**1. Vanishing Point Estimation in Image**

1) 목적

본 과제의 목적은 Vanishing Point를 찾아내는 것이다. Vanishing Point는 n개의 직선의 교점에서 찾을 수 있는데, 이 과제에서는 가장자리(Edge)를 detection한 뒤에 가장자리의 교점들 중 가장 많은 빈도로 나타나는 교점을 Vanishing Point로 확인할 수 있다.

Image1과 Image2의 코드는 parameter의 값들을 제외하면 동일하기 때문에 image 1의 코드를 통해 결과를 해석해보도록 하겠다.

2) 결과

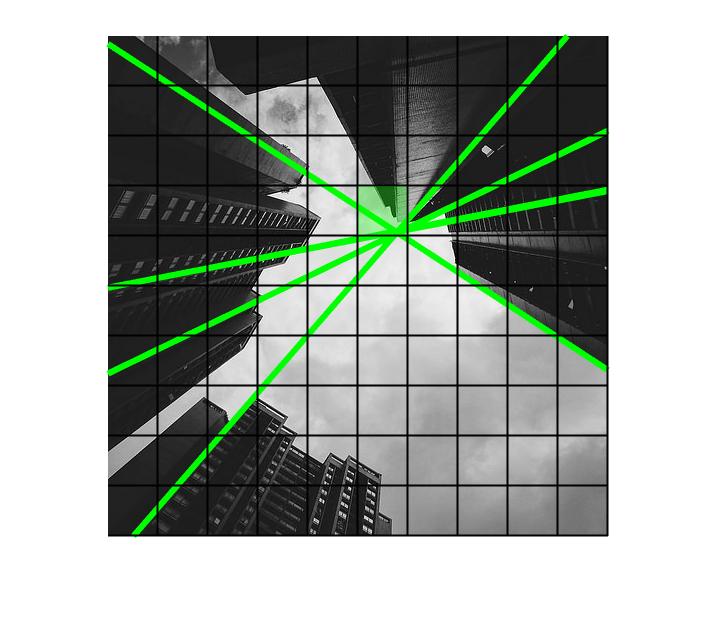


image1\_output.jpg

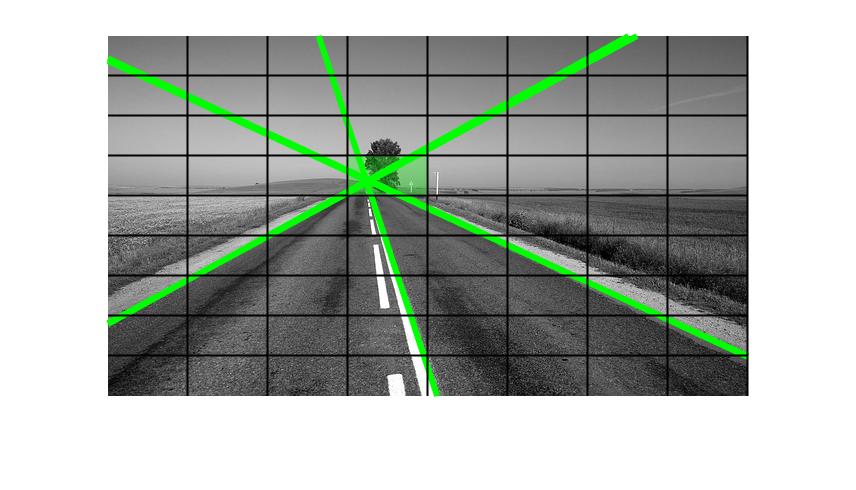
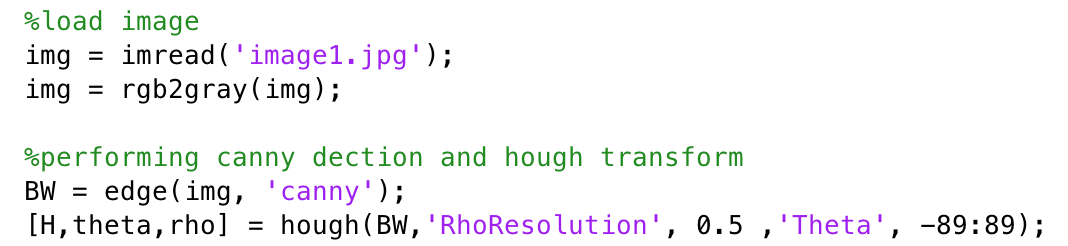
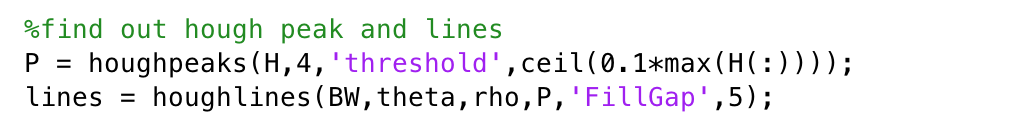


