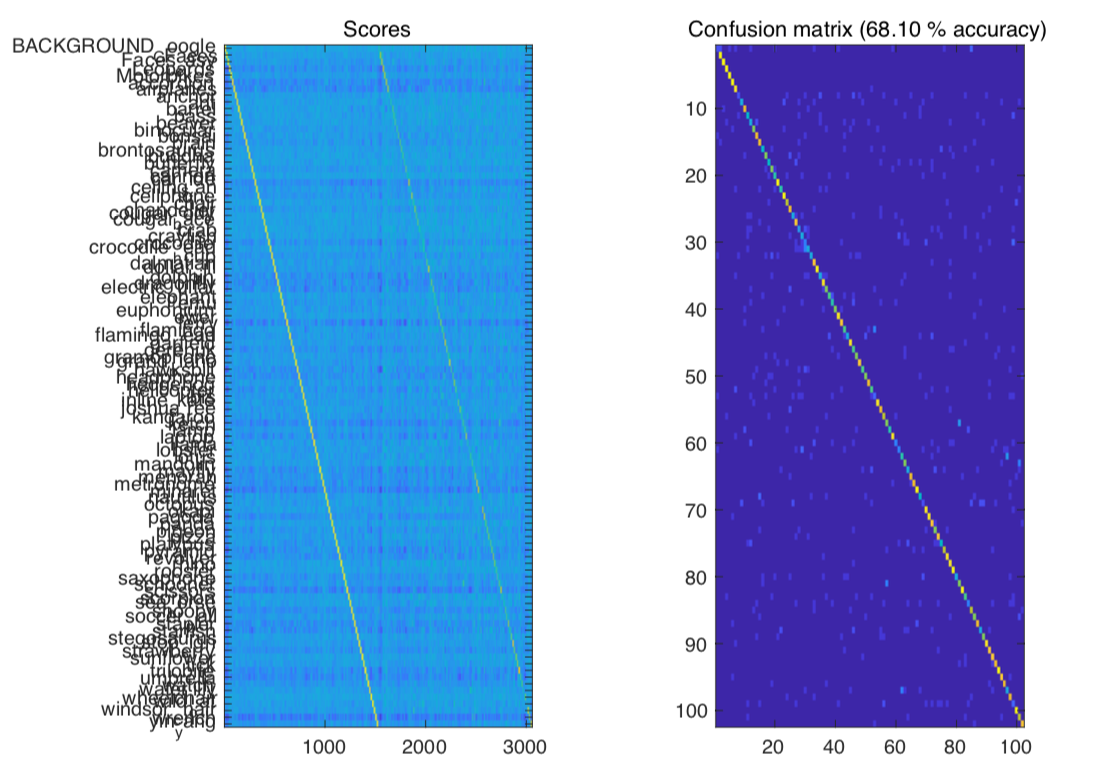
**Computer Vision Final Project**

**Using Spatial pyramid Caltech101 test accuracy**

소프트웨어학과 201723274

이서현

* **Result Summary**



<사진 1. 결과 Score와 Confusion matrix>

이번 Bag of Word project에서 test 정확성을 67.58%에서 68.10%로 upgrade했다. 사진 1의 confusion matrix를 보면 68.10% accuracy를 나타낸다. Confusion matrix는 간단하게 supervised training algorithm의 성능을 시각화 할 수 있는 표이다. 파란색에 가까울 수록 정확성이 높은 것이고 빨간색에 가까울 수록 정확성이 낮은 것이다.

성능을 높이기 위해 **Spatial Pyramid Matching**을 이용해서 input image을 n 등분하여 영역마다 histogram을 따로 계산하여 spatial information을 더하고자 했다. 또한 **k-means clustering**에서 k 의 값을 구하기 위해 다양한 k 값을 대입해보았다. Input image를 절반으로 나눠서 각자의 histogram을 독립적으로 계산해서 spatial information을 추가했다. K값은 30, 50, 100, 300을 대입해보았고 k=100에서 가장 높은 accuracy가 나타났다.

