File Execution Sequence

**prepare\_image.ipynb**

**extract\_features.ipynb**

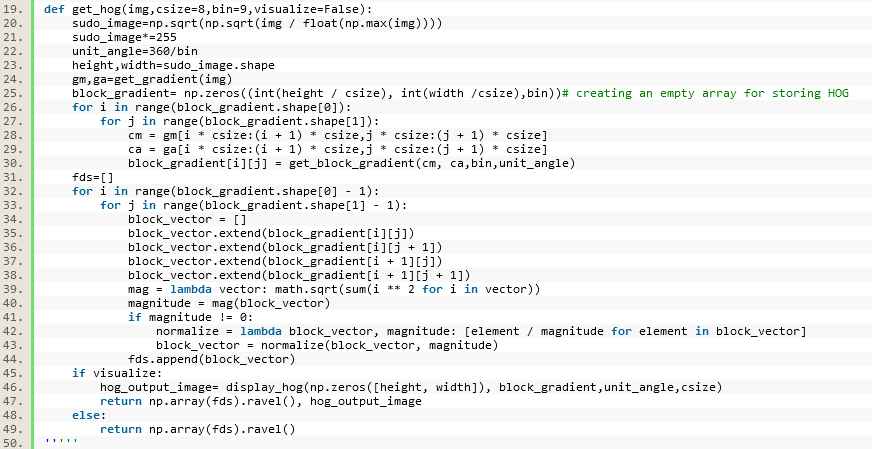
**svm\_model.ipynb**

**svm\_detection.ipynb**

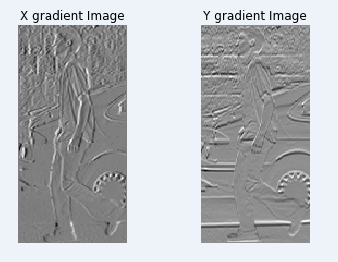
**random\_forest\_moded.ipynb**

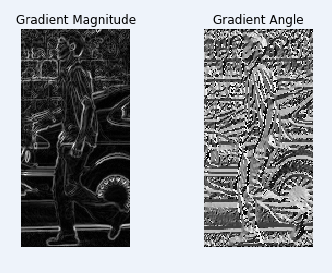
**rdf\_detection.ipynb**

1. HOG Implementation

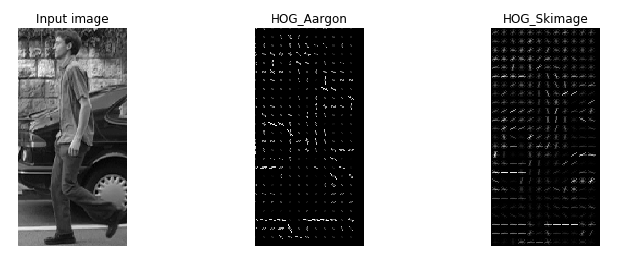


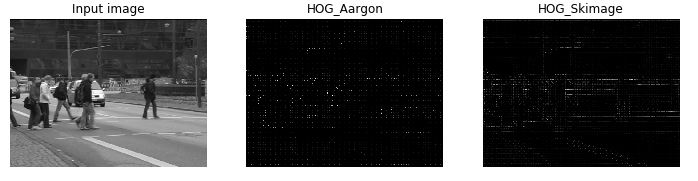
1. Gradient (Gx, Gy)

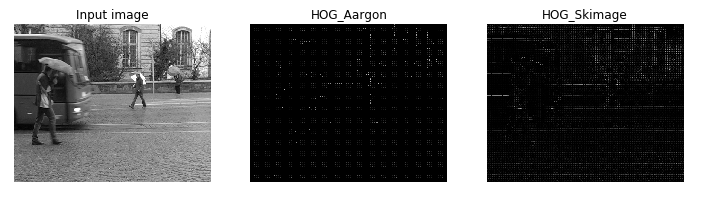




1. HOG output Visualization. Greyscale pixel values are used to represent magnitudes and arrows to represent orientations





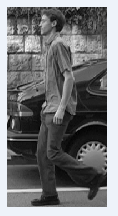


1. Generate positive and negative instance. The positive and negative instances which are generated are kept in two separate directories

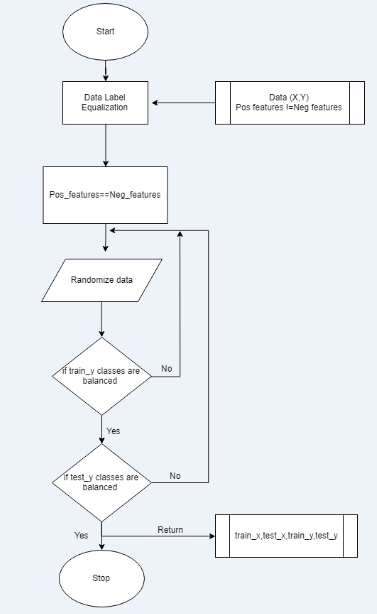
…\data\total\_neg



…\data\train\_pos

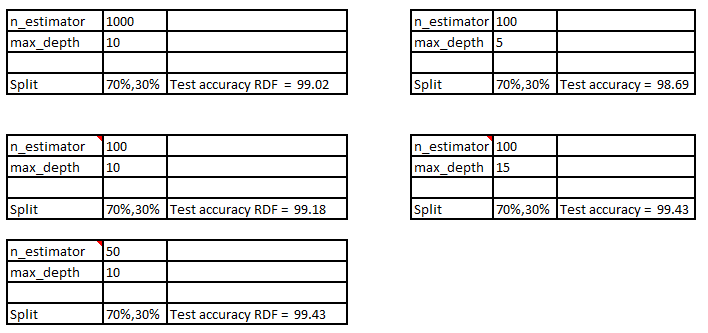


1. Class-balanced random train/test split using sklearn.model\_selection methods

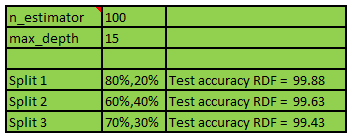


1. Train an Sklearn random decision forest on feature vectors generated using your HOG

function from (1)



Optimal



1. Grid-based cross validation

0.9969 (+/-0.0083) for {'max\_depth': 5, 'n\_estimators': 200}

0.9979 (+/-0.0051) for {'max\_depth': 5, 'n\_estimators': 400}

0.9969 (+/-0.0083) for {'max\_depth': 5, 'n\_estimators': 600}

0.9969 (+/-0.0083) for {'max\_depth': 5, 'n\_estimators': 800}

0.9969 (+/-0.0083) for {'max\_depth': 5, 'n\_estimators': 1000}

0.9990 (+/-0.0041) for {'max\_depth': 10, 'n\_estimators': 200}

0.9990 (+/-0.0041) for {'max\_depth': 10, 'n\_estimators': 400}

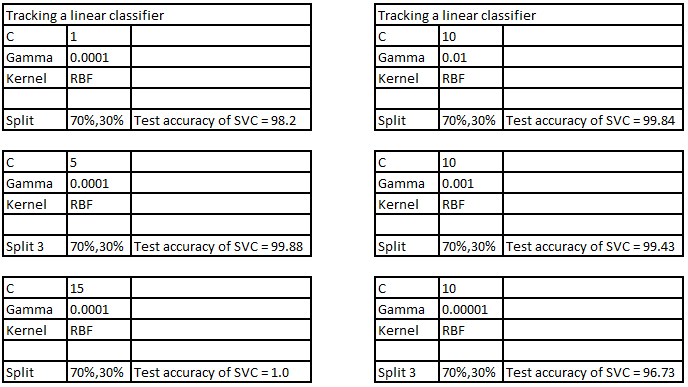
0.9990 (+/-0.0041) for {'max\_depth': 10, 'n\_estimators': 600}

0.9990 (+/-0.0041) for {'max\_depth': 10, 'n\_estimators': 800}

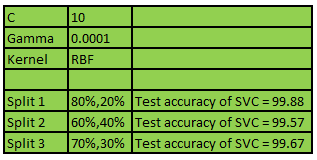
0.9979 (+/-0.0051) for {'max\_depth': 10, 'n\_estimators': 1000}

max\_depth -> Change in max depth from 5 to 10 increased the grip score

1. Training an Sklearn support vector classifier on feature vectors generated using your HOG function from (1)



Optimal



1. Use grid-based cross validation to optimize the support vector classifier hyper parameters

0.9884 (+/-0.0129) for {'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}

0.9705 (+/-0.0124) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}

0.9457 (+/-0.0156) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}

0.9932 (+/-0.0099) for {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}

0.9903 (+/-0.0148) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}

0.9705 (+/-0.0124) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}

0.9932 (+/-0.0099) for {'C': 15, 'gamma': 0.01, 'kernel': 'rbf'}

0.9932 (+/-0.0099) for {'C': 15, 'gamma': 0.001, 'kernel': 'rbf'}

0.9742 (+/-0.0127) for {'C': 15, 'gamma': 0.0001, 'kernel': 'rbf'}

0.9942 (+/-0.0113) for {'C': 1, 'kernel': 'linear'}

0.9942 (+/-0.0113) for {'C': 10, 'kernel': 'linear'}

0.9942 (+/-0.0113) for {'C': 15, 'kernel': 'linear'}

It is both dependent on C and gamma parameter

1. Apply model to the test Image

