VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
VELLORE ▪ CHENNAI
www.vit.ac.in

IMAGE CONTENT ANALYSIS
OBJECT DETECTION

IMAGE PROCESSING (CSE 4019)

NAME	REGISTRATION NO.
Shreyash kawalkar	16BCE0221

SLOT: G1 + TG

Under the guidance of **Prof. Rajakumar K.**

VIT University, Vellore

ACKNOWLEDGEMENT

We acknowledge our sincere gratitude to Prof. Rajakumar K that he gave us the permission to do this project work. It was only with his backing and support that we could complete our project work. Without his support We couldn't even start the work.

Our sincere gratitude and thanks towards one of our senior Mr. Savin Malik. It was only with his backing and support that we could complete the report. He provided me all sorts of help and corrected me if ever seemed to make mistakes. We have no such words to express my gratitude.

We are very thankful to the information provided by the website which were very helpful for the making and successful completion of our project.

INTRODUCTION

Object detection and recognition, whose goal is to find out different objects in a given image. It Is the technology in the field of computer vision for finding and identifying objects in an image or video sequence. Humans recognize a multitude of objects in images with little effort, despite the fact that the image of the objects may vary somewhat in different viewpoints, in many different sizes and scales or even when they are translated or rotated. Objects can even be recognized when they are partially obstructed from view. This task is still a challenge for computer vision systems. Many approaches to the task have been implemented over multiple decades.

- The whole process begins with capturing an image using an external device and providing it as an input using Python.
- Certain simple interfaces help in creating User-MATLAB Interfaces that allows user to call MATLAB functions
- It begins with extracting the coordinates of objects from different angels and dimensions.
- The resultant input coordinates are compared with the measurements stored in the database and return the closest record.
- The analysis depends on best possible algorithmic approach called as image processing.
- The resulting match is sent back to python from the database and expressed on a suitable platform.

Object detection can be done by Different Colour Spaces – RGB, YCbCr, HIS, UCS (Perceptually uniform colour system) and Histogram-based Approaches – certain specific colour distribution model, Hair colour distribution model.

Then the dimensions and coordinates extraction is to be carry forwarded by Edge Detectors, Wavelet and wavelet packets, Discrete Cosine transformation, Gabor filters and several important statistics such as Moments, Shape/colour features, Fuzzified features.

Techniques to be used for object recognition are, a) Distance-wise classification,
K – nearest neighbour (K-NN), Neural Network, Support Vector Machine (SVM). Combining all these techniques, object recognition application will be developed.

Object detection and recognition is challenging due to the wide variety of distinct object having similar structures and the complexity of noises and image backgrounds. Moreover, implementing it in real life applications is also a difficult task itself.

SHORT DESCRIPTION OF PROJECT

- Training classes:

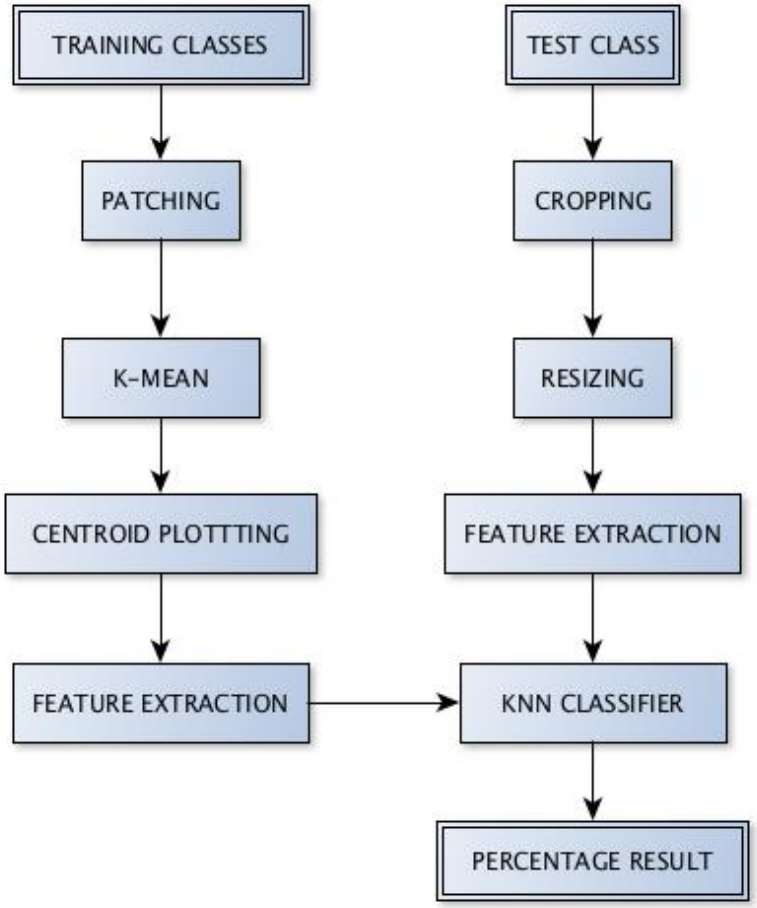
  The dataset is divided into five training batches and one test batch, each with 10000 images. Each training batches contains 10000 images of a particular class. Like, train_batch_1.mat contains aeroplanes, train_batch_2.mat contains cars, train_batch_3.mat contains ships, train_batch_4.mat contains trucks, train_batch_5.mat contains horses. These batches undergo numerous numbers of pre-processes and algorithms, in order to define their class characteristics.

- Test class:

  The test batch contains exactly 1000 randomly-selected images from each class. It is named as test batch.mat. It is loaded every time a a new class is tested. This batch undergoes lest number of processes comparatively to other batches.

- All image stored, read, generated are always reshaped to 32X32X3 format. These are RBG images.
- The images broken into small patches for detail examination are to be in 6X6X3 format.
- Number of patches to generated from the dataset is taken as 40000.
- Number of features to be extracted from a dataset, i.e, number of centroids is 320. More the patches, centroids, k-mean iterations, the more we get enhanced results.
- These patches undergo many process step by step, starting from patching, k-mean computation, centroid plotting, feature extraction and finally image recognition by k-nn classification.

PROCESSING FLOWCHART



DESCRIPTION OF IMPORTANT FUNCTIONS AND ALGORITHMS

K-MEAN:

k-mean is the simplest and significant enough method to classify the given training dataset. The main aim of this algorithm is to find k-centers.

Distances (our case: Euclidean distance) are calculated between each data-points, and clusters are formed by clubbing these data-points to the point nearest to them. This point is k-center. Further the k-center is recalculated again and again such that clusters breakdown into more clusters creating significant given number of clusters. In our case we have regenerated k-mean centers for 50 times.