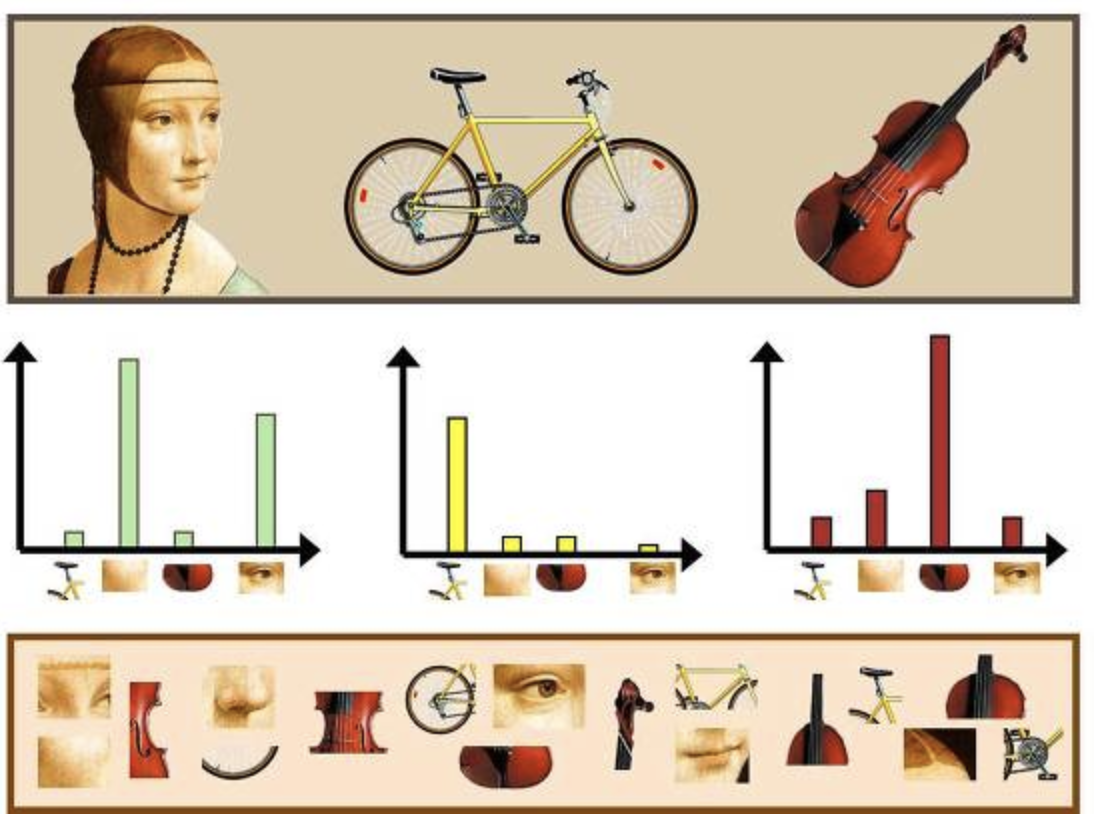
* **Table of content**

1. Bag of words에 대한 이론
   1. Image categorization
   2. Bag of words
   3. Spatial pyramid matching
2. Source code 해석
3. Result
4. 보충할 점
5. 참고
6. **Bag of words에 대한 이론**
   1. Image categorization

Image를 category로 나누기 위해서는 training과 testing 파트로 나눠진다. Training에서는 training image가 입력되면 image의 feature를 추출한다. 추출된 image feature와 입력된 training labels를 이용해서 classifier training을 한다. 그 결과 trained된 classifier가 나오게 된다. Testing 파트는 image의 feature를 추출하고 trained된 classifier를 이용해서 prediction을 출력한다.