image2\_output.jpg

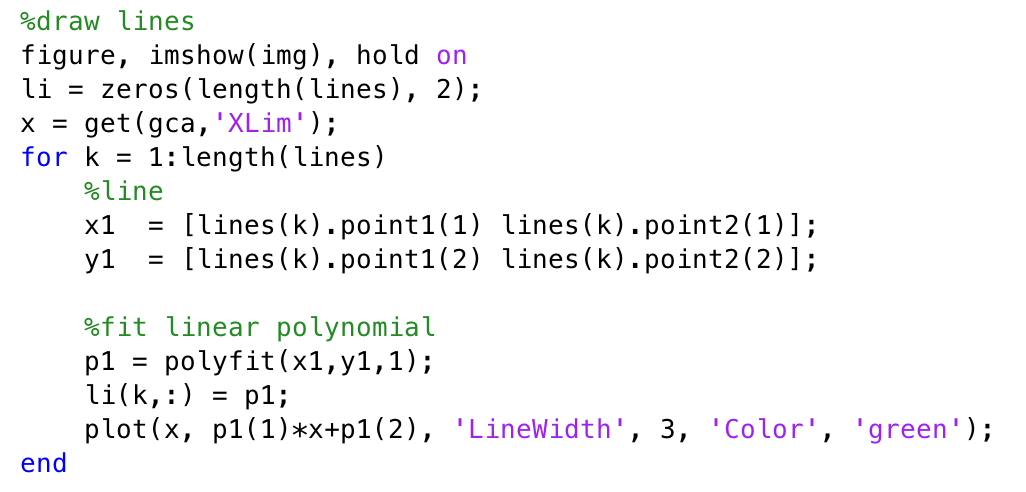
3) 코드 분석



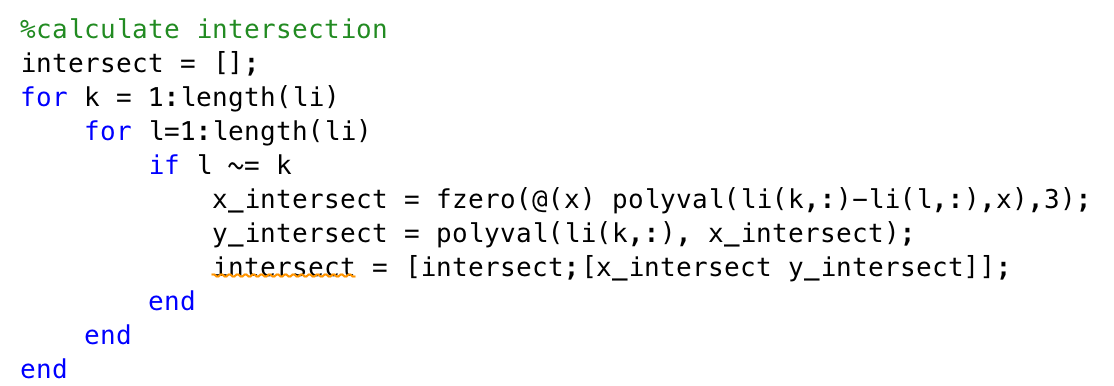
우선, 이미지를 불러온 뒤에 RGB channel을 Gray Channel로 변환시킨다. 그리고 Canny Edge detection을 통해 Edge을 찾아낼 수 있다. Edge들 중에서 Line을 이루는 점들을 걸러내기 위해서 Hough transform을 시행하는데, 이 때 origin (1,1) 에서부터 Line까지의 거리를 Rho Resolution parameter를 통해 설정하게 된다. Theta는 origin에서 Line까지의 수선이 이루는 각을 뜻하는데, image1에서는 모든 방향에서 line이 나타나야 하므로 한계 값이 없도록 설정하였다.



