Eugen Feng
EECE5644
Final Project

1. Data Preparation for Image Processing

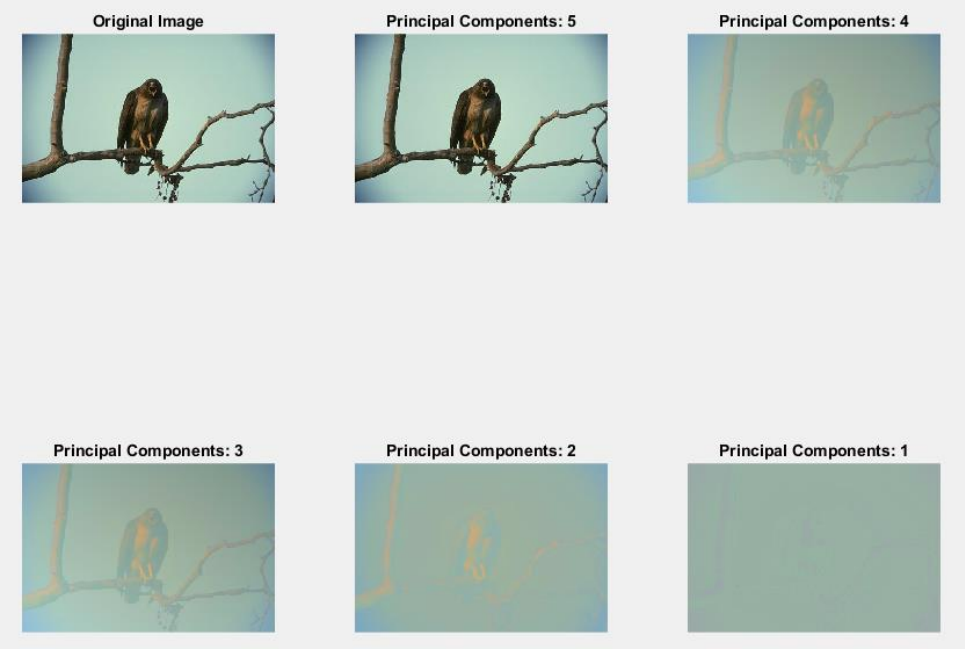

Figure 1 Original image and the number of PCs in the image.

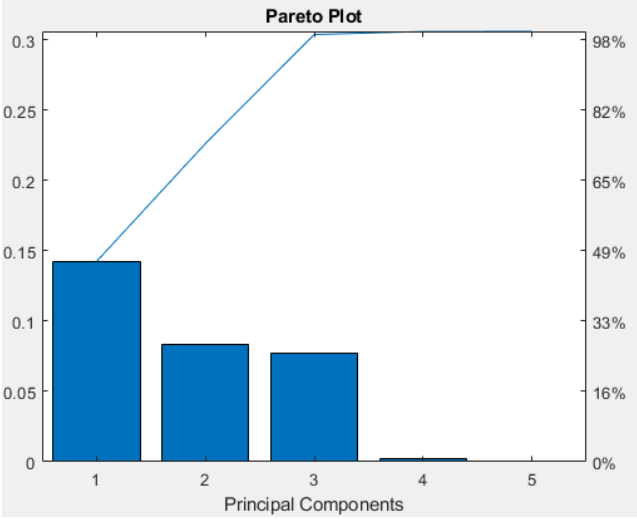

Figure 2 Pareto Plot of PCs

In figures 1, we have the original image next to an image that is approximated with all 5 principal components (PCs). As we go to the next image, we remove 1 PC and we can still see that the image is still within reason. Then as we get to an image with only 2 PCs, we can see that it is difficult to see certain details from the original image. We can also see that in pareto plot, the first 3 PCs are important in producing a reasonable image so therefore, **in my opinion, the minimum principal components we need to construct a reasonable image is 3 PCs.**