But ***k*-means clustering** is a method of vector quantization, the input dataset must be in the form of eigen vector quantized. So, we quantize image patches into eigen value and directional vector.

```
C = cov(patches);
M = mean(patches);
[V,D] = eig(C);
P = V * diag(sqrt(1./(diag(D) + 0.1))) * V';
patches = bsxfun(@minus, patches, M) * P;
```

converts patches into k-mean compatible mode.

CENTROIDS PLOTTING:

The obtained k-means for the clusters formed in each patch are plotted for the user reference. Total number of centroids are is divided into mXn matrix (320 into 18X18; H=W=6). All centroids matrices are reshaped into 6X6 matrix are appended in a single matrix. Finally, the image is displayed to the user.

FEATURE EXTRACTION:

Under this section, features are extracted for all training images (10,000) of each class in their respective loops. For each image from a training class, all the 3 forms (R,G,B) are reshaped back to 32x32 from. Using im2col (image to column) function, theses 3 forms (R,G,B 32X32) are generated into 6X6 from. These generated form is stored as "patches", which are further normalized are Eigen vector quantized.

The most important processing is "triangle activation function", where sum of patches vectors, sum of centroid vector and product of patches and centroid vector are calculated, inorder to calculated average distance to centroids for each patch. So that location for class features can be deduced.

Patches is now the data matrix of activations for each patch. It is reshaped and divided into 4 quadrants. Further individual sum of all the 4 quadrants are taken and stored as q1, q2, q3 and q4. It is then appended in XC matrix row by row.

In this manner features extracted are stored for each class.

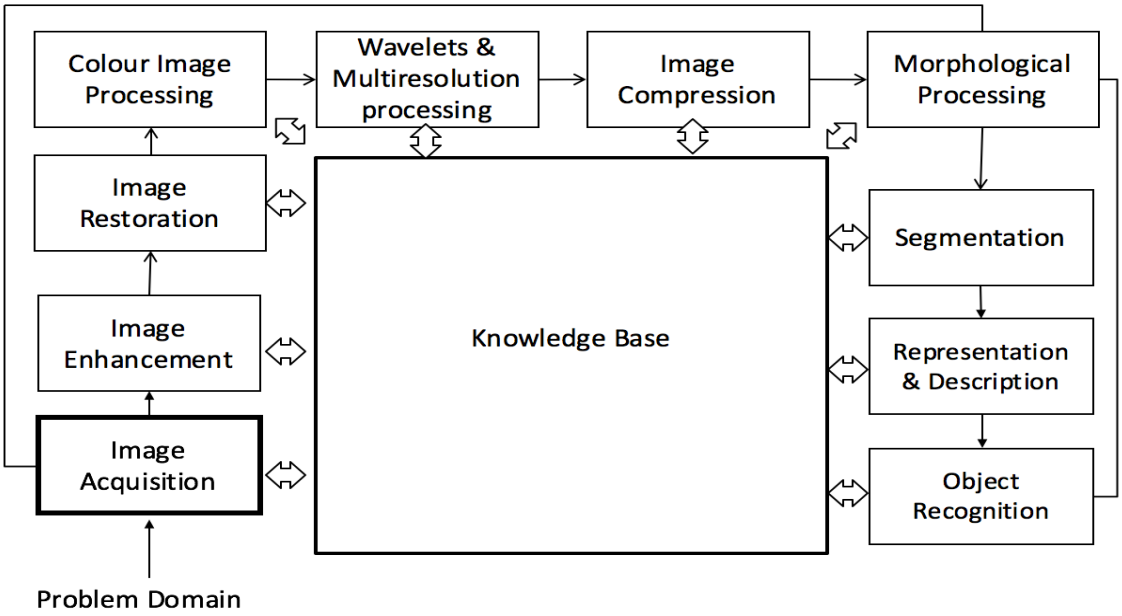
K-NN ALGORITHM:

k-NN algorithm is like an image search engine. The k-NN algorithm classifies unknown data points (feature vectors of training classes) by finding the most common class among the k-closest class (features extracted of test class).

This algorithm simply relies on the distance between feature vectors of training classes and test class. Distances between these feature vectors is calculated using distance formula like (our case: Euclidean method). Minimum the distance better is the correlation of the features. Moreover, in the dataset we have the *labels* associated with each image, this helps us to arrange the features of each training class according to test class. Labelling order of actual training

class and derived labels set is compared for best match. So, we can predict and return an actual *category* among training class for the given test image.

ASPECTS COVERED



CODE:

