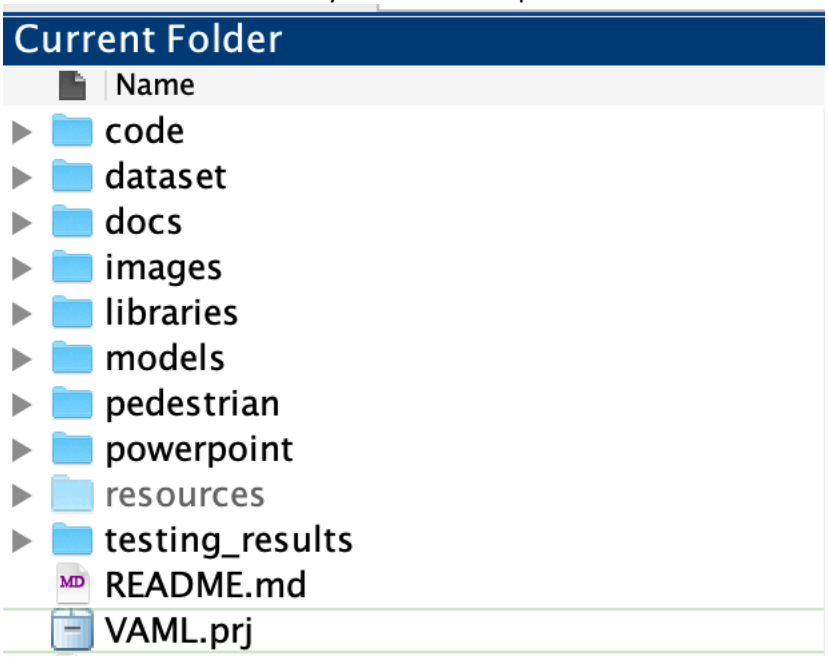
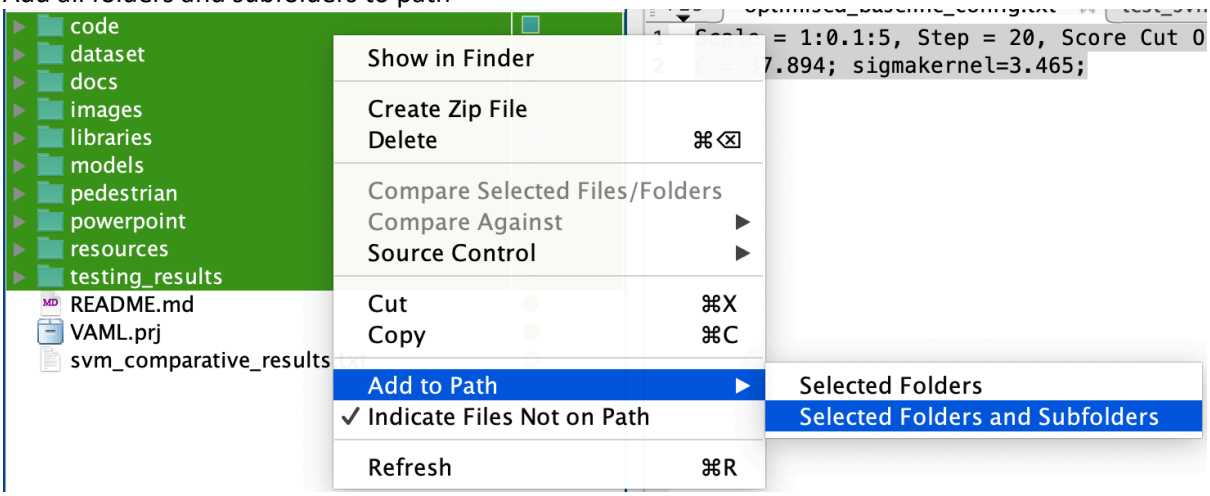
Initial Set-Up

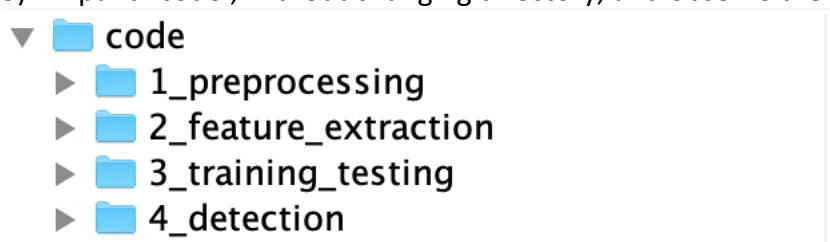
1) Ensure the current directory is set to the top-level folder



