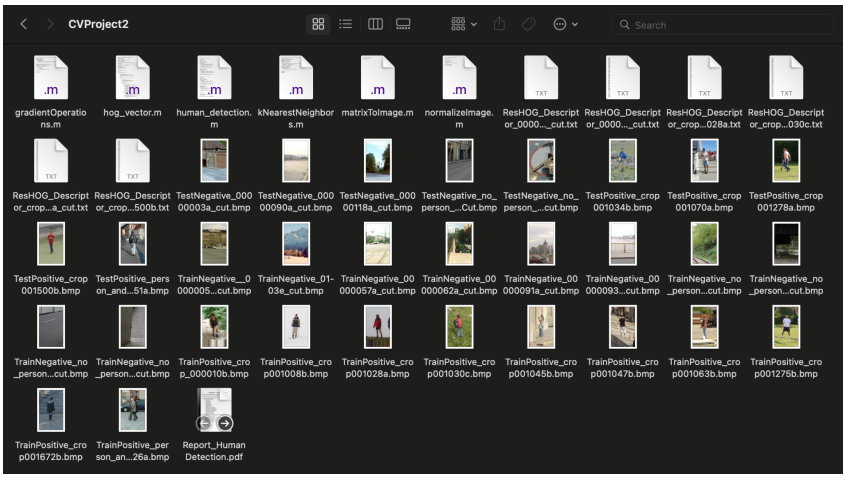
Nishith Sharma ns5014

The project is compiled and coded in Matlab. All the test result are added and enclosed with the files in the submission. The submission includes the code files for running the project in Matlab.

The following are the instructions to run the code in matlab:

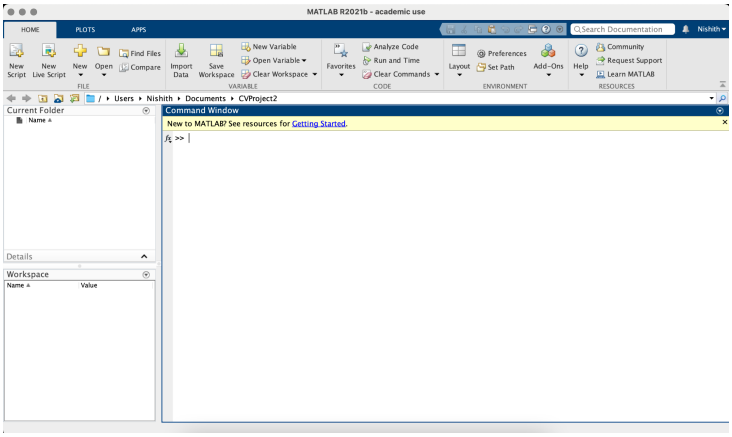
Implementation Language: Matlab

1. Download all the files from the submission from brightspace
2. **Place** all the files in a **single folder**.
    - This is what the folder can look like:

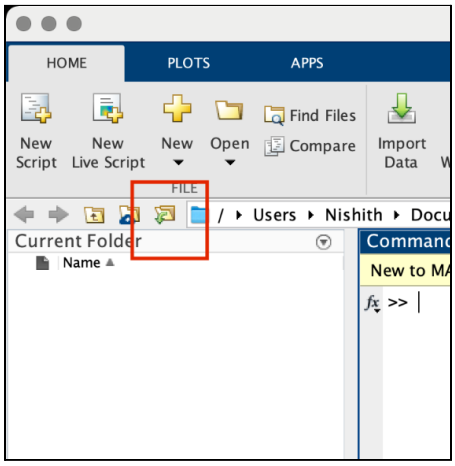


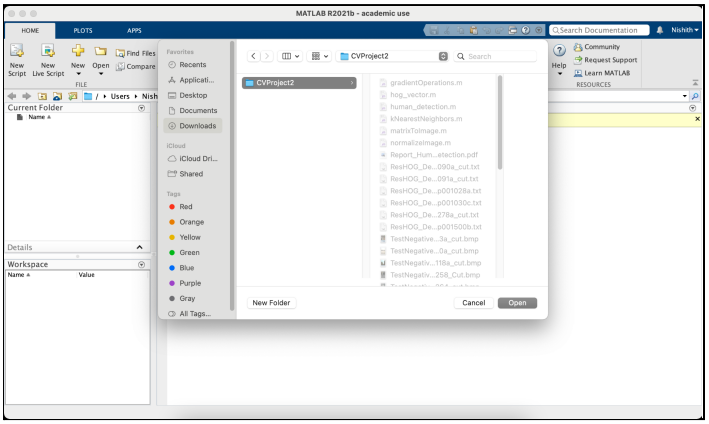
3. The names of the files are as follows:
    - human_detection.m
    - hog_vector.m
    - kNearestNeighbors.m
    - gradientOperations.m
    - matrixToImage.m
    - normalizeImage.m
    - Report_Human Detection.pdf
    - ResHOG_Descriptor_crop001028a.txt
    - ResHOG_Descriptor_crop001030c.txt
    - ResHOG_Descriptor_00000091a_cut.txt
    - ResHOG_Descriptor_crop001278a_cut.txt
    - ResHOG_Descriptor_crop001500b.txt
    - ResHOG_Descriptor_00000090a_cut.txt
    - TestNegative*.bmp : 5 images
    - TestPositive*.bmp : 5 images
    - TrainNegative*.bmp : 10 images
    - TrainPositive*.bmp : 10 images