* 1. Bag of words

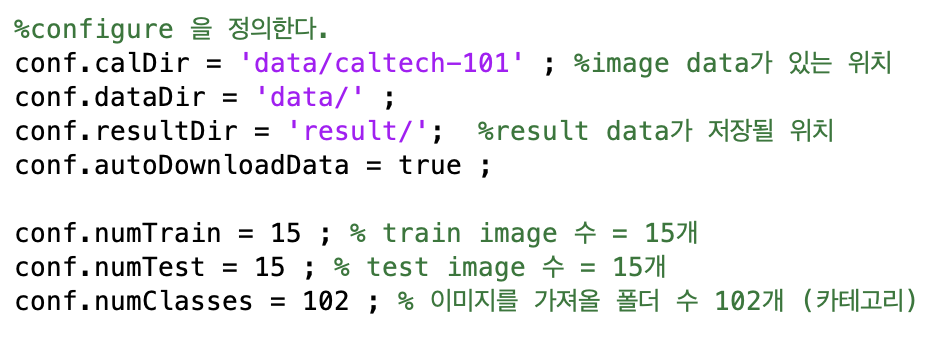
영상에서 feature을 추출한 뒤 feature를 representation할 수 있는 값인 code로 이루어진 codebook을 만든다. Codebook은 많은 image에서 추출한 feature들에 대해 k-means clustering을 수행하고 여기서 만들어진 각 cluster center인 대표 feature로 구성된다. Codebook으로 각각의 image를 representation할 수 있다. Codebook의 codeword로 image의 특성을 histogram으로 표현할 수 있다. Histogram에는 codeword가 image에 나타난 개수를 센 것이다. 요약하자면 Bag of words는 사진 2처럼 histogram이 동일한 image에서는 비슷하게 나타나고 다른 image에서는 다른 histogram이 나타난다.

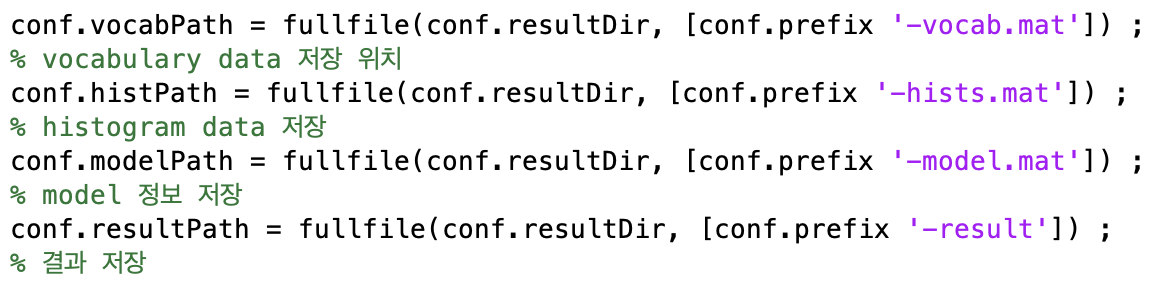
<사진 2. Bag of words>

* 1. Spatial pyramid matching

Spatial pyramid matching은 Bag of words의 accuracy를 높이기 위한 방법이다. Histogram은 동일하나 입력된 영상은 다른 것을 나타내는 단점을 보완한다. Input image에 level에 따라 image를 나눠서 나눈 영역에 대한 histogram을 각자 구해주는것이다. 즉 서로 독립적인 spatial bin에 따라 histpgram을 구해주는 것이다. 영역을 나눠주어서 histogram에는 spatial information이 더해진다.

