CS414 COMPUTER VISION

PROJECT DOCUMENT

“Detection of humans (pedestrians)”

1.0 Abstract

Our project idea (would ideally) has the aim to improve the safety of pedestrians and the rest of the traffic participants. We say ideally because the scale of such a project is too enormous for a university course project.

We will scale down to project to analysis of frames; primarily detecting pedestrians and traffic signs. That would include solving a number of computer vision challenges. Scaling, viewpoint variations, intra-class object variations, illumination problems especially at night or in bad weather conditions and quite possibly more.

After many hours of seeing materials on the web, primarily on youtube, we decided on using the (2.0) hog algorithm for pedestrian detection, creating step by step by using various youtube sources (find in references). For that, we needed CV Toolbox and ML Toolbox for matlab.

Afterwards, we decided on extracting the targeted areas with creating Historgrams of Oriented Gradients with the matlab function extractHogFeatures. It works on the principle of dividing the input in 8x16 squares, each square being 8x8 pixels of the purpose to get more precise details.

We also wanted to add another feature to our project, so we applied, partly our knowledge from lab assignments, to extract traffic signs by simple object recognition functions.

2.0 Algorithm for solving the issue of detection of Pedestrian and non-Pedestrian objects

STEP1: Load an image which hypothetically represents a frame of a video.
STEP2: Load training set (set of images – optional)
STEP3: Extract parts of the image
STEP4: call extractHogFeatures – HOG is histogram of Oriented Gradients
STEP5: Extract hog features from set of images (optional)
STEP6: Resize images to a constant height
STEP7: Extract HOG Features and HOG Visulization
STEP8: Assign labels
STEP9: Segmentation
STEP10: Test

2.1 Matlab Code for the Program

Current Implementation of Matlab code for a pedestrian detection system. This system uses a sliding window approach with a HOG feature classifier. To use this code, MatLab's Computer Vision System Toolbox and Statistics and Machine Learning Toolbox will need to be imported.

```
%%Load
the
testing
image
          testingImage = imread('singleped.jpg');
          [height, width, dim] = size(testingImage);
          disp('Dimension of Testing Image');
          disp(height);
          disp(width);
          figure;
          imshow(testingImage);
          windowHeightMin = 300;
          windowHeightMax = 530;
          windowWidth = 170;
          bboxes = [];

          for windowHeight = windowHeightMin:100:windowHeightMax

          %%Load training set
          trainDir = dir('H:\MITPedDataset\*.png');
          numTrainImages = length(trainDir);
          disp('Number of Training Images');
          disp(numTrainImages);

          imageDim = imread(strcat('H:\MITPedDataset\',trainDir(1).name));
          imageDim = imresize(imageDim, [windowHeight windowWidth]);

          [hog_4x4, vis4x4] = extractHOGFeatures(imageDim,'CellSize',[8 8]);
          cellSize = [10 10];
          hogFeatureSize = length(hog_4x4);
          disp('Size of Feature Vector');
          disp(size(hog_4x4));

          %%Extract HOG features from training set
          trainingFeatures = [];
          trainingLabels = [];

          features = zeros(numTrainImages, hogFeatureSize);
          labels = zeros(numTrainImages, 1);

          for j=1:numTrainImages
              currentImage = ...
                  strcat('H:\MITPedDataset\',trainDir(j).name);
              name = trainDir(j).name;
              img = imread(currentImage);
```

```
    % Resize images to a constant height
    img = imresize(img, [windowHeight windowWidth]);

    % Extract HOG Features and HOG Visulization
    [feat, visu] = extractHOGFeatures(img, 'CellSize', [8 8]);

    features(j, :) = feat;

    s = trainDir(j).name;

    % Assign labels based on image name
    if strfind(s, 'per')
        labels(j) = 1;
    end
    if strfind(s, 'crop')
        labels(j) = 1;
    end
end

trainingFeatures = [trainingFeatures; features];
trainingLabels = [trainingLabels; labels];
classifier = fitcsvm(trainingFeatures, trainingLabels);
disp('Training Set Completed & SVM trained');

%%Start with smallest segmentation size in left corner
pixelSlideY = 10;
pixelSlideX = 100;
slideY = 0;
slideX = 0;


for w = 0:pixelSlideX:(width - windowWidth)
    for h = 0:pixelSlideY:(height - windowHeight)
        for i = 1:windowHeight
            for j = 1:windowWidth
                temp(i, j, :) = testingImage(i + h, j + w,:);
            end
        end
        [tfeat, visu] = extractHOGFeatures(temp, 'CellSize', [8 8]);
        currSegmentLength = length(tfeat);
        [label, score] = predict(classifier, tfeat);
        disp(score);
        disp(label);
        if label == 1
            bboxes = [bboxes; [w h windowWidth windowHeight]];
        end
    end
end
end

test = insertObjectAnnotation(testingImage,'rectangle',bboxes,'Person');

figure;
imshow(test);
title('Peds Detected');
    %%Run classifier on features from image segment w training data
    %%features from
```

3.0 Detection of traffic signs

We also added a part in which we can recognize road and traffic signs.