4. **Don't Change the name** of any file, as it is being auto read by the code, even the input image, there is a regex match to read the images and given them labels
5. Open Matlab:

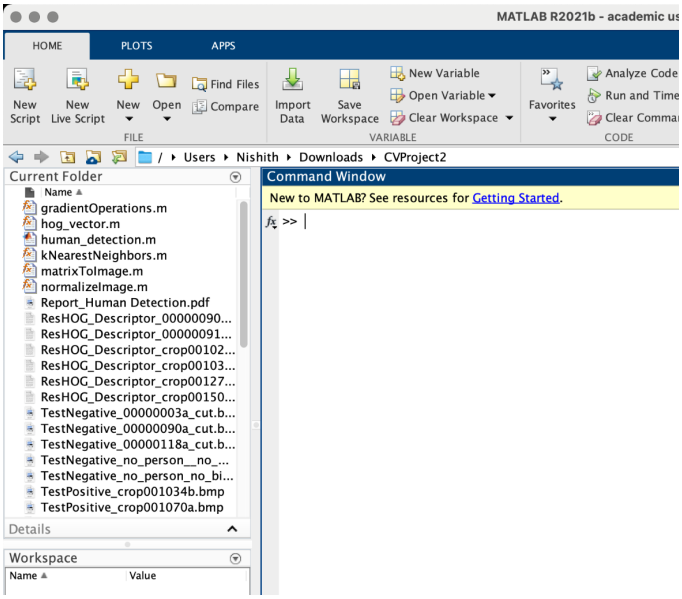


6. Browse to the folder where everything is saved and open it in matlab, click on the button:



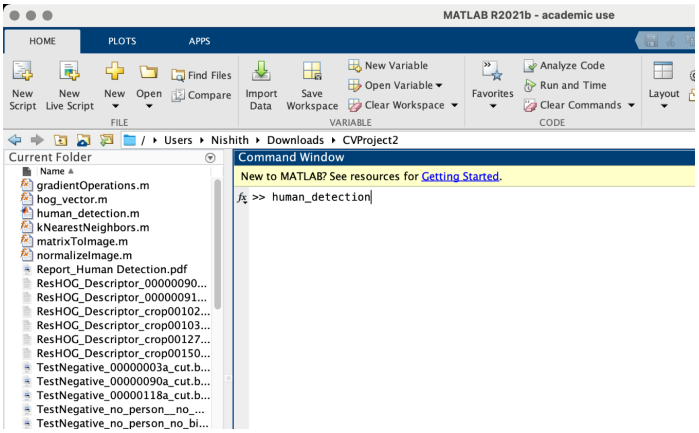


7. After opening it, it will look as follows:

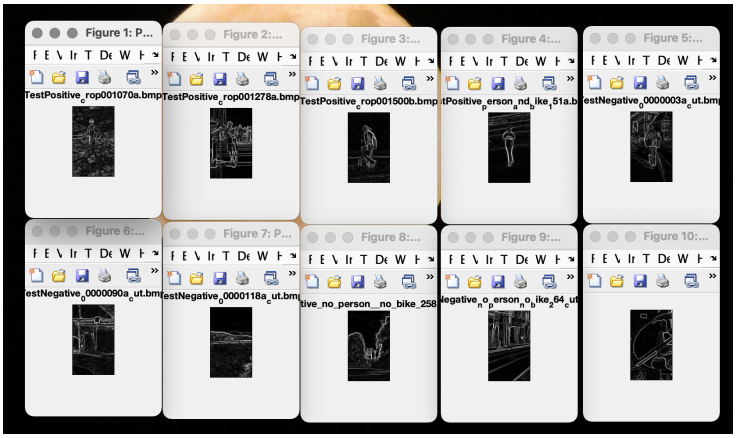


8. Once open, go to the command window, and write the command:

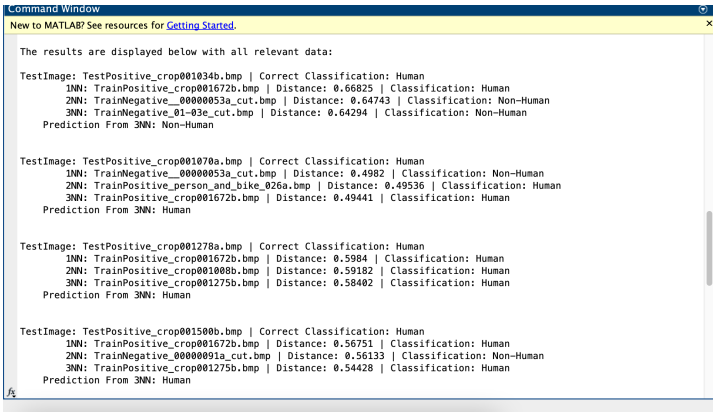
human_detection



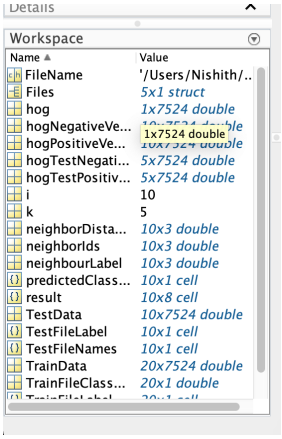
9. You will be able to see the following output elements:
    - Gradient images for Test Images



- Data for prediction for all the 10 images with all the 3 nearest neighbors and their distances.



- All the relevant variables will be visible in the workspace.

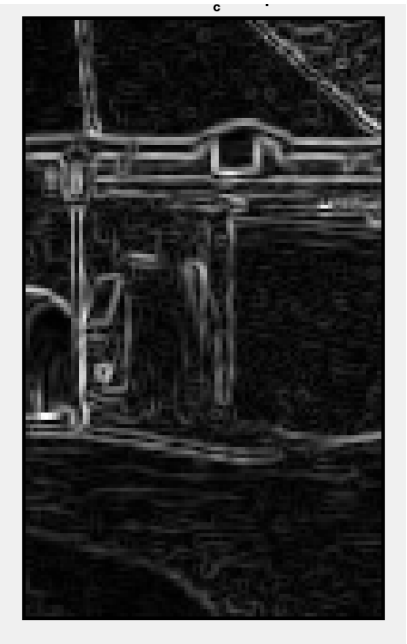


Implementation Methodology Summarized

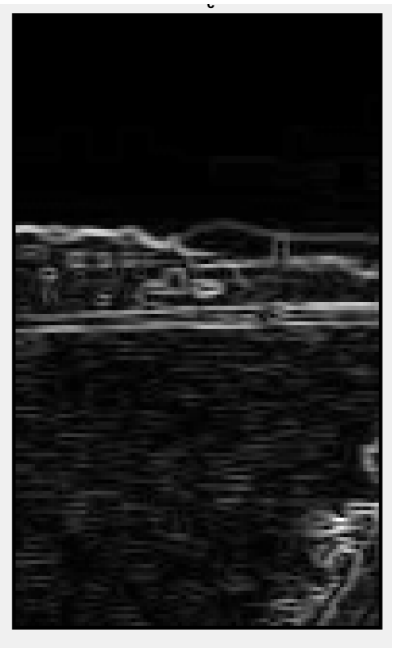
The following methodology was used and each step can be seen as a commented section in the matlab code files, I'll mention which file it is present in as well.

1. Read Image and convert to grayscale using the formula: *//hog_vector.m*
    - I = Round(0.299R + 0.587G + 0.114B) where *R, G* and *B* are
2. Gradient operation are performed on it: *//gradientOperations.m*