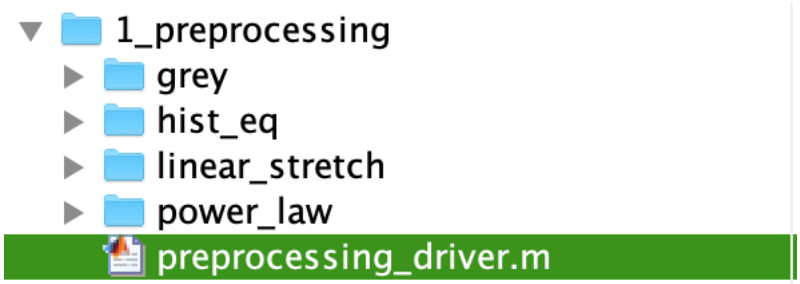
2) Add all folders and subfolders to path

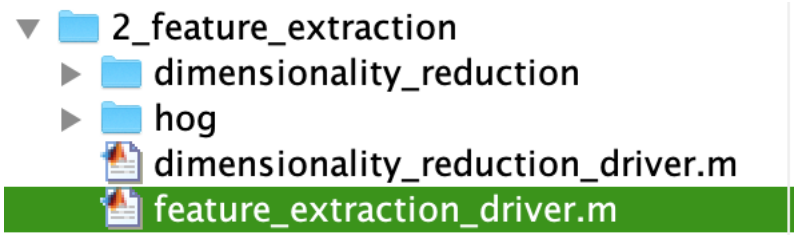


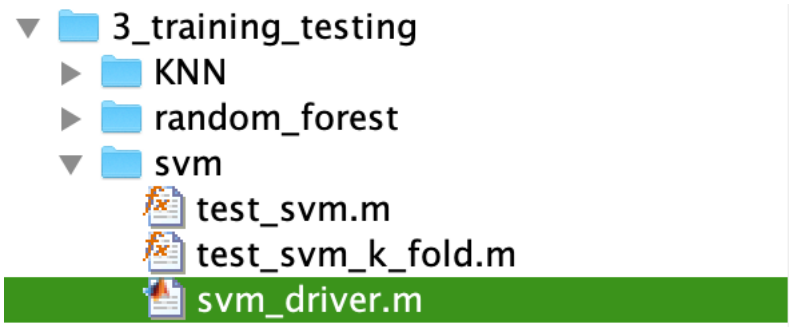
3) Expand 'code', without changing directory, and observe the following



You will note that each folder is numbered. This defines the order each 'driver' must be executed.



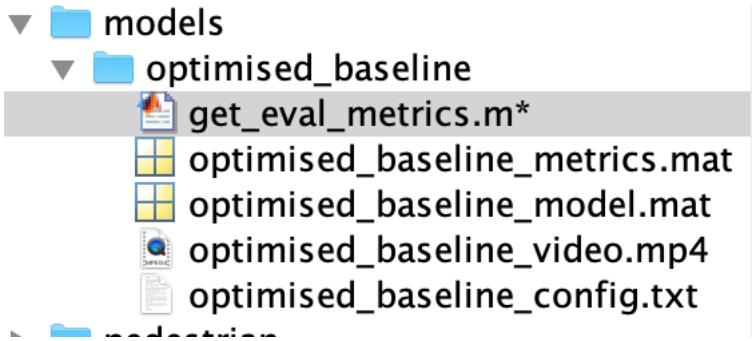




4) Simply run ‘preprocessing_driver’, ‘feature_extraction_driver’, ‘svm_driver’ and ‘sliding window_driver’ in sequence to set up the final optimal solution we reached in our report.