2. Unsupervised Clustering on Colorbird Image with Mean Shift

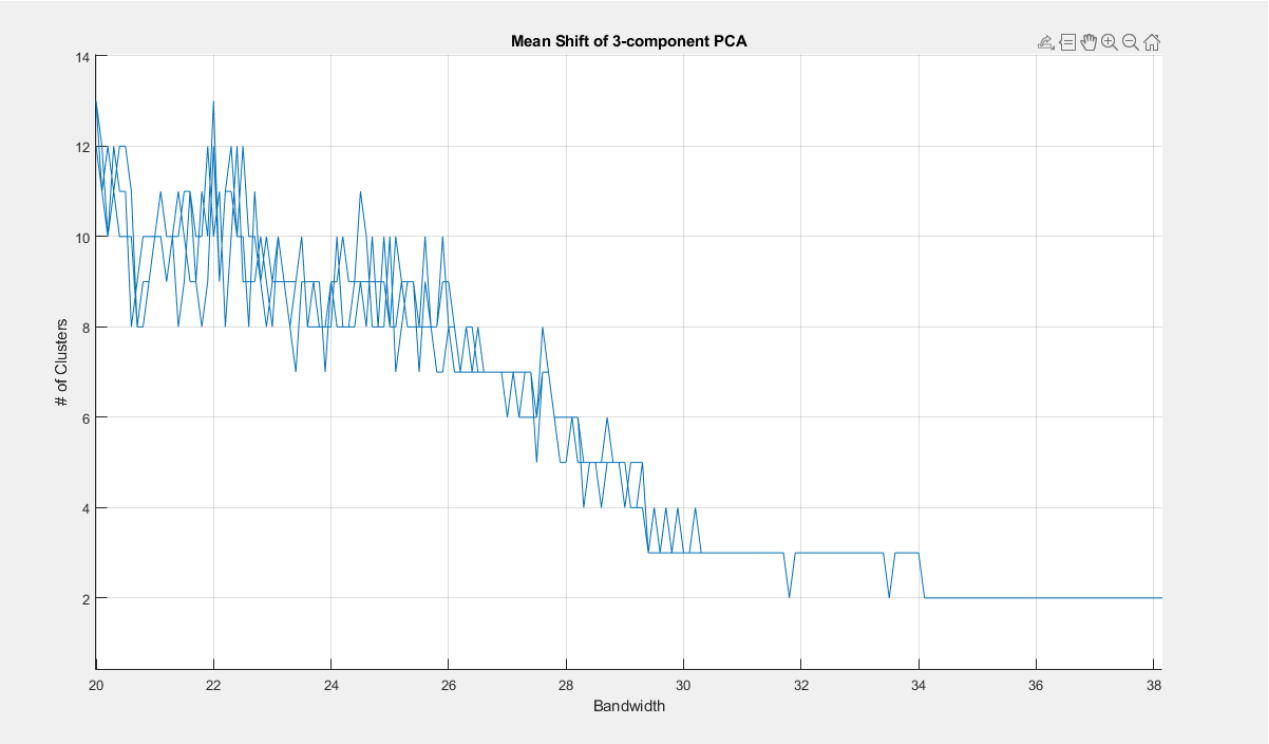

Figure 3 Mean Shift Algorithm applied on 3-component PCA, with 3 random set of vectors.

Figure 3 is a plot of bandwidth vs number of clusters for various random selection of vectors. The reproducibility according to the figure is similar. At a bandwidth greater than 30, we can see that the result of all the random selection of vectors end up with **2 clusters to accurately approximate the data**.

3. Colorbird Image Segmentation with Unsupervised Clustering: KNN

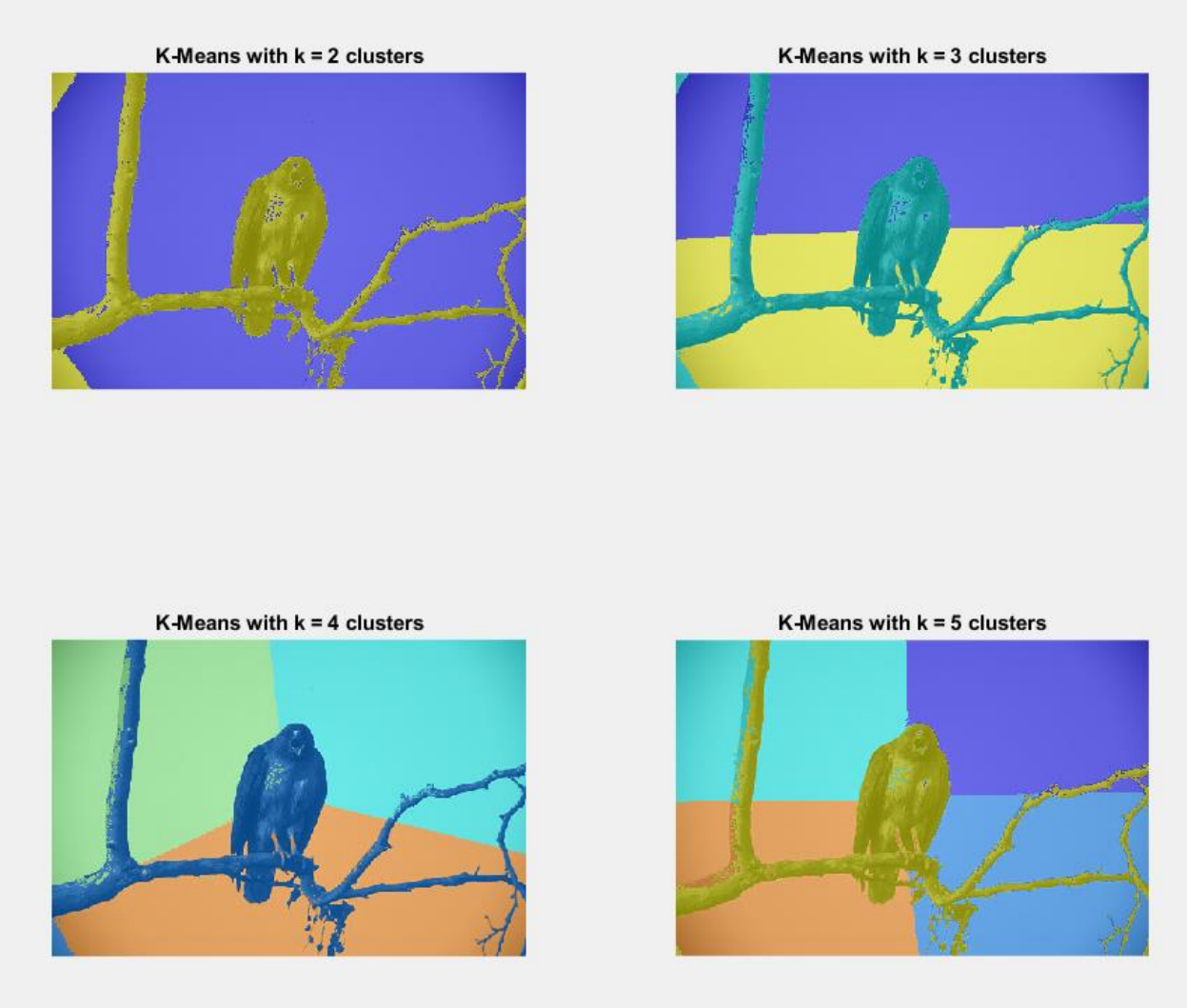

Figure 4 K-Means Algorithms with minimum Euclidean-distance-based assignment of samples to cluster centroids.

4. Colorbird Image Segmentation with Unsupervised Clustering: GMM

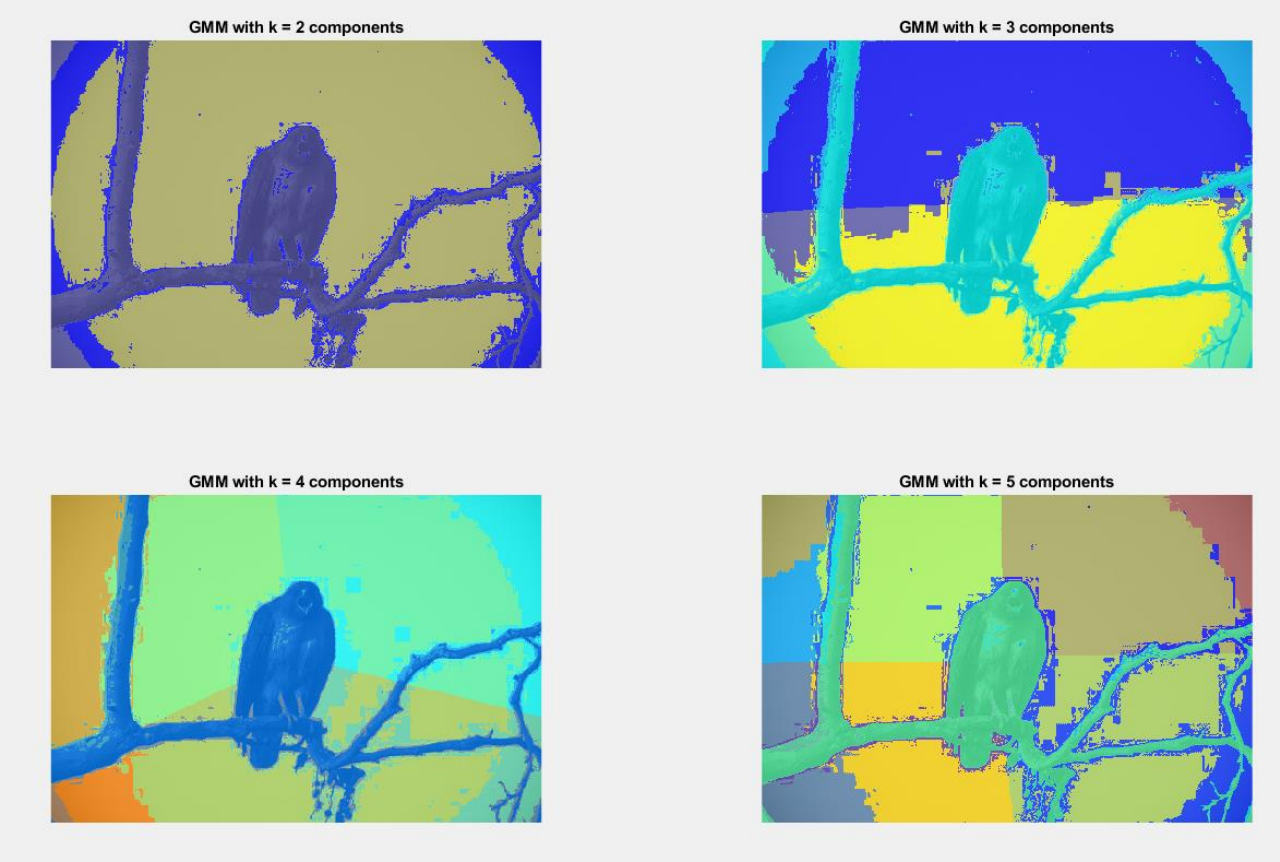

Figure 5 Gaussian Mixture Model-based clustering with EM algorithm fitting

In figures 4, we see that K-Means does hard assignment and detects spherical cluster. The Gaussian Mixture Model clustering algorithm does a soft assignment, and each pixel is expressed as a weighted sum of gaussians. K-means also calculate distance, while GM calculates “weighted” distance.

```
clear,clc,close all

%% Initialization
data=imread('42049_colorBird.jpg');
figure(1),subplot(2,3,1),imshow(data),title('Original Image');
nc=     size(data,2);
nr=     size(data,1);
featureSize =   nc*nr;
randSampFrac = 0.05;

raw_feature=zeros(featureSize,5);
l_rc=0;
for colcount=1:nc
    for rowcount=1:nr
        l_rc=l_rc+1;
        raw_feature(l_rc,:)=[rowcount colcount
double(data(rowcount,colcount,1)) double(data(rowcount,colcount,2))
double(data(rowcount,colcount,3))];
    end
end

%% PCA Algorithm Problem 1
disp('-----------------------------------------------------------------')
disp('Starting PCA Algorithm...')
tic
for pc = 5:-1:1
    [feature,mu,sigma]=zscore(raw_feature);
    x_max = max(feature);
    x_min = min(feature);
    x_range = x_max - x_min;
    norm_feat = (feature - x_min)./x_range;

    [coeff,score,latent,tsquared,explained,mu2]= pca(norm_feat);

    recon = (score(:,6-pc:5) * coeff(:,6-pc:5)') + mu2;
    tester = recon .* x_range + x_min;
    tester = tester .* sigma + mu;
    Ipc = zeros(nr,nc,3,'uint8');
    Ipc1r = reshape(tester(:,3),nr,nc);
    Ipc1g = reshape(tester(:,4),nr,nc);
    Ipc1b = reshape(tester(:,5),nr,nc);
    for j = 1:nc
        for i = 1:nr
            Ipc(i,j,:)=[Ipc1r(i,j,:) Ipc1g(i,j,:) Ipc1b(i,j,:)];
        end
    end

    figure(1)
    subplot(2,3,7-pc)
    imshow(Ipc)
    title(['Principal Components: ', num2str(pc)])
end
```

```
figure(2)
pareto(latent,1)
title('Pareto Plot')
xlabel('Principal Components')
toc

%% Mean Shift Algorithm Problem 2
disp('-----------------------------------------------------------------')
disp('Starting Mean Shift Algorithm...')
recon = (score(:,3:5) * coeff(:,3:5)') + mu2;
recon = recon .* x_range + x_min;
recon = recon .* sigma + mu;
x = recon';
numSamples = round(featureSize*randSampFrac);
figure(3);
hold on,grid on;
xlabel('Bandwidth');
ylabel('# of Clusters');
title('Mean Shift of 3-component PCA');
ylim([0 15]);
xlim([20 35]);
disp('This will take a while. Please be patient.')
tic
for iter = 1:3
    x_idx = randperm(length(x),numSamples);
    x = x(:,x_idx);
    bw = [20:0.1:40]';
    MS = [bw,zeros(length(bw),1)];
    for b = 1:length(bw)
        bandwidth = bw(b);
        [clustCent,point2cluster,clustMembsCell] =
MeanShiftCluster(x,bandwidth);
        numClust = length(clustMembsCell);
        MS(b,2)=numClust;
    end
    line(MS(:,1),MS(:,2));
end
hold off
toc
%% K-Means Algorithms Problem 3
disp('-----------------------------------------------------------------')
disp('Starting K-Means Algorithms...')
disp('Starting GMM  Algorithms...')
x = [0:nr-1];
chan_4 = repmat(x,nc,1)';
feature_ = cat(3,double(data),chan_4);
y = [0:nc-1];
chan_5 = repmat(y,nr,1);
feature_ = cat(3,feature_,chan_5);
feature_ = double(feature_);
contrast = rgb2gray(data);
tic
for channel = 1:size(feature_,3)
    x_max = max(max(feature_(:,:,channel)));
    x_min = min(min(feature_(:,:,channel)));
```

```
    x_range = x_max - x_min;
    feature_(:,:,channel) = (feature_(:,:,channel)-x_min)/x_range;
end
new_ = reshape(feature_,[featureSize,5]);

for clusters = 2:5
%K-Means Clustering
    [l_rc,C] = kmeans(new_,clusters);
    l_rc = reshape(l_rc,[nr,nc]);
    B = labeloverlay(contrast,l_rc);
    figure(4);
    subplot(2,2,clusters-1);
    imshow(B);
    title(['K-Means with k = ', num2str(clusters), ' clusters']);
%% Guassian Mixture Model Clustering Problem 4
    GMM = fitgmdist(new_,clusters);
    idx = cluster(GMM,new_);
    idx = reshape(idx,[nr,nc]);
    gmm = labeloverlay(B,idx);
    figure(5);
    subplot(2,2,clusters-1);
    imshow(gmm);
    title(['GMM with k = ', num2str(clusters), ' components'])
end
toc
%% Functions below
 function [clustCent,data2cluster,cluster2dataCell] =
MeanShiftCluster(dataPts,bandWidth,plotFlag);
if nargin < 2
    error('no bandwidth specified')
end
if nargin < 3
    plotFlag = true;
    plotFlag = false;
end
[numDim,numPts] = size(dataPts);
numClust        = 0;
bandSq          = bandWidth^2;
initPtInds      = 1:numPts;
maxPos          = max(dataPts,[],2);
minPos          = min(dataPts,[],2);
boundBox        = maxPos-minPos;
sizeSpace       = norm(boundBox);
stopThresh      = 1e-3*bandWidth;
clustCent       = [];
beenVisitedFlag = zeros(1,numPts,'uint8');
numInitPts      = numPts;
clusterVotes    = zeros(1,numPts,'uint16');
while numInitPts
    tempInd         = ceil( (numInitPts-1e-6)*rand);
    stInd           = initPtInds(tempInd);
    myMean          = dataPts(:,stInd);
    myMembers       = [];
    thisClusterVotes    = zeros(1,numPts,'uint16');
    while 1
```

```
        sqDistToAll = sum((repmat(myMean,1,numPts) - dataPts).^2);
        inInds      = find(sqDistToAll < bandSq);
        thisClusterVotes(inInds) = thisClusterVotes(inInds)+1;
        myOldMean   = myMean;
        myMean      = mean(dataPts(:,inInds),2);
        myMembers   = [myMembers inInds];
        beenVisitedFlag(myMembers) = 1;
        if plotFlag
            figure(12345),clf,hold on
            if numDim == 2
                plot(dataPts(1,:),dataPts(2,:),'.')
                plot(dataPts(1,myMembers),dataPts(2,myMembers),'ys')
                plot(myMean(1),myMean(2),'go')
                plot(myOldMean(1),myOldMean(2),'rd')
                pause
            end
        end
        if norm(myMean-myOldMean) < stopThresh

            mergeWith = 0;
            for cN = 1:numClust
                distToOther = norm(myMean-clustCent(:,cN));
                if distToOther < bandWidth/2
                    mergeWith = cN;
                    break;
                end
            end
            if mergeWith > 0
                clustCent(:,mergeWith)       =
0.5*(myMean+clustCent(:,mergeWith));
                clusterVotes(mergeWith,:)    = clusterVotes(mergeWith,:) +
thisClusterVotes;
            else
                numClust                     = numClust+1;
                clustCent(:,numClust)        = myMean;
                clusterVotes(numClust,:)     = thisClusterVotes;
            end
            break;
        end
    end
    initPtInds      = find(beenVisitedFlag == 0);
    numInitPts      = length(initPtInds);
end
[val,data2cluster] = max(clusterVotes,[],1);
if nargout > 2
    cluster2dataCell = cell(numClust,1);
    for cN = 1:numClust
        myMembers = find(data2cluster == cN);
        cluster2dataCell{cN} = myMembers;
    end
end
end
```