    - Prewitt operator for calculating magnitude using squared sum. Magnitude is normalized to [0-255] and rounded off as well.
    - For angle, the range is adjusted between 0-180. If greater than 180, subtract 180 from it.
    - If undefined, then give 0. If G_x and G_y are 0, then Magnitude and Angle is 0
    - Display the gradient magnitude images of Test Images(10 output).
3. HOG feature building: *//hog_vector.m*
    - 9 bins are used for unsigned histogram
    - Cell size is taken 8x8
    - Block size = 16 x 16 pixels (or 2 x 2 cells)
    - *block overlap* or *step size* = 8 pixels (or 1 cell.)
    - Total overlapping blocks: 19 X 11 blocks
    - Total size of descriptor: 19x11x4x9 = 7524
    - The L2 Norm is used for block normalization.
    - Final histogram and feature in floating point numbers.
4. Compute the HOG feature for Training images, and Test images. *//human_detection.m*
5. Send the features of Training with labels and Test images to the 3NN classifier.
6. 3NN Classifier, Histogram Intersection: *//kNearestNeighbour.m*
    - The histogram intersection formula is used for calculating the similarity between Test image feature, with all the Training images feature.
    - Maximum similarity top 3 are selected.
    - Return the Top 3 for each Test image, their distance, and labels
7. Display the outputs. *//Plots and command window*