Kmeans_demo.m

```
clc
clear all
CIFAR_DIR='/Users/shreyashkawalkar/Downloads/cifar-10-batches-
mat/';
fname={'/Users/shreyashkawalkar/Downloads/cifar-10-batches-
mat//data_batch_1.mat','/Users/shreyashkawalkar/Downloads/cifa
r-10-batches-
mat//data_batch_2.mat','/Users/shreyashkawalkar/Downloads/cifa
r-10-batches-
mat//data_batch_3.mat','/Users/shreyashkawalkar/Downloads/cifa
r-10-batches-
mat//data_batch_4.mat','/Users/shreyashkawalkar/Downloads/cifa
r-10-batches-mat//data_batch_5.mat'};
rfSize = 6;
numCentroids=320;
whitening=true;
numPatches = 40000;
CIFAR_DIM=[32 32 3];
percentage=[0,0,0,0,0];

%training data
for index=1:5
fprintf('Loading training data. %f..\n',index);
f1=load(fname{index});
trainX = double([f1.data]);
trainY = double([f1.labels]);
clear f1;
% extract random patches
patches = zeros(numPatches, rfSize*rfSize*3);
for i=1:numPatches
  if (mod(i,10000) == 0) fprintf('Extracting patch: %d /
%d\n', i, numPatches); end

  r = random('unid', CIFAR_DIM(1) - rfSize + 1);%unique and
randomly r between 1 and 17
  c = random('unid', CIFAR_DIM(2) - rfSize + 1);
  patch = reshape(trainX(mod(i-1,size(trainX,1))+1, :),
CIFAR_DIM); % trainX ith column in 32 32 3 format
  patch = patch(r:r+rfSize-1,c:c+rfSize-1,:);%build patch 32
32 3 to 6 6 3 matrix with r ,c
  patches(i,:) = patch(:)';%at ith column of required no. of
patches, contain 6x6x3 patch values in column
end

% normalize for contrast
patches = bsxfun(@rdivide, bsxfun(@minus, patches,
```

```
mean(patches,2)), sqrt(var(patches,[],2)+10));

% whiten
if (whitening)
  C = cov(patches);
  M = mean(patches);
  [V,D] = eig(C);
  P = V * diag(sqrt(1./(diag(D) + 0.1))) * V';
  patches = bsxfun(@minus, patches, M) * P;
end

% run K-means
centroids = run_kmeans(patches, numCentroids, 50);
show_centroids(centroids, rfSize); drawnow;

% extract training features
if (whitening)
  trainXC = extract_features(trainX, centroids, rfSize,
CIFAR_DIM, M,P);
else
  trainXC = extract_features(trainX, centroids, rfSize,
CIFAR_DIM);
end

% standardize data
trainXC_mean = mean(trainXC);
trainXC_sd = sqrt(var(trainXC)+0.01);
trainXCs = bsxfun(@rdivide, bsxfun(@minus, trainXC,
trainXC_mean), trainXC_sd);
trainXCs = [trainXCs, ones(size(trainXCs,1),1)];

%test data
fprintf('Loading test data...\n');
f1=load([CIFAR_DIR '/test_batch.mat']);
testX = double(f1.data);
testY = double(f1.labels) + 1;
clear f1;

% compute testing features and standardize
if (whitening)
  testXC = extract_features(testX, centroids, rfSize,
CIFAR_DIM, M,P);
else
  testXC = extract_features(testX, centroids, rfSize,
CIFAR_DIM);
end
testXCs = bsxfun(@rdivide, bsxfun(@minus, testXC,
trainXC_mean), trainXC_sd);
testXCs = [testXCs, ones(size(testXCs,1),1)];

tht = knnclassify(testXCs,trainXCs, trainY, 1);
percentage(index)=1-((sum(tht)-sum(testY))/length(testY));
```

```
fprintf('Train accuracy %f%%\n', 1-((sum(tht)-
sum(testY))/length(testY)));
fprintf('Train accuracy %f%%\n', 100 * ((abs(sum(tht - testY))
/ length(testY))));
end
```

extract_features.m

```
function XC = extract_features(X, centroids, rfSize,
CIFAR_DIM, M,P)
  assert(nargin == 4 || nargin == 6);
  whitening = (nargin == 6);
  numCentroids = size(centroids,1);

  % compute features for all training images
  XC = zeros(size(X,1), numCentroids*4);
  for i=1:size(X,1)
    if (mod(i,1000) == 0) fprintf('Extracting features: %d /
%d\n', i, size(X,1)); end

    % extract overlapping sub-patches into rows of 'patches'
    patches = [ im2col(reshape(X(i,1:1024),CIFAR_DIM(1:2)),
[rfSize rfSize]) ;
                im2col(reshape(X(i,1025:2048),CIFAR_DIM(1:2)),
[rfSize rfSize]) ;
                im2col(reshape(X(i,2049:end),CIFAR_DIM(1:2)),
[rfSize rfSize]) ]';

    % do preprocessing for each patch

    % normalize for contrast
    patches = bsxfun(@rdivide, bsxfun(@minus, patches,
mean(patches,2)), sqrt(var(patches,[],2)+10));
    % whiten
    if (whitening)
      patches = bsxfun(@minus, patches, M) * P;
    end

    % compute 'triangle' activation function
    xx = sum(patches.^2, 2);
    cc = sum(centroids.^2, 2)';
    xc = patches * centroids';

    z = sqrt( bsxfun(@plus, cc, bsxfun(@minus, xx, 2*xc)) ); %
distances
    [v,inds] = min(z,[],2);
    mu = mean(z, 2); % average distance to centroids for each
patch
    patches = max(bsxfun(@minus, mu, z), 0);
    % patches is now the data matrix of activations for each
patch

    % reshape to numCentroids-channel image
```

```
    prows = CIFAR_DIM(1)-rfSize+1;
    pcols = CIFAR_DIM(2)-rfSize+1;
    patches = reshape(patches, prows, pcols, numCentroids);

    % pool over quadrants
    halfr = round(prows/2);
    halfc = round(pcols/2);
    q1 = sum(sum(patches(1:halfr, 1:halfc, :), 1),2);
    q2 = sum(sum(patches(halfr+1:end, 1:halfc, :), 1),2);
    q3 = sum(sum(patches(1:halfr, halfc+1:end, :), 1),2);
    q4 = sum(sum(patches(halfr+1:end, halfc+1:end, :), 1),2);

    % concatenate into feature vector
    XC(i,:) = [q1(:);q2(:);q3(:);q4(:)]';
  end
```

run_kmeans.m

```
function centroids = runkmeans(X, k, iterations)

  x2 = sum(X.^2,2);
  centroids = randn(k,size(X,2))*0.1;%X(randsample(size(X,1),
k), :);
  BATCH_SIZE=1000;


  for itr = 1:iterations
    fprintf('K-means iteration %d / %d\n', itr, iterations);

    c2 = 0.5*sum(centroids.^2,2);

    summation = zeros(k, size(X,2));
    counts = zeros(k, 1);

    loss =0;

    for i=1:BATCH_SIZE:size(X,1)
      lastIndex=min(i+BATCH_SIZE-1, size(X,1));
      m = lastIndex - i + 1;

      [val,labels] =
max(bsxfun(@minus,centroids*X(i:lastIndex,:)',c2));
      loss = loss + sum(0.5*x2(i:lastIndex) - val');

      S = sparse(1:m,labels,1,m,k,m); % labels as indicator
matrix
      summation = summation + S'*X(i:lastIndex,:);
      counts = counts + sum(S,1)';
    end
```

```
    centroids = bsxfun(@rdivide, summation, counts);

    % just zap empty centroids so they don't introduce NaNs
everywhere.
    badIndex = find(counts == 0);
    centroids(badIndex, :) = 0;
  end
```

show_centroids.m

```
function image = show_centroids(centroids, H, W)
  if (nargin < 3)%number of functions
    W = H;
  end
  N=size(centroids,2)/(H*W);
  assert(N == 3 || N == 1);  % color and gray images

  K=size(centroids,1);
  COLS=round(sqrt(K));
  ROWS=ceil(K / COLS);
  COUNT=COLS * ROWS;

  clf; hold on;
  image=ones(ROWS*(H+1), COLS*(W+1), N)*100;
  for i=1:size(centroids,1)
    r= floor((i-1) / COLS);
    c= mod(i-1, COLS);
    image((r*(H+1)+1):((r+1)*(H+1))-
1,(c*(W+1)+1):((c+1)*(W+1))-1,:) =
reshape(centroids(i,1:W*H*N),H,W,N);
  end

  mn=-1.5;
  mx=+1.5;
  image = (image - mn) / (mx - mn);
  figure,imshow(image);
```