****Note****

If you simply want to obtain the final results go to the following folder



This contains:
- The script for generating the evaluation metrics from the optimised_baseline_metrics file.
- The final video.
- The exact configuration used for the final solution
- The model itself

User Manual

Training

Contrast Enhancement	Feature Extraction	Dimensionality Reduction (Non-Essential)	Learning Algorithm
HE	HOG	LDA	SVM
PL		PCA	KNN
LS			RF
GS			

1) Select one from each column above, for example: PL – HOG – SVM
2) In the driver files comment out all but your chosen option, run each driver

preprocessing_driver.m

```
%------------------------------POWER_LAW-----------------------------------%

[allPL] = power_law(path);
```

feature_extraction_driver.m

```
%-------------------------POWER_LAW-HOG-----------------------------------%

% Generate the hog feature set for Power Law images
[hogPL] = generate_hog(allPL);
```

svm_driver.m

```
%-----------------------POWER_LAW-HOG-SVM-------------------------------%

% 1) Partition
[trainingPL, testingPL] = partition_hold_out(hogPL, labels, 0.2);

% 2) Train the SVM model using libsvm - changed to 100% of images
% trainedSVMModelPL = SVMTraining(hogPL, labels(:,1));

trainedSVMModelPL = SVMTraining(trainingPL(:,2:size(trainingPL,2)), trainingPL(:,1));
```

Success, you have now trained the model.

Testing

Hold-Out

1) Leave only steps 3 and 4 uncommented, click run in the learning algorithm driver

svm_driver.m, KNN_driver.m or RF_driver.m

```
%------------------------POWER_LAW-HOG-SVM-------------------------------%

% 1) Partition
%[trainingPL, testingPL] = partition_hold_out(hogPL, labels, 0.2);

% 2) Train the SVM model using libsvm - changed to 100% of images
% trainedSVMModelPL = SVMTraining(hogPL, labels(:,1));

%trainedSVMModelPL = SVMTraining(trainingPL(:,2:size(trainingPL,2)), trainingPL(:,1));

% 3) Testing the model (~97% accuracy)
[predictionsPL, accuracyPL] = test_svm(testingPL, trainedSVMModelPL);

% 4) Show confusion matrix and summary statistics
conf_mat(testingPL, predictionsPL);
```

K-Fold

1) Ensure K is set

svm_driver.m, KNN_driver.m or RF_driver.m

```
3 -     k = 5;
```

2) Leave only the section below uncommented, click run

svm_driver.m, KNN_driver.m or RF_driver.m

```
%------------------------POWER_LAW-HOG-SVM-------------------------------%

% 1) Partition
%[trainingPL, testingPL] = partition_hold_out(hogPL, labels, 0.2);

% 2) Train the SVM model using libsvm - changed to 100% of images
% trainedSVMModelPL = SVMTraining(hogPL, labels(:,1));

%trainedSVMModelPL = SVMTraining(trainingPL(:,2:size(trainingPL,2)), trainingPL(:,1));

% 3) Testing the model (~97% accuracy)
% [predictionsPL, accuracyPL] = test_svm(testingPL, trainedSVMModelPL);

% 4) Show confusion matrix and summary statistics
% conf_mat(testingPL, predictionsPL);

% 5) Save Model for use
%save detectorModel trainedSVMModelPL;

% K-Fold
[accuraciesPL, predictionsPL, evalMetricsPL] = test_svm_k_fold(hogPL, labels, k);
```

Success, you have now tested the model.

Detection

Using our Pretrained Optimal Model

1) Open ‘sliding_window_driver.m’ in ‘code/4_detection’, simply click run.

This will run the sliding window using our pretrained optimised model stored in ‘models/optimised_baseline/optimised_baseline_model.mat'.

The output of the run will be saved to a video called ‘pedestrian.mp4’.

Using your Own Model

1) Change the path on line 6 of ‘sliding_window_driver’ to the path of you model you created in the training phase, click run

```
5      % Loads the detector model created before
6 -    load('models/optimised_baseline/optimised_baseline_model.mat');
```

Success, you have now started detecting pedestrians and generating the output video. In a few minutes you should begin to see each pedestrian image appear on screen, with its detections present.

Notes on Detection

If your PC cannot handle computationally intensive operations, decrease the scale range by uncommenting out the following code in ‘sliding_window_driver’.

```
15 -    scaleX = 1:0.1:5;
16
17      % scaleX = 1:0.2:3;
```

And consider increasing ‘step’ to 40

```
46 -    parfor y = 1:size(scaleX, 2)
47 -        step = 20;
48 -        object_ = SingleDetection(scaleX(y) * 3, scaleX(y), step, proc_images{i}, tra
49 -        Objects = [Objects; object_];
50 -    end
```