1. **Source code 해석**
   1. **phow\_caltech101.m**





% --------------------------------------------------------------------

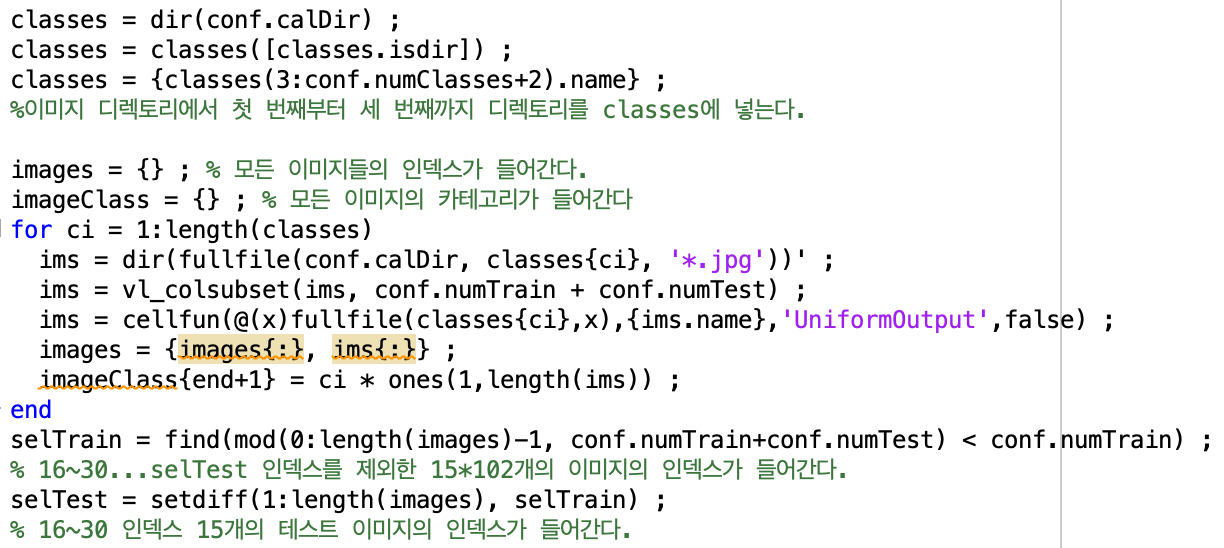
% Download Caltech-101 data

% --------------------------------------------------------------------

% --------------------------------------------------------------------

% 디렉토리에서 이미지를 불러온다. Setup data

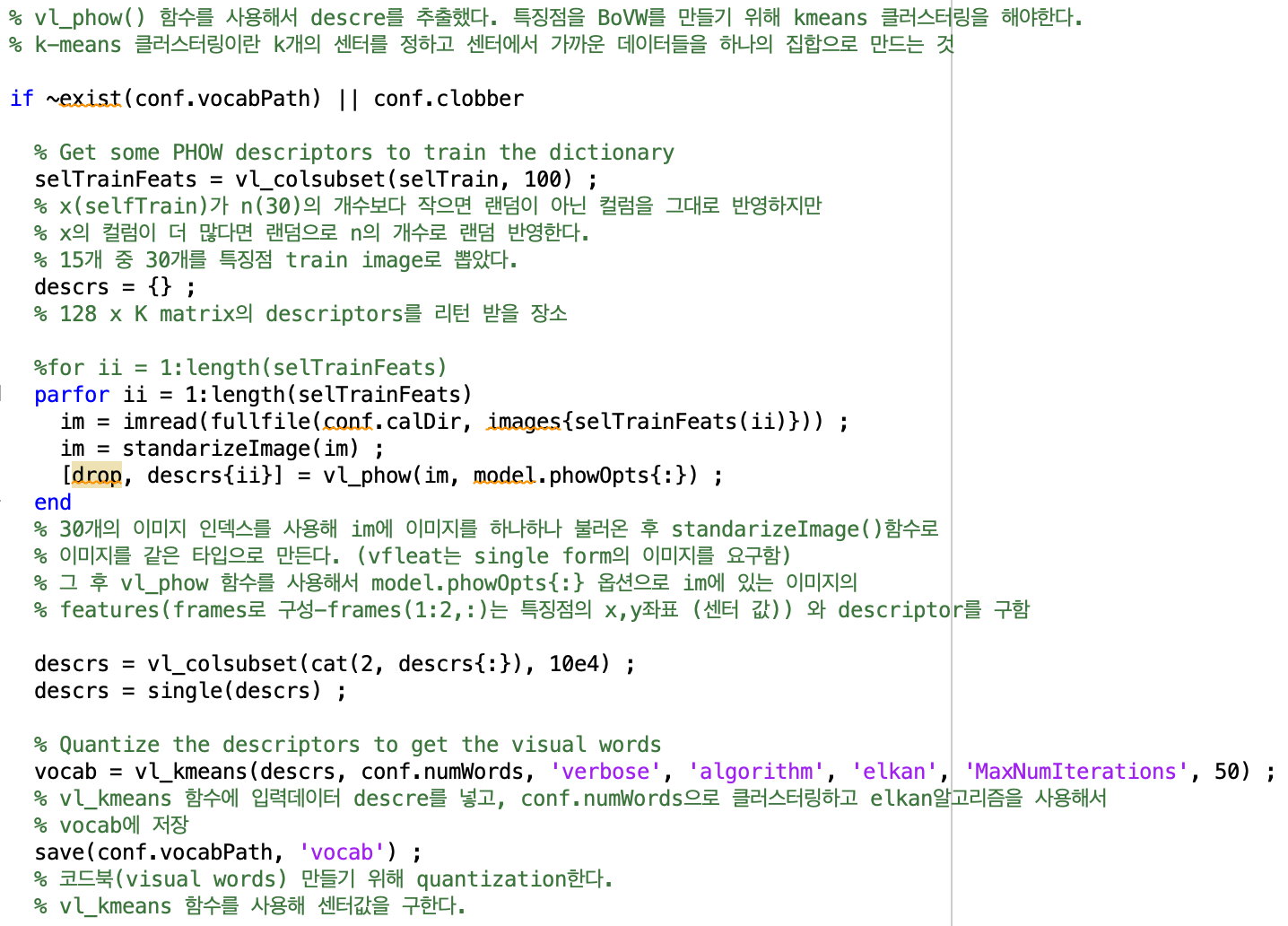
% --------------------------------------------------------------------



