Homework 10

Meta information	
Name	Savan Kiran
Program	Masters in Computer Science
Questions skipped	N/A
Questions substituted	N/A
Extra credit questions	N/A

PART A

1. I implemented K-means clustering following the instructions in the lecture slides. The termination of the algorithm is when the value of the objective function goes below the set threshold. Objective function is the sum of the squared errors from the mean of the assigned cluster. In this task, we consider color as the 3D feature vector for segmentation of pixels into clusters. The cluster color would be set to the mean of each pixel that belongs to the cluster. The number of clusters is pre-defined by setting the value of the 'k'. Below are pictures depicting the segmentation based on color in 3 images where number of clusters is first set to 5 and then to 10.

Figure 1. Sunset image with k=5 (i.e., 5 clusters)

Figure 2. Sunset image with k=10 (i.e., 10 clusters)

Figure 3. Tiger-1 image with k=5 (i.e., 5 clusters)

Figure 4. Tiger-1 image with k=10 (i.e., 10 clusters)

Figure 5. Tiger-2 image with k=5 (i.e., 5 clusters)

Figure 6. Tiger-2 image with k=10 (i.e., 10 clusters)

Code snippet is provided at the end of 3). K-means clustering function has configurable parameters lambda and texture which is the reason it is at the end of 3).

2. We now add spatial information to the 3D color feature vector from earlier. The spatial information has a constant scaling factor 'lambda' that controls the clustering based on proximity of the pixel to the mean of the cluster. The spatial information {x,y} is first scaled to match the 0-255 range of color values which in turn are multiplied with 'lambda'. Now, all 5 feature vector values are considered in the objective function while clustering. Below are pictures of Sunrise at varying 'lambda' values. As 'lambda' values increase, we see that pixels that are closer are clustered together with more weight to distance than color which might be useful in some images and might work adversely in others. The correct 'lambda' value would depend entirely on the image we're working.

Figure 7. Sunset image with k=5 and lamda=1

Figure 8. Sunset image with k=5 and lamda=2

Figure 9. Sunset image with k=5 and lamda=6

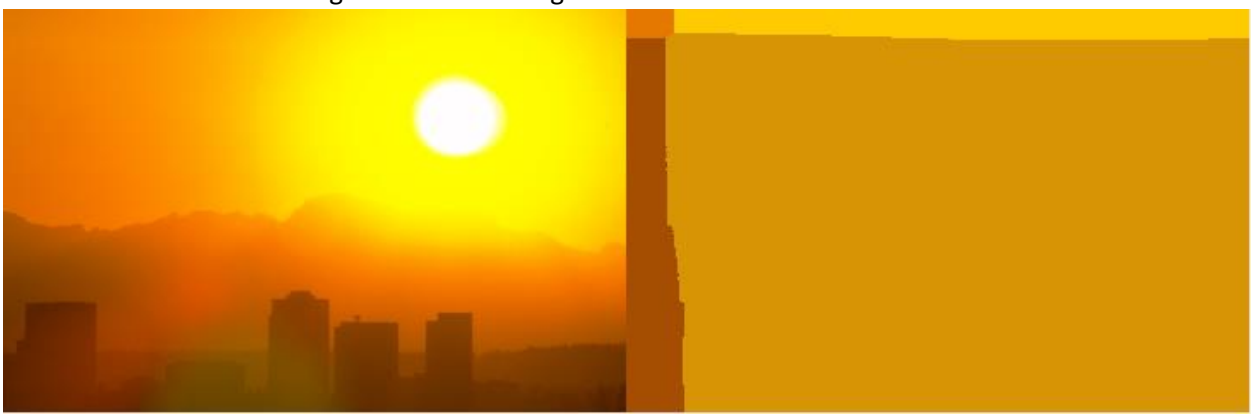

Figure 10. Sunset image with k=5 and lamda=10

3. We construct some texture features from the black & white versions of the image. To find texture features, we use the ability to find dots, horizontal and vertical edges. To find dots, we use the gaussian filter to smoothen the image. To find horizontal & vertical edges, we use the convolution with X- and Y-derivatives. We compute the different filters at varying sigma values. To aggregate the filter responses, we consider a window, W, around the pixel and compute the mean squared response of the filters at different sigmas. This will result in the feature vector with texture at each pixel consider a window, W.
    a. We use this texture feature vector alone and Figure 11 (a,b,c) shows the result.
    b. We use the texture along with RGB color and Figure 12 shows the result.
    c. We use the texture along with RGB color and spatial info and Figure 13 shows the result.

Figure 11 a. Sunset image with Texture feature alone. Window size=3

Figure 11 b. Sunset image with Texture feature alone. Window size=7

Figure 11 c. Sunset image with Texture feature alone. Window size=11

Figure 12. Sunset image with texture and color features.

Figure 13. Sunset image with texture, color and spatial features.

The resulting images show evidently that the clustering now also depends on texture (i.e., edge information) apart from color and spatial information. Seems like the resulting image has one cluster dedicated for edges (textures). The other clusters seems to be dominated by color as seen in Figure 12 (and spatial information in Figure 13).

Code snippet for K-means clustering with configurable k, lambda and texture feature.

```
function kmeans(file_path, k, thres, lambda, texture_enabled, T)
image=imread(file_path);
[row,col,channel]=size(image);

c=zeros(k,2);
c_color=zeros(k,6);
for i=1:k
    c(i,1)=randi([1,row]);
    c(i,2)=randi([1,col]);
    color=image(c(i,1),c(i,2),:);
    c_color(i,1)=color(1,1,1);
    c_color(i,2)=color(1,1,2);
    c_color(i,3)=color(1,1,3);
    x=(c(i,1)-1)*255/(row-1);
    y=(c(i,2)-1)*255/(col-1);
    c_color(i,4)=x*lambda;
```

```
    c_color(i,5)=y*lambda;
    if texture_enabled==1
        c_color(i,6)=T(c(i,1), c(i,2));
    end
end

condition=1;
while condition==1
    imageS=zeros(row,col);
    for i=1:row
        for j=1:col
            pixel_color=image(i,j,:);
            x=(i-1)*255/(row-1);
            y=(j-1)*255/(col-1);
            if texture_enabled==1
                pixel_color=[pixel_color(1,1,1) pixel_color(1,1,2)
pixel_color(1,1,3) double(x*lambda) double(y*lambda) T(i,j)];
            else
                pixel_color=[pixel_color(1,1,1) pixel_color(1,1,2)
pixel_color(1,1,3) double(x*lambda) double(y*lambda) 0];
            end
            imageS(i,j)=which_cluster(c_color, k, pixel_color);
        end
    end

    c_color_new=zeros(k,6);
    c_color_new=double(c_color_new);
    c_count=zeros(k,1);
    for i=1:row
        for j=1:col
            x=(i-1)*255/(row-1);
            y=(j-1)*255/(col-1);

c_color_new(imageS(i,j),1)=c_color_new(imageS(i,j),1)+double(image(i,j,1));

c_color_new(imageS(i,j),2)=c_color_new(imageS(i,j),2)+double(image(i,j,2));

c_color_new(imageS(i,j),3)=c_color_new(imageS(i,j),3)+double(image(i,j,3));

c_color_new(imageS(i,j),4)=c_color_new(imageS(i,j),4)+double(x*lambda);

c_color_new(imageS(i,j),5)=c_color_new(imageS(i,j),5)+double(y*lambda);
            if texture_enabled==1

c_color_new(imageS(i,j),6)=c_color_new(imageS(i,j),6)+double(T(i,j));
            else

c_color_new(imageS(i,j),6)=c_color_new(imageS(i,j),6)+double(0);
            end
            c_count(imageS(i,j),1)=c_count(imageS(i,j),1)+1;
        end
    end

    for i=1:k
        c_color_new(i,1)=c_color_new(i,1)/c_count(i,1);
        c_color_new(i,2)=c_color_new(i,2)/c_count(i,1);
```

```
        c_color_new(i,3)=c_color_new(i,3)/c_count(i,1);
        c_color_new(i,4)=uint8(c_color_new(i,4)/c_count(i,1));
        c_color_new(i,5)=uint8(c_color_new(i,5)/c_count(i,1));
        c_color_new(i,6)=uint8(c_color_new(i,6)/c_count(i,1));
    end
    c_color_new=uint8(c_color_new);

    c_obj=zeros(k,1);
    for i=1:k
        c_obj(i,1)=sqrt(objective_func(c_color(i,:),c_color_new(i,:)));
    end

    c_color=c_color_new;
    condition=0;
    for i=1:k
        if c_obj(i,1) > thres
            condition=1;
        end
    end
end

imageSeg=zeros(row,col,3);
for i=1:row
    for j=1:col
        imageSeg(i,j,1)=c_color(imageS(i,j),1);
        imageSeg(i,j,2)=c_color(imageS(i,j),2);
        imageSeg(i,j,3)=c_color(imageS(i,j),3);
    end
end
imageSeg=uint8(imageSeg);

I=zeros(row,col*2,3);
for i=1:row
    for j=1:col
        I(i,j,1)=image(i,j,1);
        I(i,j,2)=image(i,j,2);
        I(i,j,3)=image(i,j,3);
    end
end

for i=1:row
    for j=1:col
        I(i,col+j,1)=imageSeg(i,j,1);
        I(i,col+j,2)=imageSeg(i,j,2);
        I(i,col+j,3)=imageSeg(i,j,3);
    end
end

I=uint8(I);
figure;
imshow(I);
title([file_path ' - k=' num2str(k) ', lambda=' num2str(lambda) ' and
texture=' num2str(texture_enabled)]);

end
```

Below is the code snippet for finding the texture feature.

```
function [It]=texture_features(file_path, w)
I=imread(file_path);
I=rgb2gray(I);
I=double(I);

dx=[-1 0 1; -1 0 1; -1 0 1];
dy=dx';
I=conv2(conv2(I, dx, 'same'), dy, 'same');

h=3;
sigma=0.1;
g=fspecial('gaussian', h, sigma);
Is1=conv2(I, g, 'same');

sigma=0.3;
g=fspecial('gaussian', h, sigma);
Is2=conv2(I, g, 'same');

sigma=0.5;
g=fspecial('gaussian', h, sigma);
Is3=conv2(I, g, 'same');

sigma=0.7;
g=fspecial('gaussian', h, sigma);
Is4=conv2(I, g, 'same');

It=zeros(size(I,1),size(I,2));
for i=1+w:size(I,1)-w
    for j=1+w:size(I,2)-w
        ws1=Is1(i-w:i+w,j-w:j+w).^2;
        ws2=Is2(i-w:i+w,j-w:j+w).^2;
        ws3=Is3(i-w:i+w,j-w:j+w).^2;
        ws4=Is4(i-w:i+w,j-w:j+w).^2;
        ws=[ws1(:) ws2(:) ws3(:) ws4(:)];
        It(i,j)=mean(ws(:));
    end
end
It=uint8(It);

figure;
imshow(It);
title([file_path ' - w=' num2str(w)]);

end
```

PART B

We start by creating texton representation from a training set. We use the matlab's `gabor` function to create the filter bank. We created filter bank with wavelength=4 and orientation={0 45 90 135 180}. We use the `imgaborfilt` function to apply these filters and find the filter responses are stored in the vector form for each pixel of the image. The dimension of the response vector is equal to the number of different wavelengths times number of different orientations (which in our case is 1x5=5). In the earlier examples, we saw that 5 clusters were able to segment the tiger images well. Based on that and a little experimentation with different values, we arrive at k=6. We now use K-means clustering on each of the filter response vector from all images in the training set and arrive at the clusters (textons).

In the next step, we have a test image that we must classify based on the information we have gathered so far. We calculate the filter responses for each pixel using the same earlier approach (i.e., `gabor` filter) and find the closest matching cluster for each pixel using the objective function described in Part A. Each pixel in the test image now belongs to a cluster. We use a window of size=7 to find the histogram for each cluster at each pixel. Since we have k=6, we now arrive at a feature vector of size 6 at each pixel. We now use this vector for K-means clustering.

Like before, we will use this vector alone, with color and with color & spatial information. Below are pictures depicting the result of each of them.

Figure 14. Test-1 image with texton feature clustering

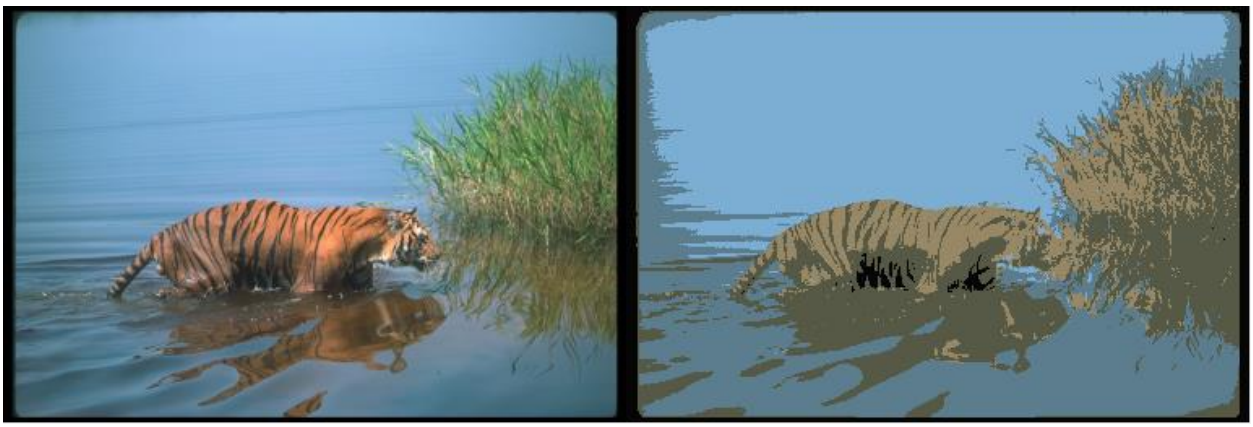

Figure 15. Test-1 image with texton and color feature clustering

Figure 16. Test-1 image with texton, color and spatial feature clustering

Unlike the more naïve form, we can see that texton based texture representation is more sophisticated and robust. It picks up hidden/subtle details much better than the naïve approach.

Below is the code snippet for the same.

```
function texton_texture(train_imgs)
k=6;
thres=0.5;

FR=[];
for i=1:size(train_imgs,1)
    I=imread(train_imgs(i,:));
    I=rgb2gray(I);
    gaborBank=gabor(4,[0 45 90 135 180]);
    I=imgaborfilt(I,gaborBank);

    [row,col,channel]=size(I);
    I=reshape(I,[row*col,5]);
    FR=[FR; I];
end

textons=kmeans_filter(FR, k, thres);

test_path='tigers_small/test_small/108028.tiff';
test=imread(test_path);
[row,col,channel]=size(test);
test=rgb2gray(test);
gaborBank=gabor(4,[0 45 90 135 180]);
test=imgaborfilt(test,gaborBank);

test_seg=zeros(row,col,1);
for i=1:row
    for j=1:col
        filt_resp=test(i,j,:);
        filt_resp=[filt_resp(1,1,1) filt_resp(1,1,2) filt_resp(1,1,3)
filt_resp(1,1,4) filt_resp(1,1,5)];
        test_seg(i,j)=which_cluster_2(textons, k, filt_resp);
    end
end
```

```
w=3;
test_hist=zeros(row,col,k);
for i=1+w:row-w
    for j=1+w:col-w
        texton_count=zeros(k,1);
        for l=-w:w
            for m=-w:w

texton_count(test_seg(i+l,j+m),1)=texton_count(test_seg(i+l,j+m),1)+1;
            end
        end
        for q=1:k
            test_hist(i,j,q)=texton_count(q,1);
        end
    end
end

kmeans_2(test_path, 5, 0, 0, test_hist, 0);
kmeans_2(test_path, 5, 0, 0, test_hist, 1);
kmeans_2(test_path, 5, 0, 1, test_hist, 1);
end

function [c_filt] = kmeans_filter(FR, k, thres)
[row,channel]=size(FR);

c=zeros(k,1);
c_filt=zeros(k,5);
for i=1:k
    c(i,1)=randi([1,row]);
    filt_resp=FR(c(i,1),:);
    c_filt(i,1)=filt_resp(1);
    c_filt(i,2)=filt_resp(2);
    c_filt(i,3)=filt_resp(3);
    c_filt(i,4)=filt_resp(4);
    c_filt(i,5)=filt_resp(5);
end

condition=1;

while condition==1
    imageS=zeros(row,1);
    for i=1:row
        filt_resp=FR(i,:);
        filt_resp=[filt_resp(1) filt_resp(2) filt_resp(3) filt_resp(4)
filt_resp(5)];
        imageS(i,1)=which_cluster_2(c_filt, k, filt_resp);
    end

    c_filt_new=zeros(k,5);
    c_filt_new=double(c_filt_new);
    c_count=zeros(k,1);
    for i=1:row
        c_filt_new(imageS(i,1),1)=c_filt_new(imageS(i,1),1)+double(FR(i,1));
        c_filt_new(imageS(i,1),2)=c_filt_new(imageS(i,1),2)+double(FR(i,2));
```

```
        c_filt_new(imageS(i,1),3)=c_filt_new(imageS(i,1),3)+double(FR(i,3));
        c_filt_new(imageS(i,1),4)=c_filt_new(imageS(i,1),4)+double(FR(i,4));
        c_filt_new(imageS(i,1),5)=c_filt_new(imageS(i,1),5)+double(FR(i,5));
        c_count(imageS(i,1),1)=c_count(imageS(i,1),1)+1;
    end

    for i=1:k
        c_filt_new(i,1)=c_filt_new(i,1)/c_count(i,1);
        c_filt_new(i,2)=c_filt_new(i,2)/c_count(i,1);
        c_filt_new(i,3)=c_filt_new(i,3)/c_count(i,1);
        c_filt_new(i,4)=c_filt_new(i,4)/c_count(i,1);
        c_filt_new(i,5)=c_filt_new(i,5)/c_count(i,1);
    end

    c_obj=zeros(k,1);
    for i=1:k
        c_obj(i,1)=sqrt(objective_func_2(c_filt(i,:),c_filt_new(i,:)));
    end

    c_filt=c_filt_new;
    condition=0;
    for i=1:k
        if c_obj(i,1) > thres
            condition=1;
        end
    end
end

end
```