Gradient Images for Test Negative Non Human



00000003a_cut.bmp



00000090a_cut.bmp

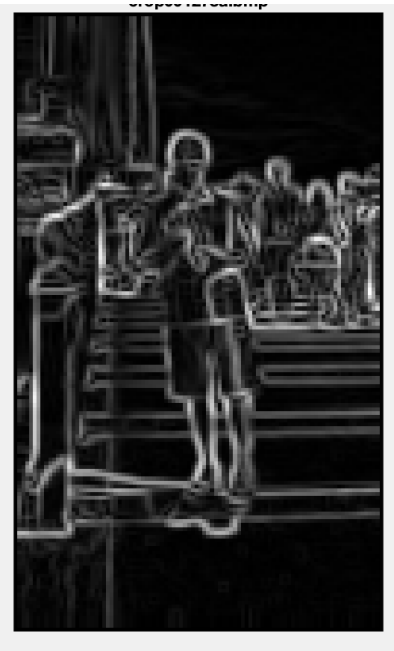
00000118a_cut.bmp

no_person__no_bike_258_Cut.bmp

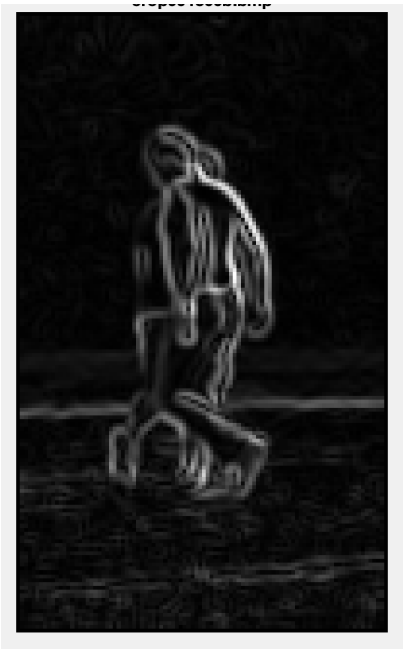
no_person__no_bike_264_cut.bmp

Gradient Images for Test Positive Human

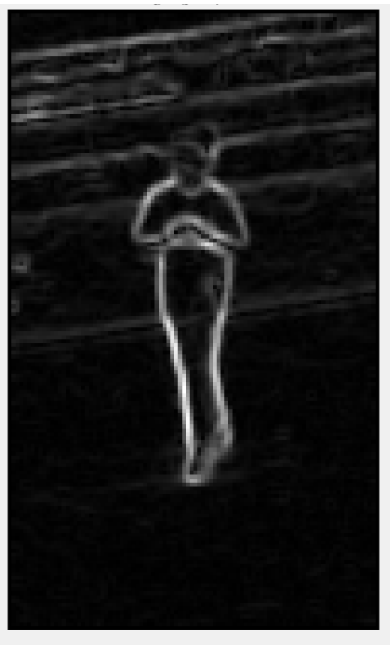
crop001034b.bmp



crop001070a.bmp



crop001278a.bmp



crop001500b.bmp

person_and_bike_151a.bmp

Table

Test Image	Correct Class	1st NN			2nd NN			3rd NN			Classfication from 3NN
		File Name	Distance	Class	File Name	Distance	Class	File Name	Distance	Class	
crop001034b	Human	crop001672b	0.66825	Human	00000053a_cut	0.64743	Non-Human	01-03e_cut	0.64294	Non-Human	Non-Human
crop001070a	Human	00000053a_cut	0.4982	Non-Human	person_and_bike_026a	0.49536	Human	crop001672b	0.49441	Human	Human
crop001278a	Human	crop001672b	0.5984	Human	crop001008b	0.59182	Human	crop001275b	0.58402	Human	Human
crop001500b	Human	crop001672b	0.56751	Human	00000091a_cut	0.56133	Non-Human	crop001275b	0.54428	Human	Human
person_and_bike_151a	Human	crop001030c	0.50466	Human	person_and_bike_026a	0.502	Human	crop001275b	0.49403	Human	Human
00000003a_cut	Non-Human	00000053a_cut	0.5766	Non-Human	crop001672b	0.5739	Human	00000093a_cut	0.54823	Non-Human	Non-Human
00000090a_cut	Non-Human	00000093a_cut	0.47897	Non-Human	00000057a_cut	0.47176	Non-Human	crop001672b	0.44589	Human	Non-Human
00000118a_cut	Non-Human	00000093a_cut	0.56144	Non-Human	00000053a_cut	0.55531	Non-Human	00000091a_cut	0.54894	Non-Human	Non-Human
no_person__no_bike_258_Cut	Non-Human	00000057a_cut	0.49611	Non-Human	crop001672b	0.48911	Human	crop001275b	0.48456	Human	Human
no_person_no_bike_264_cut	Non-Human	00000053a_cut	0.4415	Non-Human	01-03e_cut	0.44046	Non-Human	crop001672b	0.43909	Human	Non-Human