kdemo.m

```
%%CIFAR_DIR='/path/to/cifar/cifar-10-batches-mat/';
CIFAR_DIR='/Users/shreyashkawalkar/Downloads/kmeans_demo-
master/kmeans_demo/';

assert(~strcmp(CIFAR_DIR, '/path/to/cifar/cifar-10-batches-
mat/'), ...
       ['You need to modify kmeans_demo.m so that CIFAR_DIR
points to ' ...
        'your cifar-10-batches-mat directory.  You can
download this ' ...
        'data from:  http://www.cs.toronto.edu/~kriz/cifar-10-
matlab.tar.gz']);
```

```
addpath minFunc;
rfSize = 6;
numCentroids=1600;
whitening=true;
numPatches = 400000;
CIFAR_DIM=[64 64 3];
img1(1).data=imread('cam1.jpg');
img1(2).data=imread('cam2.jpg');
img2(1).data=imread('bot1.jpg');
img2(2).data=imread('bot2.jpg');
img3(1).data=imread('bag1.jpg');
img3(2).data=imread('bag2.jpg');
img4(1).data=imread('bk1.jpg');
img4(2).data=imread('bk2.jpg');
img5(1).data=imread('cl1.jpg');
img5(2).data=imread('cl2.jpg');
img1(1).labels=1;
img1(2).labels=2;
img2(1).labels=3;
img2(2).labels=4;
img3(1).labels=5;
img3(2).labels=6;
img4(1).labels=7;
img4(2).labels=8;
img5(1).labels=9;
img5(2).labels=9;
save('ImageDatabase1.mat', 'img1');
save('ImageDatabase2.mat', 'img2');
save('ImageDatabase3.mat', 'img3');
save('ImageDatabase4.mat', 'img4');
save('ImageDatabase5.mat', 'img5');

fprintf('Loading training data...\n');
%f1=load([CIFAR_DIR 'ImageDatabase1.mat']);
%f2=load([CIFAR_DIR 'ImageDatabase2.mat']);
%f3=load([CIFAR_DIR 'ImageDatabase3.mat']);
%f4=load([CIFAR_DIR 'ImageDatabase4.mat']);
%f5=load([CIFAR_DIR 'ImageDatabase5.mat']);
trainX = double([img1.data; img2.data; img3.data; img4.data;
img5.data]);
trainY = double([img1.labels; img2.labels; img3.labels;
img4.labels; img5.labels]) + 1; % add 1 to labels!
clear f1 f2 f3 f4 f5;

patches = zeros(numPatches, rfSize*rfSize*3);
for i=1:numPatches
  if (mod(i,10000) == 0) fprintf('Extracting patch: %d /
%d\n', i, numPatches); end

  r = random('unid', CIFAR_DIM(1) - rfSize + 1);%unique and
randomly r between 1 and 17
  c = random('unid', CIFAR_DIM(2) - rfSize + 1);
```

```
  patch = reshape(trainX(mod(i-1,size(trainX,1))+1, :),
CIFAR_DIM); % trainX ith column in 32 32 3 format
  patch = patch(r:r+rfSize-1,c:c+rfSize-1,:);%build patch 32
32 3 to 6 6 3 matrix with r ,c
  patches(i,:) = patch(:)';%at ith column of required no. of
patches, contain 6x6x3 patch values in column
end

% normalize for contrast
patches = bsxfun(@rdivide, bsxfun(@minus, patches,
mean(patches,2)), sqrt(var(patches,[],2)+10));

% whiten
if (whitening)
  C = cov(patches);
  M = mean(patches);
  [V,D] = eig(C);
  P = V * diag(sqrt(1./(diag(D) + 0.1))) * V';
  patches = bsxfun(@minus, patches, M) * P;
end

% run K-means
centroids = run_kmeans(patches, numCentroids, 50);
show_centroids(centroids, rfSize); drawnow;

% extract training features
if (whitening)
  trainXC = extract_features(trainX, centroids, rfSize,
CIFAR_DIM, M,P);
else
  trainXC = extract_features(trainX, centroids, rfSize,
CIFAR_DIM);
end

% standardize data
trainXC_mean = mean(trainXC);
trainXC_sd = sqrt(var(trainXC)+0.01);
trainXCs = bsxfun(@rdivide, bsxfun(@minus, trainXC,
trainXC_mean), trainXC_sd);
trainXCs = [trainXCs, ones(size(trainXCs,1),1)];

% train classifier using SVM
C = 100;

%theta = train_svm(trainXCs, trainY, C);

%[val,labels] = max(trainXCs*theta, [], 2);
%fprintf('Train accuracy %f%%\n', 100 * (1 - sum(labels ~=
trainY) / length(trainY)));

%%%%% TESTING %%%%%
```

```
%% Load CIFAR test data
fprintf('Loading test data...\n');
f1=load([CIFAR_DIR '/test_batch.mat']);
testX = double(f1.data);
testY = double(f1.labels) + 1;
clear f1;

% compute testing features and standardize
if (whitening)
  testXC = extract_features(testX, centroids, rfSize,
CIFAR_DIM, M,P);
else
  testXC = extract_features(testX, centroids, rfSize,
CIFAR_DIM);
end
testXCs = bsxfun(@rdivide, bsxfun(@minus, testXC,
trainXC_mean), trainXC_sd);
testXCs = [testXCs, ones(size(testXCs,1),1)];

% test and print result
%[val,labels] = max(testXCs*theta, [], 2);
%fprintf('Test accuracy %f%%\n', 100 * (1 - sum(labels ~=
testY) / length(testY)));
tht = knnclassify(testXCs,trainXCs, trainY, 1);
fprintf('Train accuracy %f%%\n', 100 * (1 - sum(tht ~= testY)
/ length(testY)));
```