% --------------------------------------------------------------------

% vocab 만드는 과정 Train vocabulary

% --------------------------------------------------------------------



if strcmp(model.quantizer, 'kdtree')

model.kdtree = vl\_kdtreebuild(vocab) ;

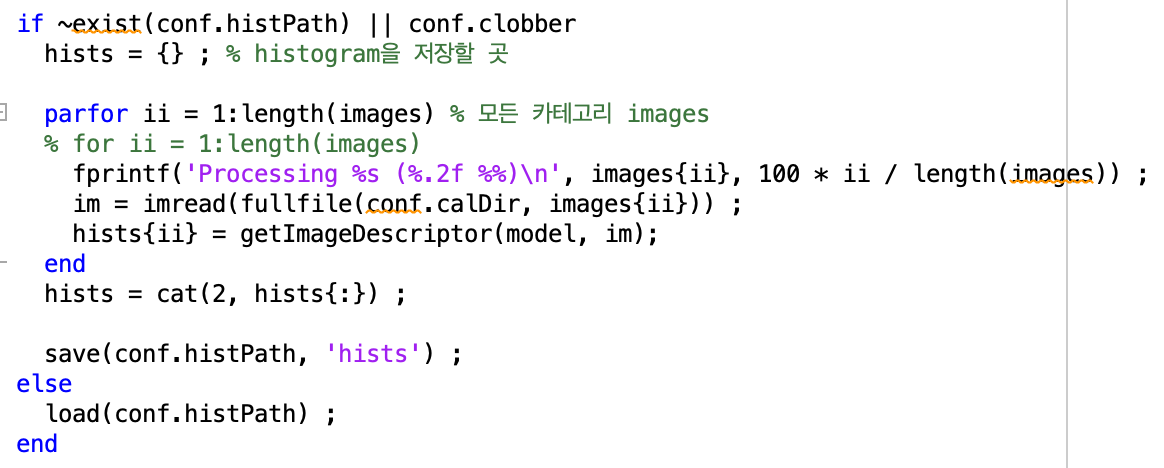
end

% vl\_kdtreebuild(x) x 데이터의 인덱스로 kd-tree forest를 만든다.

% --------------------------------------------------------------------

% histogram을 만드는 과정 Compute spatial histograms

% --------------------------------------------------------------------



% --------------------------------------------------------------------

% hists로 커널 맵 구성 (linear SVM solver PEGASOS를 train하기 위해) Compute feature map

% --------------------------------------------------------------------

psix = vl\_homkermap(hists, 1, 'kchi2', 'gamma', .5) ;

**2.1 getImageDescriptor.m**

% get PHOW features

[frames, descrs] = vl\_phow( im, model.phowOpts{:}) ;

% 이미지 im의 f,d 키포인트의 위치와 속성을 나타낸다. decres는 4x4x18 방향을 나타낸다.



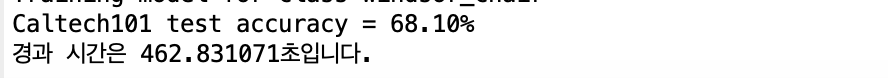
* 1. **standarizeImage.m**

% 이미지 type을 single로 변경

% 이미지 double을 single로 만든다.

% 이미지의 크기가 480을 넘으면 resize 한다.

1. **Result**



<사진 3. 출력값>

SIFT algorithm을 이용한 결과값은 67.58%로 출력되었다. SIFT algorithm보다 accuracy를 높이기 위해서 spatial pyramid matching 방법을 구현하는 것에 중점을 두었다. Spatial image는 하나의 input image가 histogram을 계산했을 때 다른 image와 histogram와 똑같을 문제을 보완한다. 각 영역을 나눠서 각자의 bin에 따른 histogram을 계산하는 것이다. Image를 나누기 위해 getImageDiscriptor.m에 코드를 추가해보았다. phow\_caltect101.m에서 image를 받아오면 하나의 image의 전체적인 histogram을 계산하는데 이 때는 spatial information이 없기 때문에 정확성이 낮다.



<사진 4. getImageDiscrptor.m의 일부분 >

Spatial pyramind를 구현하기 위해 Image의 size를 반으로 나눠서 각각 i의 범위를 설정해주었다. Image를 반으로 나눠 feature를 추출하고 각각 영역에 따른 독립적인 histogram을 계산한다. 즉, 각자 spatial bin에 따른 histogram을 계산하는 것이다. 이와 같이 spatial information을 주었더니 accuracy가 상승하는 것을 관찰할 수 있었다.

**selTrainFeats = vl\_colsubset(selTrain, 100) ;**

위 코드는 phow\_caltech101.m의 Train vocabulary 영역이다. selTrain의 개수가 100보다 작으면 랜덤이 아닌 column을 그대로 반영하지만 selTrain의 개수가 더 많다면 랜덤으로 수를 뽑아서 특징점의 개수를 설정한다. 즉 k-means algorithm의 k의 값을 설정하는 것이다. 30, 50, 100, 300을 넣어봤지만 100이 가장 정확도가 좋았다.

1. 보완할 점

<사진 5. getImageDiscriptor.m 오류>

Spatial Pyramid에서 level1인 image를 4등분하는 코드를 구현해보고 싶었지만, 사진 5와 같은 오류가 계속 발생하였다. 4등분으로 정확도를 크게 높여보고 싶었지만 배열 인덱스의 적절한 값을 넣지 못해 발생하는 오류를 잡아내지 못해서 아쉬웠다. Image를 어떻게 분리하여 spatial pyramid의 성능을 극대화시킬 수 있는지 고민해보고 성능이 좋아지는 다른 방법도 생각해볼 것이다.

1. 참고

- Computer Vision: Algorithms and Applications <Richard Szeliski>

- Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories <Svetlana Lazebnik1, Cordelia Schmid2, Jean Ponce1,3>

- Bag of Words 기법 : <https://darkpgmr.tistory.com/125>

- k-meanns algorithm: https://ratsgo.github.io/machine%20learning/2017/04/19/KC/