Test Image Results:

Images	Correctly Classified	Wrongly Classifier	Error Rate
5 Human Test Images	4	1	20%
5 Non-Human Test Images	4	1	20%
10 Test Images	8	2	20%

Source Code

human_detection.m

```
clc; clear;

%% Creating HOG Vectors for Training Images
TrainFileNames = [];
TrainFileLabel = [];
TrainFileClassLabel = [];

% Positive Training Images
% Reading files from directory
Files=dir('TrainPositive*.bmp');
hogPositiveVector = [];
for k=1:length(Files)
   TrainFileNames = [TrainFileNames ; {Files(k).name}];
   TrainFileLabel = [TrainFileLabel ; {'Human'}];
   % Positive images have 1 Class
   TrainFileClassLabel = [TrainFileClassLabel ; 1 ];
   FileName = strcat(Files(k).folder,'/',Files(k).name);

   % Generating HOG Vector
   hog = hog_vector(FileName,false,Files(k).name);
   hogPositiveVector = [hogPositiveVector; hog];
end

% Negative Training Images
% Reading files from directory
Files=dir('TrainNegative*.bmp');
hogNegativeVector = [];
for k=1:length(Files)
   TrainFileNames = [TrainFileNames ; {Files(k).name}];
   TrainFileLabel = [TrainFileLabel ; {'Non-Human'}];
   % Negative images have 0 Class
   TrainFileClassLabel = [TrainFileClassLabel ; 0 ];
   FileName = strcat(Files(k).folder,'/',Files(k).name);

   % Generating HOG Vector
   hog = hog_vector(FileName,false,Files(k).name);
   hogNegativeVector = [hogNegativeVector; hog];
end

%% Creating HOG Vectors for Test Images
TestFileNames=[];
TestFileLabel=[];

% Positive Test Images
% Reading files from directory
Files=dir('TestPositive*.bmp');
hogTestPositiveVector = [];
```

```
for k=1:length(Files)
   TestFileNames = [TestFileNames ; {Files(k).name}];
   TestFileLabel = [TestFileLabel ; {'Human'}];
   FileName = strcat(Files(k).folder,'/',Files(k).name);

   % Generating HOG Vector
   hog = hog_vector(FileName,true,Files(k).name);
   hogTestPositiveVector = [hogTestPositiveVector; hog];
end

% Negative Test Images
% Reading files from directory
Files=dir('TestNegative*.bmp');
hogTestNegativeVector = [];
for k=1:length(Files)
   TestFileNames = [TestFileNames ; {Files(k).name}];
   TestFileLabel = [TestFileLabel ; {'Non-Human'}];
   FileName = strcat(Files(k).folder,'/',Files(k).name);

   % Generating HOG Vector
   hog = hog_vector(FileName,true,Files(k).name);
   hogTestNegativeVector = [hogTestNegativeVector; hog];
end

%% Collating data
% Training Images data
% First 10 are 1(HumanClass). Last 10 are 0(nonHumanClass)
TrainData = [hogPositiveVector; hogNegativeVector];

% Test Images data
% First 5 are 1(HumanClass). Last 5 are 0(nonHumanClass)
TestData = [hogTestPositiveVector ; hogTestNegativeVector];


%% Running 3-KNN on the test images using train images
[predictedClassLabels neighbourLabel neighborIds neighborDistances] =
kNearestNeighbors(TrainData,TrainFileClassLabel, TestData, 3)


% Collating the result using neighbour indexes returned from KNN
result = [TestFileNames TestFileLabel TrainFileNames(neighborIds(:,1))
TrainFileLabel(neighborIds(:,1)) TrainFileNames(neighborIds(:,2))
TrainFileLabel(neighborIds(:,2)) TrainFileNames(neighborIds(:,3))
TrainFileLabel(neighborIds(:,3))];


%% Print results:
fprintf("The results are displayed below with all relevant data:\n\n")
for i = 1:length(result)
   fprintf("TestImage: "+ TestFileNames(i)+" | Correct Classification:
"+TestFileLabel(i)+"\n");
   fprintf("\t1NN: "+ TrainFileNames(neighborIds(i,1))+" | Distance:
"+neighborDistances(i,1)+" | Classification:
"+TrainFileLabel(neighborIds(i,1))+"\n");
```

```
   fprintf("\t2NN: "+ TrainFileNames(neighborIds(i,2))+" | Distance:
"+neighborDistances(i,2)+" | Classification:
"+TrainFileLabel(neighborIds(i,2))+"\n");
   fprintf("\t3NN: "+ TrainFileNames(neighborIds(i,3))+" | Distance:
"+neighborDistances(i,3)+" | Classification:
"+TrainFileLabel(neighborIds(i,3))+"\n");
   fprintf("    Prediction From 3NN: "+ predictedClassLabels(i)+"\n\n\n");
end
```