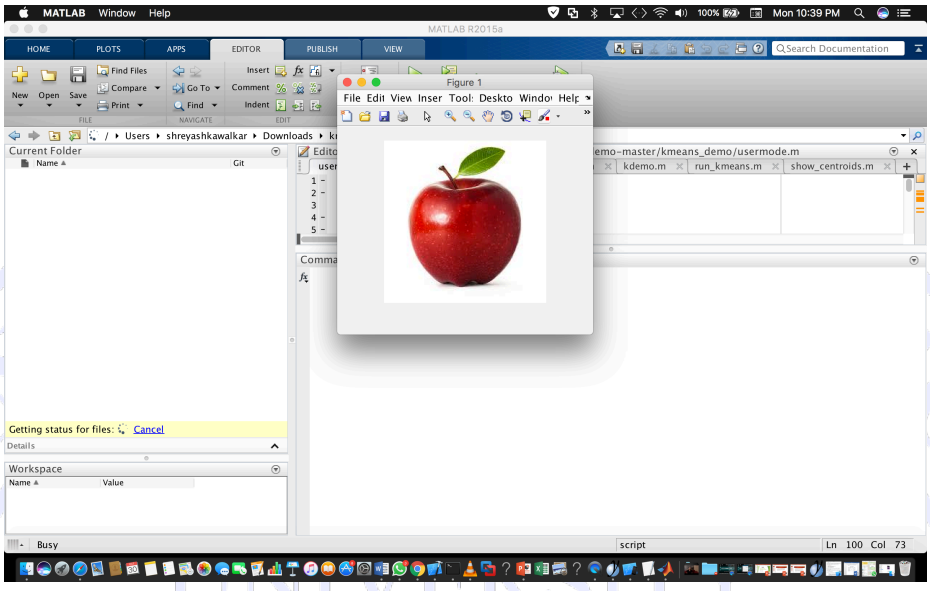
user_mode:

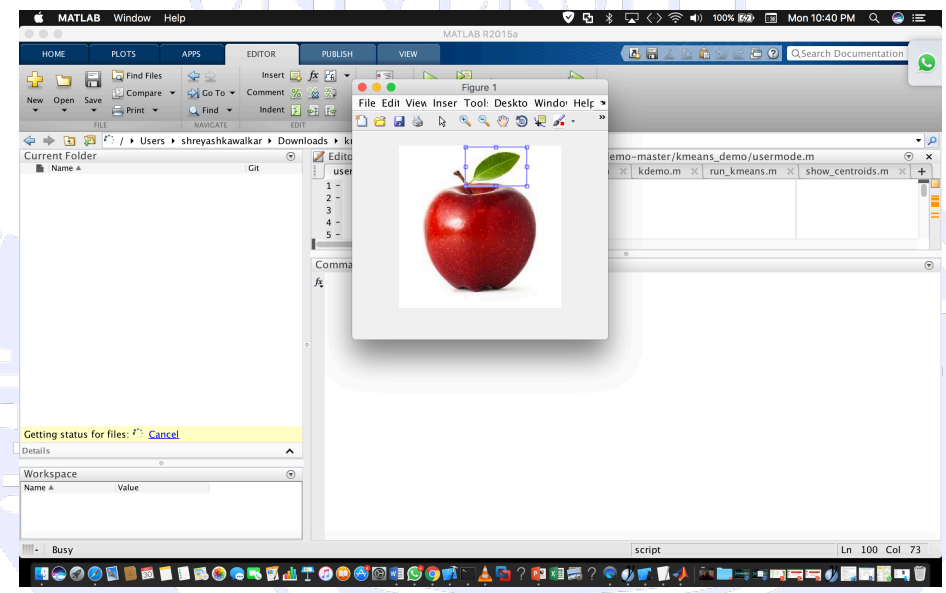
```
clc
clear

rfSize = 6;
numCentroids=320;
whitening=true;
numPatches = 40000;
CIFAR_DIM=[32 32 3];
percentage=[0,0,0,0,0];

%test case
folder = '/Users/shreyashkawalkar/Downloads/kmeans_demo-
master/kmeans_demo/test';
ImageTestFiles = dir([folder '/*.jpg']);
om=imread('test.jpg');
[I2, rect] = imcrop(om);
[x y] =size(I2);
I3=imresize(I2,[32 32]);
imshow(I3);
timage=I3;
timage = timage(:);
timage=timage.';
save('/Users/shreyashkawalkar/Downloads/kmeans_demo-
```

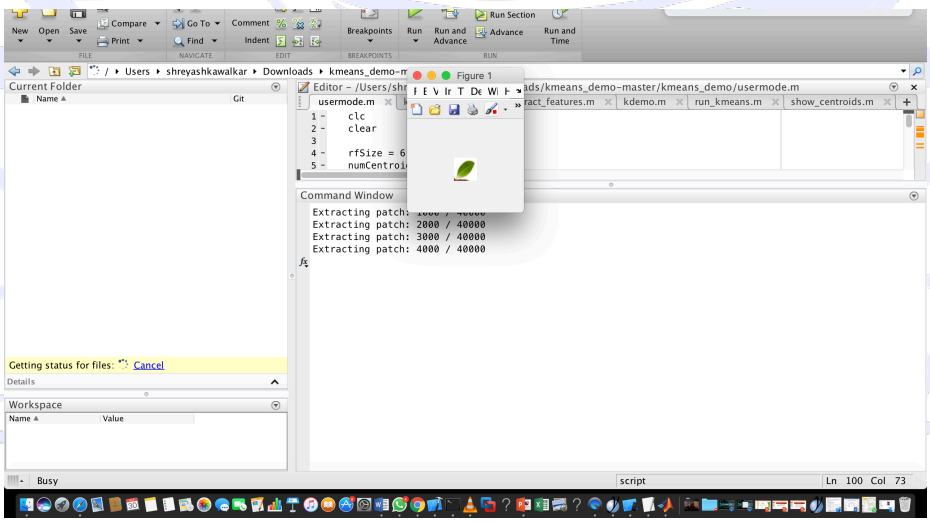
```
master/kmeans_demo/test/test.mat','I3');
testX = double([timage]);

%training data
folder = '/Users/shreyashkawalkar/Downloads/kmeans_demo-
master/kmeans_demo/collection/';
NoOfImageNames = 5;
fname={'/Users/shreyashkawalkar/Downloads/kmeans_demo-
master/kmeans_demo/collection//data_batch_1.mat','/Users/shrey
ashkawalkar/Downloads/kmeans_demo-
master/kmeans_demo/collection//data_batch_2.mat','/Users/shrey
ashkawalkar/Downloads/kmeans_demo-
master/kmeans_demo/collection//data_batch_3.mat','/Users/shrey
ashkawalkar/Downloads/kmeans_demo-
master/kmeans_demo/collection//data_batch_4.mat','/Users/shrey
ashkawalkar/Downloads/kmeans_demo-
master/kmeans_demo/collection//data_batch_5.mat'};
ImageFiles = dir([folder '/*.jpg']); % Get all files.
m=zeros(NoOfImageNames,3072);
for Indexing = 1:NoOfImageNames
    I2=imread('test.jpg');
    %I2=imread(ImageFiles(Indexing).name);
    [x y] =size(I2);
    I3=imresize(I2,[32 32]);
    images=I3;
    new_row = images(:);
    m(Indexing,:) = new_row;
    lbl(Indexing).labels=Indexing;
end
%save(fname(1),'m');
%f11=load(fname(1));
trainX=double([m]);
trainY=double([lbl.labels]);

% extract random patches
patches = zeros(numPatches, rfSize*rfSize*3);
for i=1:numPatches
  if (mod(i,1000) == 0) fprintf('Extracting patch: %d / %d\n',
i, numPatches); end

  r = random('unid', CIFAR_DIM(1) - rfSize + 1);%unique and
randomly r between 1 and 17
  c = random('unid', CIFAR_DIM(2) - rfSize + 1);
  patch = reshape(trainX(mod(i-1,size(trainX,1))+1, :),
CIFAR_DIM);% trainX ith column in 32 32 3 format
  patch = patch(r:r+rfSize-1,c:c+rfSize-1,:);%build patch 32
32 3 to 6 6 3 matrix with r ,c
  patches(i,:) = patch(:)';%at ith column of required no. of
patches, contain 6x6x3 patch values in column
end

% normalize for contrast
```

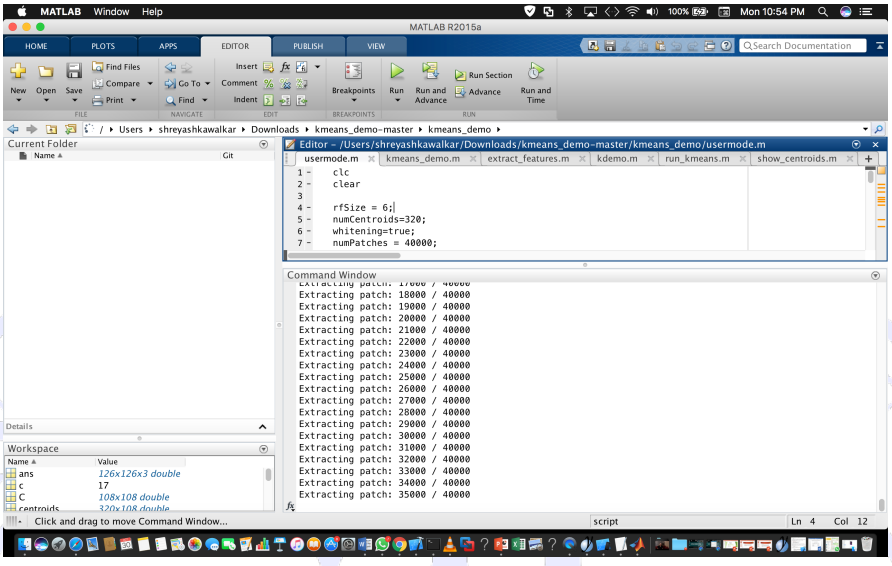
```
patches = bsxfun(@rdivide, bsxfun(@minus, patches,
mean(patches,2)), sqrt(var(patches,[],2)+10));

% whiten
if (whitening)
  C = cov(patches);
  M = mean(patches);
  [V,D] = eig(C);
  P = V * diag(sqrt(1./(diag(D) + 0.1))) * V';
  patches = bsxfun(@minus, patches, M) * P;
end

% run K-means
centroids = run_kmeans(patches, numCentroids, 50);
show_centroids(centroids, rfSize); drawnow;

% extract training features
if (whitening)
  trainXC = extract_features(trainX, centroids, rfSize,
CIFAR_DIM, M,P);
else
  trainXC = extract_features(trainX, centroids, rfSize,
CIFAR_DIM);
end

% standardize data
trainXC_mean = mean(trainXC);
trainXC_sd = sqrt(var(trainXC)+0.01);
trainXCs = bsxfun(@rdivide, bsxfun(@minus, trainXC,
trainXC_mean), trainXC_sd);
trainXCs = [trainXCs, ones(size(trainXCs,1),1)];

%test data
% compute testing features and standardize
if (whitening)
  testXC = extract_features(testX, centroids, rfSize,
CIFAR_DIM, M,P);
else
  testXC = extract_features(testX, centroids, rfSize,
CIFAR_DIM);
end
%stand
testXCs = bsxfun(@rdivide, bsxfun(@minus, testXC,
trainXC_mean), trainXC_sd);
testXCs = [testXCs, ones(size(testXCs,1),1)];

tht = knnclassify(abs(testXCs),abs(trainXCs), trainY, 1);
percentage=1-((sum(tht)-sum(trainY))/length(tht));
fprintf('Train accuracy %f%%\n', 1-((sum(tht)-
sum(trainY))/length(tht)));
%fprintf('Train accuracy %f%%\n', 100 * ((abs(sum(tht -
trainY)) / length(tht))));
```

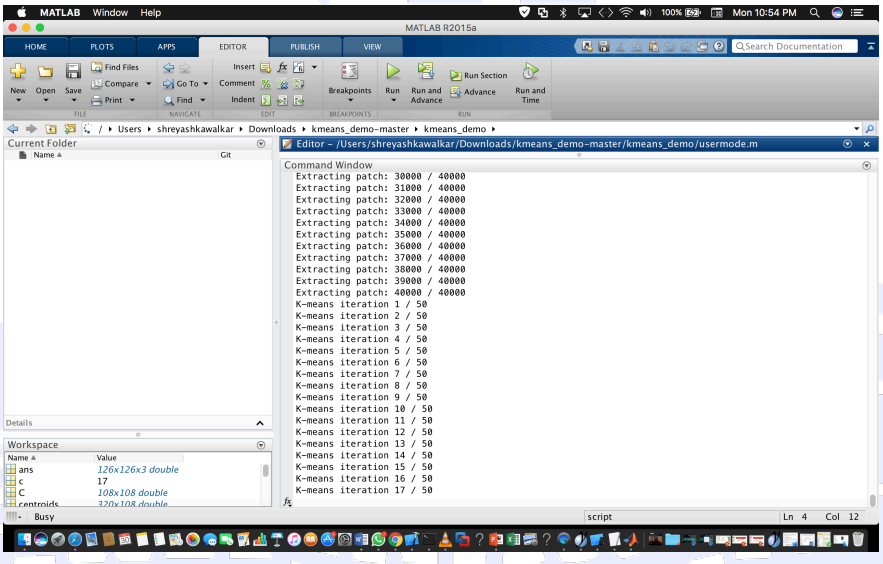
SCREENSHOTS:

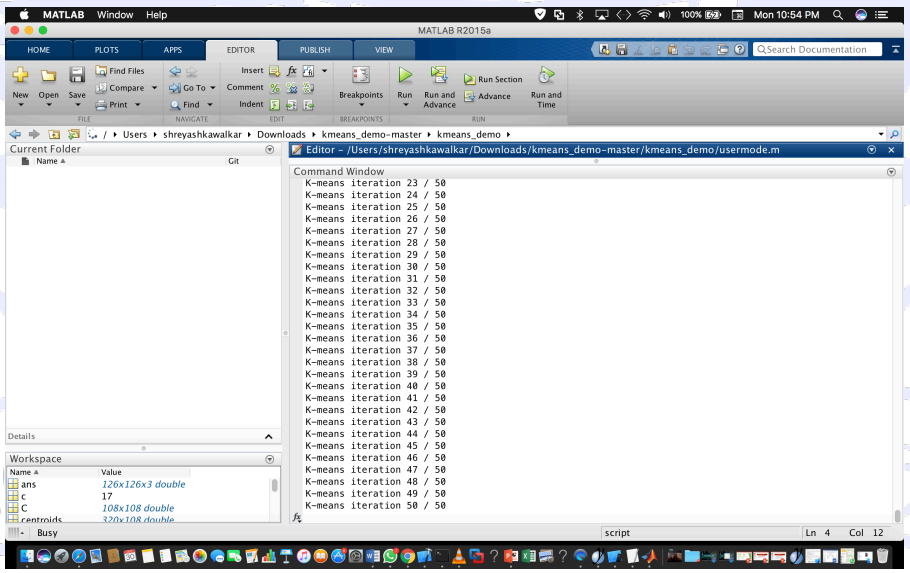


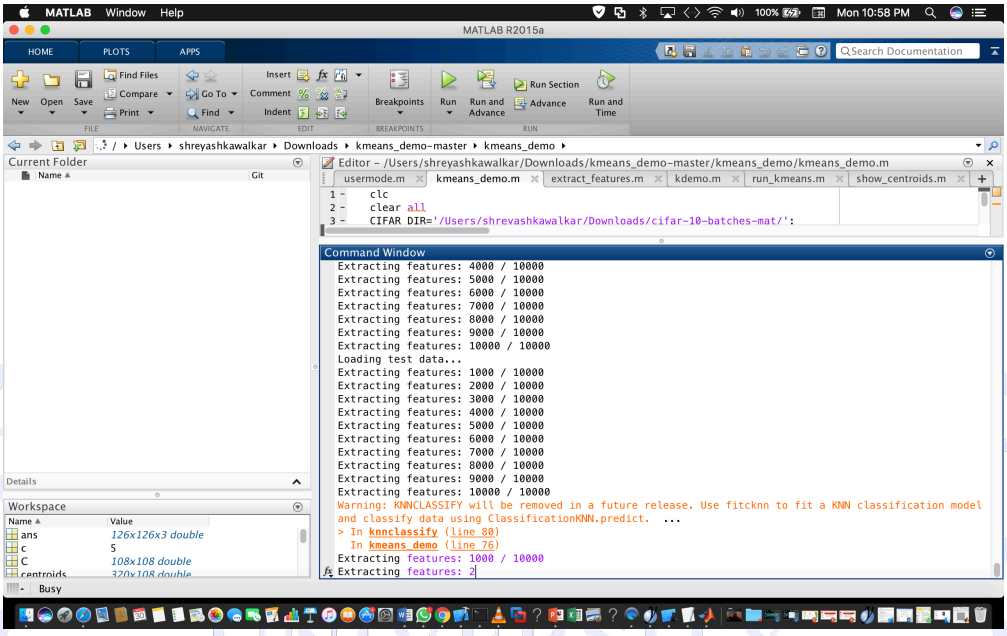


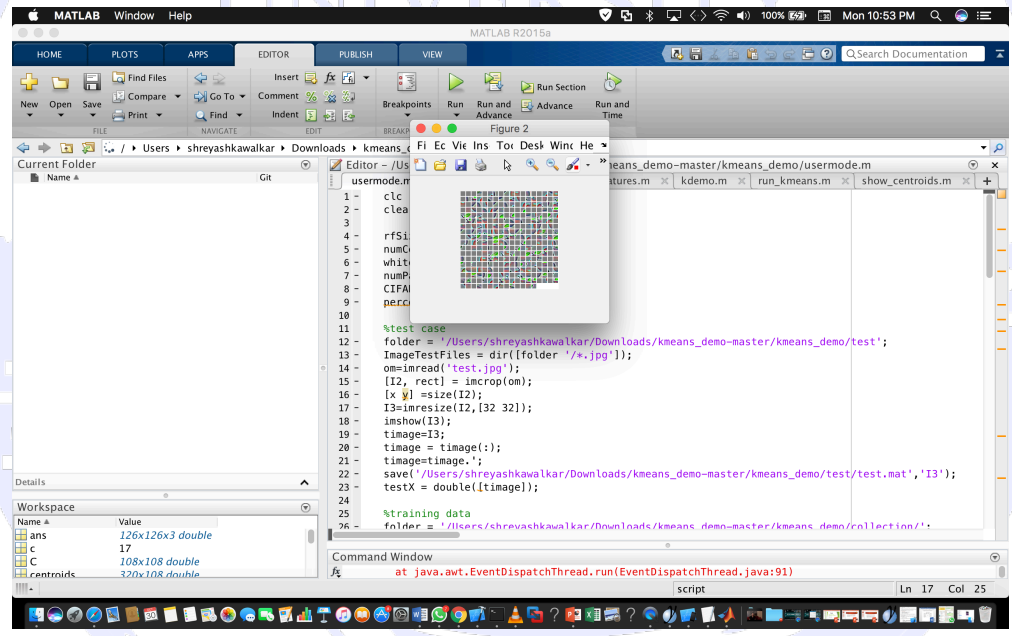



MATLAB Window Help
MATLAB R2015a
HOME PLOTS APPS EDITOR PUBLISH VIEW
Search Documentation
New Open Save Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time
/ Users shreyashkawalkar Downloads kmeans_demo-master kmeans_demo
Current Folder
Name Git
Editor - /Users/shreyashkawalkar/Downloads/kmeans_demo-master/kmeans_demo/usermode.m
usermode.m kmeans_demo.m extract_features.m kdemo.m run_kmeans.m show_centroids.m
1 clc
2 clear
3
4 rfSize = 6;
5 numCentroids=320;
6 whitening=true;
7 numPatches = 40000;
Command Window
Extracting patch: 18000 / 40000
Extracting patch: 19000 / 40000
Extracting patch: 20000 / 40000
Extracting patch: 21000 / 40000
Extracting patch: 22000 / 40000
Extracting patch: 23000 / 40000
Extracting patch: 24000 / 40000
Extracting patch: 25000 / 40000
Extracting patch: 26000 / 40000
Extracting patch: 27000 / 40000
Extracting patch: 28000 / 40000
Extracting patch: 29000 / 40000
Extracting patch: 30000 / 40000
Extracting patch: 31000 / 40000
Extracting patch: 32000 / 40000
Extracting patch: 33000 / 40000
Extracting patch: 34000 / 40000
Extracting patch: 35000 / 40000
Details
Workspace
Name Value
ans 126x126x3 double
c 17
C 108x108 double
centroids 320x108 double
Click and drag to move Command Window...
script Ln 4 Col 12


MATLAB Window Help
MATLAB R2015a
HOME PLOTS APPS EDITOR PUBLISH VIEW
Search Documentation
/ Users shreyashkawalkar Downloads kmeans_demo-master kmeans_demo
Current Folder
Name Git
Editor - /Users/shreyashkawalkar/Downloads/kmeans_demo-master/kmeans_demo/usermode.m
Command Window
Extracting patch: 30000 / 40000
Extracting patch: 31000 / 40000
Extracting patch: 32000 / 40000
Extracting patch: 33000 / 40000
Extracting patch: 34000 / 40000
Extracting patch: 35000 / 40000
Extracting patch: 36000 / 40000
Extracting patch: 37000 / 40000
Extracting patch: 38000 / 40000
Extracting patch: 39000 / 40000
Extracting patch: 40000 / 40000
K-means iteration 1 / 50
K-means iteration 2 / 50
K-means iteration 3 / 50
K-means iteration 4 / 50
K-means iteration 5 / 50
K-means iteration 6 / 50
K-means iteration 7 / 50
K-means iteration 8 / 50
K-means iteration 9 / 50
K-means iteration 10 / 50
K-means iteration 11 / 50
K-means iteration 12 / 50
K-means iteration 13 / 50
K-means iteration 14 / 50
K-means iteration 15 / 50
K-means iteration 16 / 50
K-means iteration 17 / 50
Details
Workspace
Name Value
ans 126x126x3 double
c 17
C 108x108 double
centroids 320x108 double
Busy
script Ln 4 Col 12


MATLAB Window Help
MATLAB R2015a
HOME PLOTS APPS EDITOR PUBLISH VIEW
Search Documentation
/ Users shreyashkawalkar Downloads kmeans_demo-master kmeans_demo
Current Folder
Name Git
Editor - /Users/shreyashkawalkar/Downloads/kmeans_demo-master/kmeans_demo/usermode.m