3.1 Algorithm for detection of traffic signs

STEP1: set values for thresholding, seeds and limitations of size
STEP2: load the picture
STEP3: Resize
STEP4: Detect the sign
STEP5: Result

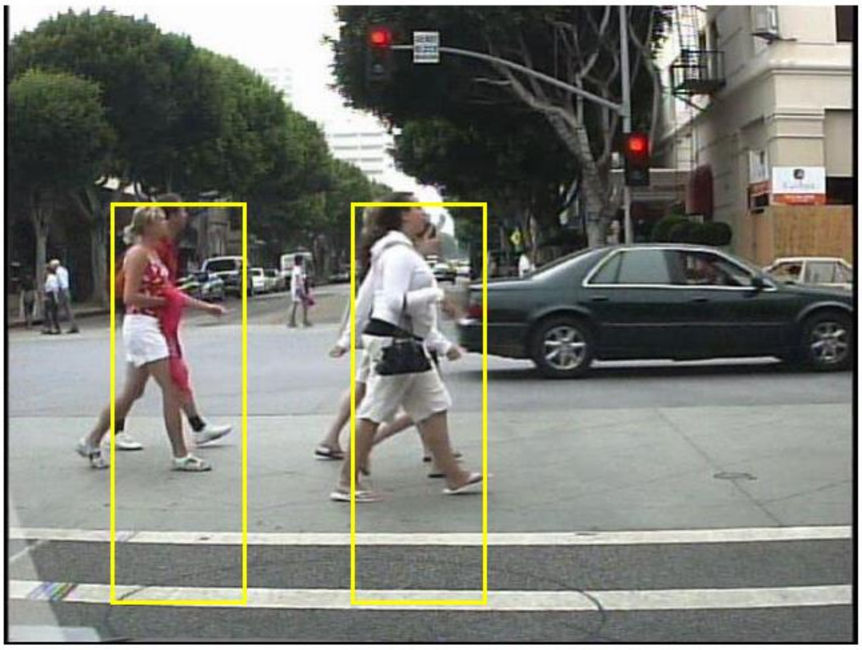
3.2 Matlab code for detection of traffic signs

```
Th = 0.01; %threshold
N = 19; %number of seeds per dimension
sideMax = 600;%maximum side length in pixels

%load
[filename, pathname] = uigetfile (‘’);
Im = importdata ([pathname, filename]);

%resize
Scale = sidemax/(max(size(im(:, : ,:1))));
Im = imresize(im, scale*size(Im(:, :, 1)));

%%detect the sign
Mask = getSign (im, th, n, sideMax);
Bbox = regionprops(Mask, ‘BoundingBox’);
Bbox = bbox.BoundingBox;
SignImg = imcrop(im, bbox);

%result
Figure(‘Position’, [153, 137, 1610, 780]);
Subplot(1,2,1), imshow(Im), title(‘Input image’);
Subplot(1,2,2), imshow(Im);
```

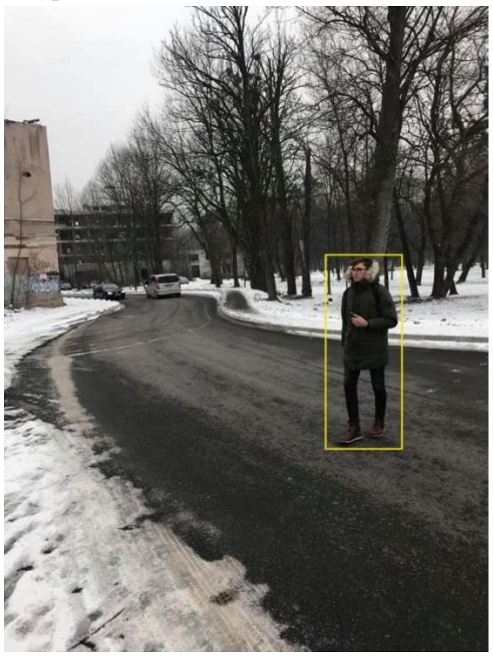
```
Hold on, rectangle (‘Position’, bbox, ‘Edge color’, ‘g’, ‘LineWidth’, 3);
Hold off, title (‘Processed image’);
Figure(), imshow(SignImg), title(‘Detected sign’);
```

2.2 Resulting images

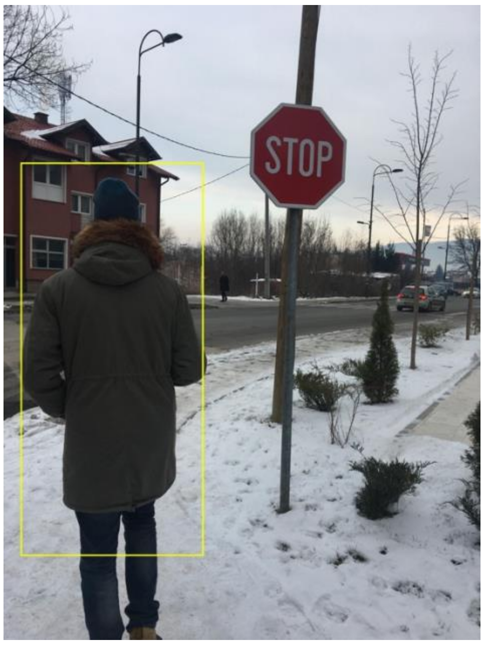
Original photo



Pedestrians detected on original photo



STOP


STOP

Ilidža
Hrasnica

STOP

Ilidža
Hrasnica

STOP

3.0 Summary

We learned about the many ways we can extract pedestrians from frames hence why it has such a wide use mainly in car industry. In this code, we applied Computer Vision System Toolbox and Statistics and Machine Learning Toolbox functions in Matlab.

3.1 References

https://www.mathworks.com/help/vision/ref/extracthogfeatures.html
https://www.youtube.com/watch?v=28xk5i1_7Zc
https://www.youtube.com/watch?v=4ESLTAd3IOM

Table of Contents