---

hog_vector.m

```
function [feature] = hog_vector(Img, displayImage,imageName)

%% Image read operations
rgbImage = imread(Img);
rgbImage = double(rgbImage);

% Use custom channel proportion to get images
redChannel = rgbImage(:, :, 1);
greenChannel = rgbImage(:, :, 2);
blueChannel = rgbImage(:, :, 3);

% Round off the values using the formula to generate matrix
im = round(0.299*redChannel + 0.587*greenChannel + 0.114*blueChannel);
rows=size(im,1);
cols=size(im,2);

%% Gradient Operations to Calculate Magnitude, Gx, Gy and GradientAngle
[magnitude, angle] = gradientOperations(im);
if(displayImage==1)

% Display Image
figure('Name','Prewitt Magnitude');
imshow(matrixToImage(magnitude));
title(imageName)
end

%% Generate the feature
% Initialized the feature vector
feature=[];

% Iterations for Blocks(19x11 blocks)
for i = 0: rows/8 - 2
   for j= 0: cols/8 -2
```

```
% Each block is of 16x16
% Getting Block magnitude and angles with overlap
mag_block = magnitude(8*i+1 : 8*i+16 , 8*j+1 : 8*j+16);
ang_block = angle(8*i+1 : 8*i+16 , 8*j+1 : 8*j+16);
block_feature=[];

% Iterations for cells in a block(2x2 cells in block)
for x= 0:1
    for y= 0:1

        % Each Cell is 8x8
        % Getting Cell magnitude and angles
        angle_cell =ang_block(8*x+1:8*x+8, 8*y+1:8*y+8);
        mag_cell =mag_block(8*x+1:8*x+8, 8*y+1:8*y+8);
        histr = zeros(1,9);

        %Iterations for pixels in one cell
        for p=1:8
            for q=1:8
                alpha= angle_cell(p,q);

                % Division of Magnitude in Buckets proportionally
                if alpha>10 && alpha<=30
                    histr(1)=histr(1)+ mag_cell(p,q)*(30-alpha)/20;
                    histr(2)=histr(2)+ mag_cell(p,q)*(alpha-10)/20;
                elseif alpha>30 && alpha<=50
                    histr(2)=histr(2)+ mag_cell(p,q)*(50-alpha)/20;
                    histr(3)=histr(3)+ mag_cell(p,q)*(alpha-30)/20;
                elseif alpha>50 && alpha<=70
                    histr(3)=histr(3)+ mag_cell(p,q)*(70-alpha)/20;
                    histr(4)=histr(4)+ mag_cell(p,q)*(alpha-50)/20;
                elseif alpha>70 && alpha<=90
                    histr(4)=histr(4)+ mag_cell(p,q)*(90-alpha)/20;
                    histr(5)=histr(5)+ mag_cell(p,q)*(alpha-70)/20;
                elseif alpha>90 && alpha<=110
                    histr(5)=histr(5)+ mag_cell(p,q)*(110-alpha)/20;
                    histr(6)=histr(6)+ mag_cell(p,q)*(alpha-90)/20;
                elseif alpha>110 && alpha<=130
                    histr(6)=histr(6)+ mag_cell(p,q)*(130-alpha)/20;
                    histr(7)=histr(7)+ mag_cell(p,q)*(alpha-110)/20;
                elseif alpha>130 && alpha<=150
                    histr(7)=histr(7)+ mag_cell(p,q)*(150-alpha)/20;
                    histr(8)=histr(8)+ mag_cell(p,q)*(alpha-130)/20;
                elseif alpha>150 && alpha<=170
                    histr(8)=histr(8)+ mag_cell(p,q)*(170-alpha)/20;
                    histr(9)=histr(9)+ mag_cell(p,q)*(alpha-150)/20;
                elseif alpha>=0 && alpha<=10
                    histr(1)=histr(1)+ mag_cell(p,q)*(alpha+10)/20;
                    histr(9)=histr(9)+ mag_cell(p,q)*(10-alpha)/20;
                elseif alpha>170 && alpha<=180
                    histr(9)=histr(9)+ mag_cell(p,q)*(190-alpha)/20;
                    histr(1)=histr(1)+ mag_cell(p,q)*(alpha-170)/20;
                end
```

```
                    end
                end

                % Concatenation of Four histograms to form one block feature
                block_feature=[block_feature histr];
            end
        end

        % Block Normalization using L2 Norm
        block_feature = block_feature/sqrt(sum(block_feature.^2));

        %Features concatenation
        feature=[feature block_feature];

    end
end

% Removing Infinitiy values
feature(isnan(feature))=0;
feature;
```