Command Window
K-means iteration 23 / 50
K-means iteration 24 / 50
K-means iteration 25 / 50
K-means iteration 26 / 50
K-means iteration 27 / 50
K-means iteration 28 / 50
K-means iteration 29 / 50
K-means iteration 30 / 50
K-means iteration 31 / 50
K-means iteration 32 / 50
K-means iteration 33 / 50
K-means iteration 34 / 50
K-means iteration 35 / 50
K-means iteration 36 / 50
K-means iteration 37 / 50
K-means iteration 38 / 50
K-means iteration 39 / 50
K-means iteration 40 / 50
K-means iteration 41 / 50
K-means iteration 42 / 50
K-means iteration 43 / 50
K-means iteration 44 / 50
K-means iteration 45 / 50
K-means iteration 46 / 50
K-means iteration 47 / 50
K-means iteration 48 / 50
K-means iteration 49 / 50
K-means iteration 50 / 50
Details
Workspace
Name Value
ans 126x126x3 double
c 17
C 108x108 double
centroids 320x108 double
Busy
script Ln 4 Col 12


MATLAB Window Help
MATLAB R2015a
HOME PLOTS APPS
Current Folder
Editor - /Users/shreyashkawalkar/Downloads/kmeans_demo-master/kmeans_demo/kmeans_demo.m
usermode.m kmeans_demo.m extract_features.m kdemo.m run_kmeans.m show_centroids.m
clc
clear all
CIFAR_DIR='/Users/shreyashkawalkar/Downloads/cifar-10-batches-mat/';
Command Window
Extracting features: 4000 / 10000
Extracting features: 5000 / 10000
Extracting features: 6000 / 10000
Extracting features: 7000 / 10000
Extracting features: 8000 / 10000
Extracting features: 9000 / 10000
Extracting features: 10000 / 10000
Loading test data...
Extracting features: 1000 / 10000
Extracting features: 2000 / 10000
Extracting features: 3000 / 10000
Extracting features: 4000 / 10000
Extracting features: 5000 / 10000
Extracting features: 6000 / 10000
Extracting features: 7000 / 10000
Extracting features: 8000 / 10000
Extracting features: 9000 / 10000
Extracting features: 10000 / 10000
Warning: KNNCLASSIFY will be removed in a future release. Use fitcknn to fit a KNN classification model and classify data using ClassificationKNN.predict. ...
> In knnclassify (line 80)
In kmeans_demo (line 76)
Extracting features: 1000 / 10000
Extracting features: 2
Workspace
ans 126x126x3 double
c 5
C 108x108 double
centroids 320x108 double
Busy


MATLAB Window Help
MATLAB R2015a
HOME PLOTS APPS EDITOR PUBLISH VIEW
Figure 2
Current Folder
usermode.m
%test case
folder = '/Users/shreyashkawalkar/Downloads/kmeans_demo-master/kmeans_demo/test';
ImageTestFiles = dir([folder '/*.jpg']);
om=imread('test.jpg');
[I2, rect] = imcrop(om);
[x y] =size(I2);
I3=imresize(I2,[32 32]);
imshow(I3);
timage=I3;
timage = timage(:);
timage=timage.';
save('/Users/shreyashkawalkar/Downloads/kmeans_demo-master/kmeans_demo/test/test.mat','I3');
testX = double([timage]);
%training data
Command Window
at java.awt.EventDispatchThread.run(EventDispatchThread.java:91)
Workspace
ans 126x126x3 double
c 17
C 108x108 double
centroids 320x108 double
script Ln 17 Col 25