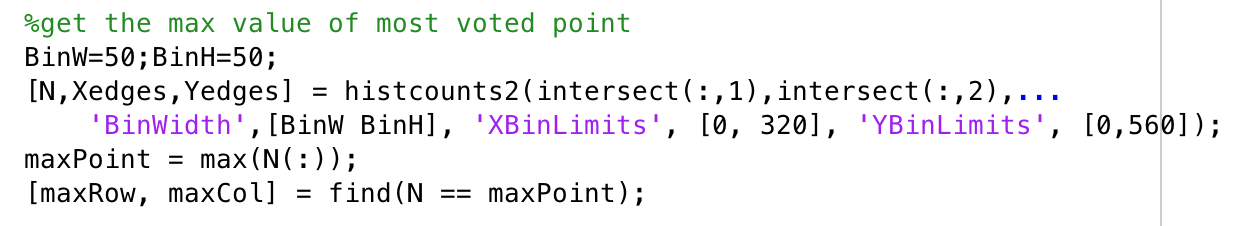
houghpeaks 함수를 통해서 hough function의 max값 중 일부를 찾아낸다. 여기서는 max값의 0.1만큼 threshold를 주고 이 중에서 4개를 선택하도록 하였다. 여기서 얻어진 값들을 통해 line을 이루는 점들을 찾을 수 있다. P에서 선택된 값들과 hough함수를 통해 얻은 theta 와 rho값을 parameter로 하여 line을 규명한다. 결과값은 structure로 저장된다.



Line을 이루는 두 점들이 li의 하나의 structure로 구성되어 있다. 우리는 이 두 점들의 x, y값을 통해서 실제 line을 그려낼 수 있다.



한 line이 자신을 제외한 다른 line들과 접하는 교점을 모두 구한다. 이를 통해 가장 많은 vote를 받은 점이 vanishing point가 된다.



이미지를 일정 크기의 Bin Size로 나눈 뒤에 교점들이 지닌 값을 통해 voting map을 가지도록 한다. 이를 통해 어떤 bin에서 가장 많은 voting이 이루어졌는지 알 수 있다.

**2. Object Classification**

1) 목적

본 과제를 통해서 Support Vector Machine을 통해 Class을 나누는 작업을 진행한다. 이 때, 사용되는 feature는 Histogram of Oriented Gradients을 이용하여 이미지들의 feature을 추출하도록 한다. 여기서 우리는 5가지의 이미지를 사용하기 때문에, Label은 5개가 된다. 따라서 One vs Rest 전략을 통해 Classify을 진행한다.

Dataset은 각 Label 별로 50개의 이미지를 사용하고 이 중 각 30개는 Training Set, 나머지 20개씩의 이미지는 Test Set로 사용이 된다.

Label 1 = airplane

Label 2 = Car side

Label 3 = Chair

Label 4 = Elephant

Label 5 = Ferry

2) 결과

airplane : Accuracy = 100% (100/100) (classification)

Car side : Accuracy = 100% (100/100) (classification)

Chair : Accuracy = 85% (85/100) (classification)

Elephant : Accuracy = 95% (95/100) (classification)

Ferry :Accuracy = 86% (86/100) (classification)

acc = 0.8600

C =

20 0 0 0 0

0 20 0 0 0

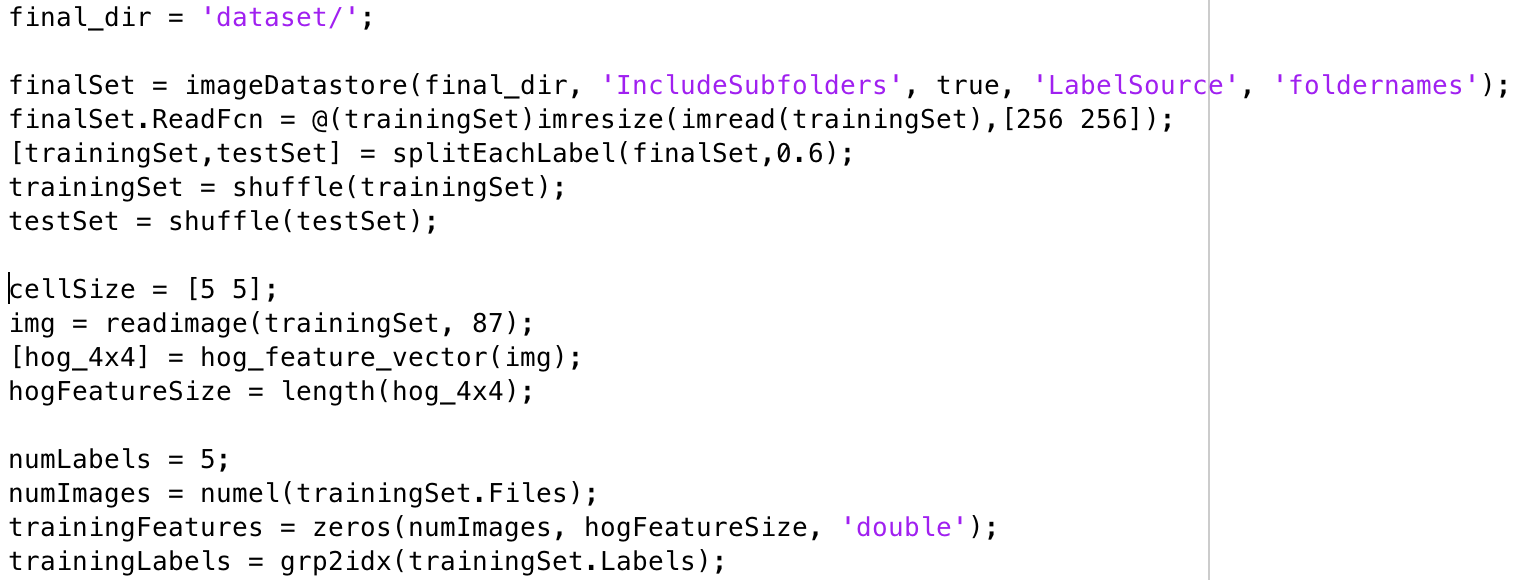
1 0 12 3 4

0 0 0 19 1

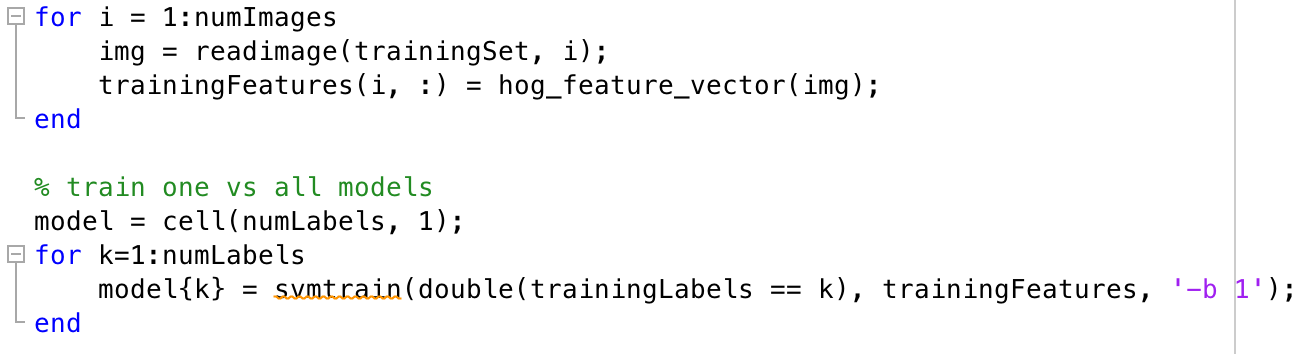
0 0 0 5 15

위의 Confusion Table에 의해 Airplane과 Car Side는 20개의 test set 모두 classification이 성공했음을 확인할 수 있다 (100%). Chair의 경우, 12개만 정확하게 맞추었고 Airplane/ Elephant/ Ferry로 혼동을 하는 것을 확인가능하다. 나머지도 비슷하게 해석이 가능하다.

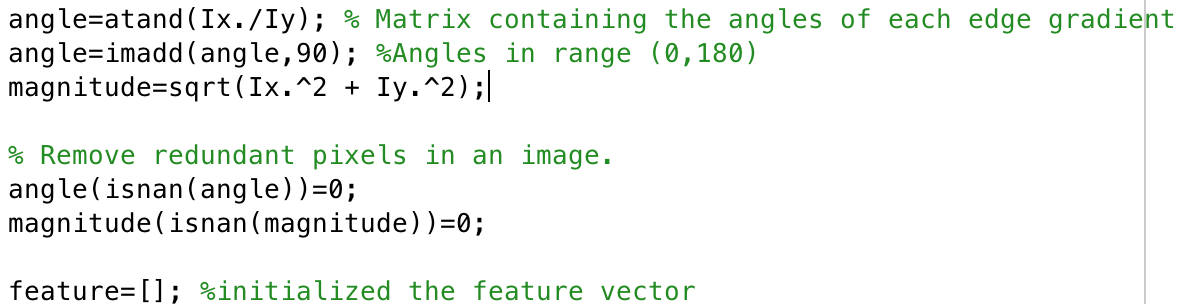
3) 코드 분석



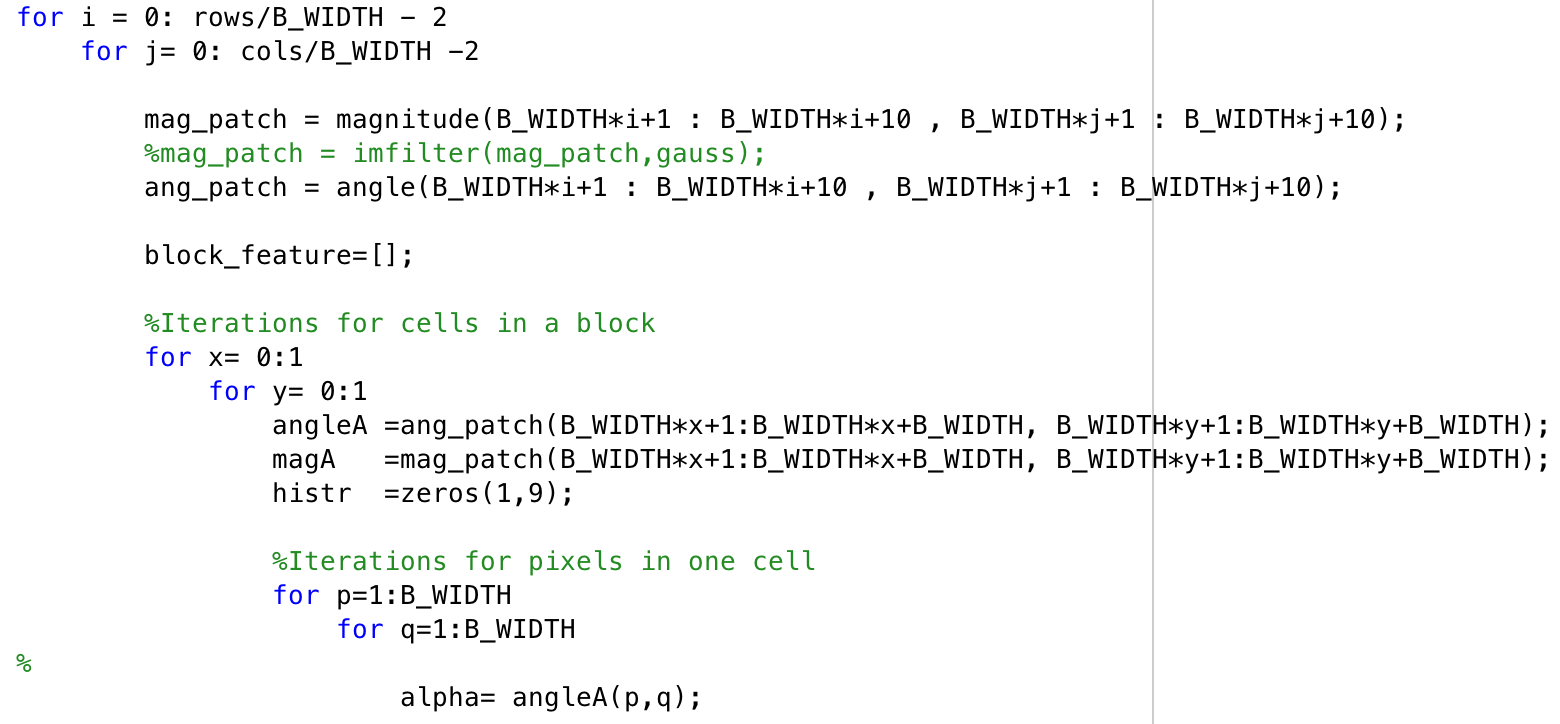
dataset 폴더 내에서 folder name에 따라서 Label을 나누었다. 이 dataset 중에서 앞의 60%는 training set으로 사용되며, 40%는 test set으로 사용되었다. 이렇게 나누어진 Training Set과 Test Set은 [256, 256] size로 사이즈가 조절된 이후 순서가 무작위로 Shuffle된 이후에 사용된다.



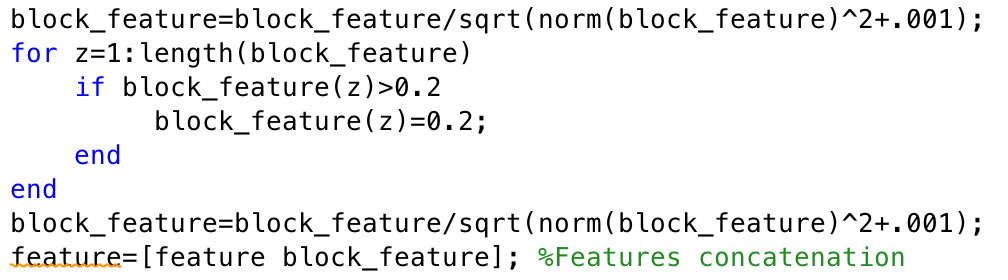
모든 Training Set에 있는 이미지들은 HOG를 통해 Feature를 추출해 낸다. 이를 통해 추출한 150개 이미지들의 feature들을 통해 각 label에 해당하는 모델을 train할 수 있다. Train은 one vs rest 전략을 취하는데, 위의 코드와 같이 해당 label에 적합한 feature는 1로 잡고 그 이외의 feature들은 0으로 Labelling 이 이루어진다. 이를 각각의 label에 적용한다. 예측 값을 알기 위해서 ProbA와 ProB를 알아야만 한다. 따라서, ‘-b 1’ 옵션을 활용한다. 이는 svmpredictd에서도 사용하여 probability estimate vector를 추출한다.



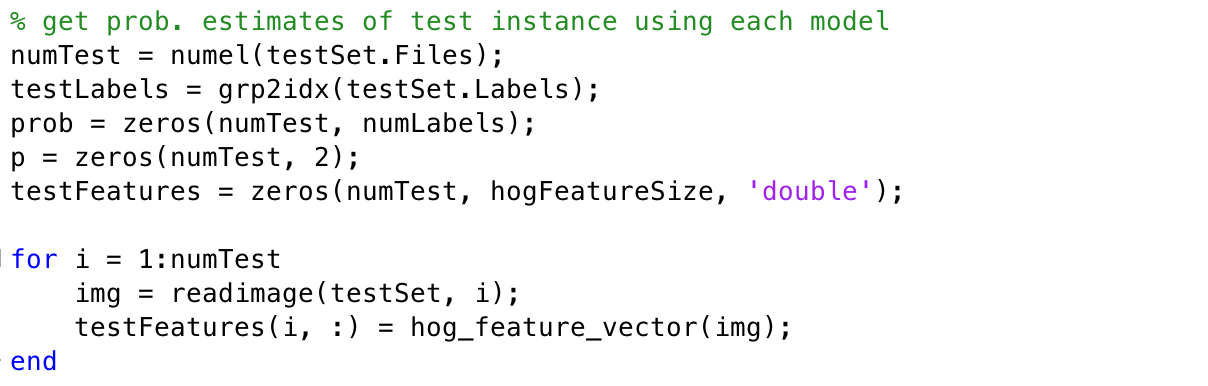
HOG는 각 픽셀에서 방향과 크기를 가진 gradient vector를 추출하여 feature를 만들어낸다.



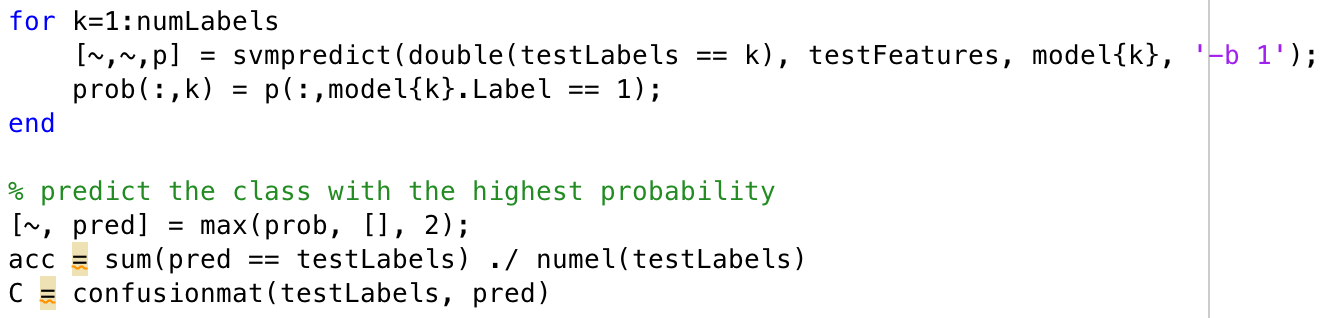
Block size가 [5 5]이므로 B\_width는 5를 뜻하며, stide가 50%이므로 각 픽셀들을 단위 5씩 이동하면서 feature들을 추출하게 된다. 이 때 bin size가 9이므로 20씩 끊어져서 bin을 구성하게 된다.



이렇게 하나의 block에 feature들을 추출한 뒤, L2 Normalization을 시행하게 된다. 이를 통해 보다 빛에 덜 민감한 feature를 만들어낼 수 있다. L2 Normalization 이후 feature가 0.2보다 큰 경우는 0.2로 조정을 한 뒤 다시 Normalization을 거친다. (L2 Hys)



Train Set과 동일하게 Test Set에서도 HoG Feature를 추출해낸다.



추출한 Test Feature를 모델을 통하여 각 Label 별로 예측을 한다.