---

gradientOperations.m

```
function [prewittmagnitude, prewittangle] = gradientOperations(inputImageMatrix)

%% Store input size and initialize matrix
image = inputImageMatrix;
sizeImage = size(image);
gradientx = zeros(sizeImage);
gradienty = zeros(sizeImage);

%% Calculate the X and Y gradient using the formula
for i=2:sizeImage(1)-1
    for j=2:sizeImage(2)-1
        gradientx(i,j) =image(i-1,j+1)-image(i-1,j-1) +image(i,j+1)-image(i,j-1)
+image(i+1,j+1)-image(i+1,j-1);

        gradienty(i,j) =image(i-1,j-1)-image(i+1,j-1) +image(i-1,j)-image(i+1,j)
+image(i-1,j+1)-image(i+1,j+1);
    end
end

%% Calculate magnitude using squared sum method
prewittmagnitude = sqrt(gradientx.^2+ gradienty.^2);

% Normalize the Components after the gradient operation to display the
% Image correctly in the range of 0-255
prewittmagnitude = round(normalizeImage(prewittmagnitude));

%% Calculate the angle using tan-1
```

```
prewittangle =atan2d(gradienty,gradientx);
prewittangle(isnan(prewittangle))=0;

% If the angle are less than 0, then increment 180
% Range of angle should between 0-180
for i = 1:sizeImage(1)
   for j = 1:sizeImage(2)
       if(prewittangle(i,j)<0)
           prewittangle(i,j)=prewittangle(i,j)+180;
       end
   end
end
prewittmagnitude;
prewittangle;
```

---

kNearestNeighbors.m

```
function [predictedClassLabel neighbourLabel neighborIds neighborDistances] =
kNearestNeighbors(dataMatrix, dataLabels,queryMatrix, k)

%% Initialize all the matrix to store different data to be returned
neighborIds = zeros(size(queryMatrix,1),k);
neighbourLabel = zeros(size(queryMatrix,1),k);
predictedLabel = zeros(size(queryMatrix,1),1);
predictedClassLabel=[];
neighborDistances = neighborIds;
numDataVectors = size(dataMatrix,1);
numQueryVectors = size(queryMatrix,1);

%% Similarity computation for distance
for i=1:numQueryVectors

   % Histogram intersection sum(min(test,train))/sum(train)
   similarity =
sum(min(repmat(queryMatrix(i,:),numDataVectors,1),dataMatrix)')./sum(dataMatrix');

   % Sort in decreasing order of similarity
   [sortval sortpos] = sort(similarity,'descend');

   % Get relevant data point based on sorting order
   neighborIds(i,:) = sortpos(1:k);
   neighbourLabel(i,:) = dataLabels(neighborIds(i,:));
   neighborDistances(i,:) = sortval(1:k);
end
```

```
%% Prediction is where more labels of same sign
predictedLabel = sum(neighbourLabel')';
predictedLabel(predictedLabel<2)=0;
predictedLabel(predictedLabel>=2)=1;

% From 0-1 Lables, build back Human and Non-Human classification
for i = 1:length(predictedLabel)
   if(predictedLabel(i)==1)
       predictedClassLabel=[predictedClassLabel; {"Human"}];
   else
       predictedClassLabel=[predictedClassLabel; {"Non-Human"}];
   end
end

%% Return the results
predictedClassLabel;
neighbourLabel;
neighborDistances;
neighborIds;
```

---

normalizeImage.m

```
function normalizedMatrix = normalizeImage(imageMatrix)

% Check what is the lowest value
lo = min(min(imageMatrix));

% Adjust low to 0
if(lo<0)
   imageMatrix=imageMatrix+abs(lo);
end

% Check the highest and the normalize all values by
% scalling factor to get in 0-255 range
hi=max(max(imageMatrix));
if(hi>255)
   normalizeComponent=255/hi;
   imageMatrix = imageMatrix.*normalizeComponent;
end

%% Return Values
new_lo = min(min(imageMatrix));
new_hi = max(max(imageMatrix));
normalizedMatrix = imageMatrix;
```

---

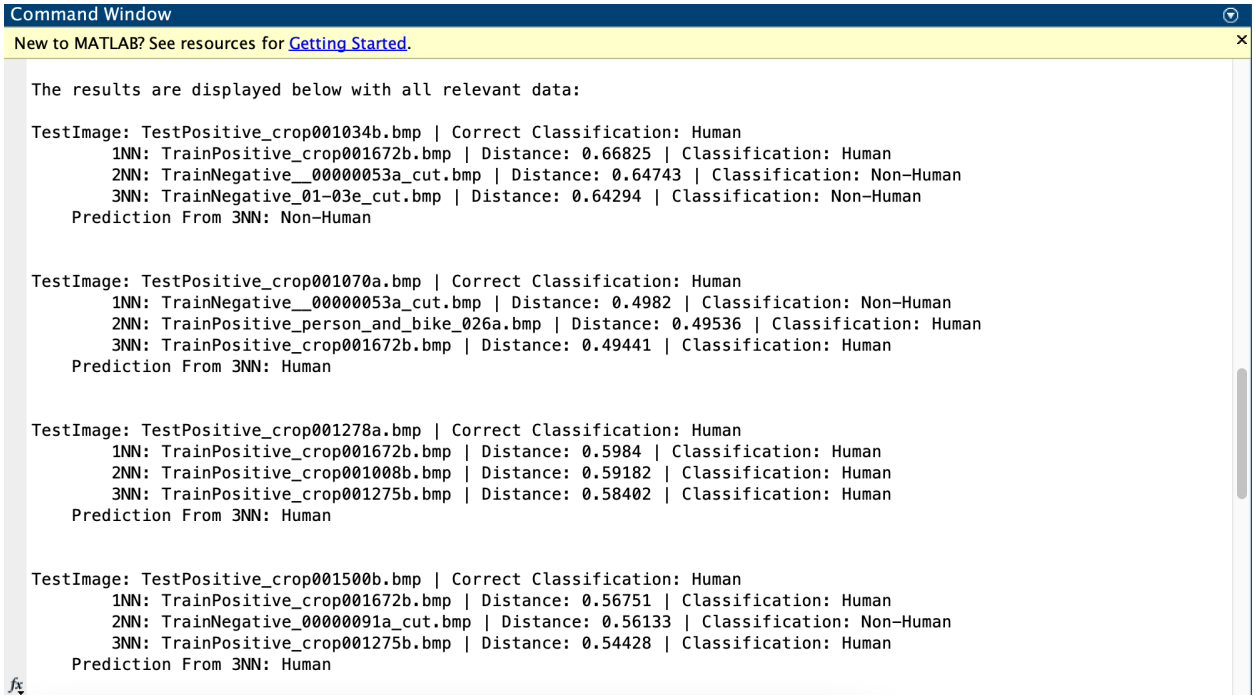
matrixToImage.m

```
function image = matrixToImage(matrix)

%% Display Image
image = mat2gray(matrix);
```

Output:

Data for 3 Nearest neighbour classification



```
TestImage: TestPositive_person_and_bike_151a.bmp | Correct Classification: Human
        1NN: TrainPositive_crop001030c.bmp | Distance: 0.50466 | Classification: Human
        2NN: TrainPositive_person_and_bike_026a.bmp | Distance: 0.502 | Classification: Human
        3NN: TrainPositive_crop001275b.bmp | Distance: 0.49403 | Classification: Human
    Prediction From 3NN: Human


TestImage: TestNegative_00000003a_cut.bmp | Correct Classification: Non-Human
        1NN: TrainNegative__00000053a_cut.bmp | Distance: 0.5766 | Classification: Non-Human
        2NN: TrainPositive_crop001672b.bmp | Distance: 0.5739 | Classification: Human
        3NN: TrainNegative_00000093a_cut.bmp | Distance: 0.54823 | Classification: Non-Human
    Prediction From 3NN: Non-Human


TestImage: TestNegative_00000090a_cut.bmp | Correct Classification: Non-Human
        1NN: TrainNegative_00000093a_cut.bmp | Distance: 0.47897 | Classification: Non-Human
        2NN: TrainNegative_00000057a_cut.bmp | Distance: 0.47176 | Classification: Non-Human
        3NN: TrainPositive_crop001672b.bmp | Distance: 0.44589 | Classification: Human
    Prediction From 3NN: Non-Human


TestImage: TestNegative_00000118a_cut.bmp | Correct Classification: Non-Human
        1NN: TrainNegative_00000093a_cut.bmp | Distance: 0.56144 | Classification: Non-Human
        2NN: TrainNegative__00000053a_cut.bmp | Distance: 0.55531 | Classification: Non-Human
        3NN: TrainNegative_00000091a_cut.bmp | Distance: 0.54894 | Classification: Non-Human
    Prediction From 3NN: Non-Human
```

```
TestImage: TestNegative_no_person__no_bike_258_Cut.bmp | Correct Classification: Non-Human
        1NN: TrainNegative_00000057a_cut.bmp | Distance: 0.49611 | Classification: Non-Human
        2NN: TrainPositive_crop001672b.bmp | Distance: 0.48911 | Classification: Human
        3NN: TrainPositive_crop001275b.bmp | Distance: 0.48456 | Classification: Human
    Prediction From 3NN: Human


TestImage: TestNegative_no_person_no_bike_264_cut.bmp | Correct Classification: Non-Human
        1NN: TrainNegative__00000053a_cut.bmp | Distance: 0.4415 | Classification: Non-Human
        2NN: TrainNegative_01-03e_cut.bmp | Distance: 0.44046 | Classification: Non-Human
        3NN: TrainPositive_crop001672b.bmp | Distance: 0.43909 | Classification: Human
    Prediction From 3NN: Non-Human
```

Also showing the 10 Test Gradient Images as follows: