Final project script 3

Step 3. Perform k-means clustering of features for an unsupervised learning coin-type classifier and visualiz

- Apply the k-means algorithm with k=3 on feature matrix **D** to obtain classification **cls_init**. Make sur different result since k-means uses random numbers in its optimization. Clustering should result in c
- Map the labels assigned by k-means to each class to a numeric label for each coin of 1, 2, and 3 for store the average area of the components in each of the 3 classes in the [3x1] vector **class_ave_ol** stored in **component_size**). The class corresponding to dimes should have the smallest average s labels in **cls_init** to [1,2,3] for smallest to largest coin ordering, find the sorting indices **classmap** ne function. **classmap(1)** contains the label of the smallest (dime) class, **classmap(2)** the middle (nick [Nx1] classification vector **cls** such that **cls**(i) equals 1 if **cls_init**(i) equals **classmap**(1), **cls**(i) equa equals **classmap**(3). This converts the k-means labels in **cls_init** to be [1,2,3] for dimes, nickels, ar the results in a way that we can easily see if they are correct.
- In a separate MATLAB grader problem, you are asked to create function `AddCoinToPlotAndCour` ***finished***. Embed the resulting function in the bottom of the script with the other Helper Functions
- For each classified centroid, use the `AddCoinToPlotAndCount` function to draw a circle on the in and diameter approximately matching that of the coin.
- The function also outputs the value of the coin being plotted. Sum this result across all coins and sto

Script

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
% Define the filter size we will use in step 2:
filtsize = 85;

% Creating test image 'im' by splicing together two built in images.
% Also zero-padding (adding zeros around the border) with half the
% filter size (filtsize) we will use so that the filter could be
% centered on any actual image pixel, including those near the border.
% 'coins.png' contains bright nickels and dimes on a dark background
% 'eight.tif' contains dark quarters on a bright background, so we invert i
% to match 'coins.png'
im1 = imread('coins.png');
[r,c] = size(im1);
im2 = imread('eight.tif');
[r2,c2] = size(im2);
filtsizeh = floor(filtsize/2);
im = zeros(r+r2+filtsize,c+filtsize);
im(filtsizeh+1:filtsizeh+r+r2,filtsizeh+1:filtsizeh+c) = [im1;255-im2(:,1:c
[r,c] = size(im);
imagesc(im);colormap(gray);title('test image');axis equal;

% Initializing assessed/displayed variables as empty so that code is execut
msk=[]; msk_dil=[]; msk_dil_erd=[]; centroid=[]; component_size=[];
```

```matlab

%%%%% 1. Localize the centroid of each coin
% Otsu threshold
msk = OtsuThreshold(im);
figure; imagesc(msk); colormap(gray); title('Otsu'); axis equal;

% Dilate 9x9
msk_dil = imdilate(msk,ones(9,9));
figure; imagesc(msk_dil); colormap(gray); title('Dilated'); axis equal;

% Erode 23x23
msk_dil_erd = imerode(msk_dil,ones(23,23));
figure; imagesc(msk_dil_erd); colormap(gray); title('Eroded'); axis equal;


% Connected components to get centroids of coins:
cc = bwconncomp(msk_dil_erd);
props_struct = regionprops(cc);
centroid = zeros(length(props_struct),2);
component_size = zeros(length(props_struct),1);
for i=1:length(props_struct)
    centroid(i,:) = round(props_struct(i).Centroid);
    component_size(i) = props_struct(i).Area;
end


%%%%% 2. Measure features for each coin using a bank of matching filters
% make matching filters to create features
% Define diameters to use for filters
dimediameter = 31;
quarterdiameter = 51;
nickeldiameter = 41;

% Initialize assessed variable D
D=[]; nickelfilter = []; dimefilter = []; quarterfilter = [];

% Use the MakeCircleMatchingFilter function to create matching filters for
% (This is in a separate Matlab grader problem. Save your work,
%       complete the corresponding grader problem and embed the solution
%       in the helper function list below.)
nickelfilter = MakeCircleMatchingFilter(nickeldiameter,filtsize);
dimefilter = MakeCircleMatchingFilter(dimediameter,filtsize);
quarterfilter = MakeCircleMatchingFilter(quarterdiameter,filtsize);

figure;
subplot(1,3,1); imagesc(dimefilter); colormap(gray); title('dime filter');
subplot(1,3,2); imagesc(nickelfilter); colormap(gray); title('nickel filter
subplot(1,3,3); imagesc(quarterfilter); colormap(gray); title('quarter filt

% Evaluate each of the 3 matching filters on each coin to serve as 3 featur
D = zeros(length(centroid),3);
```

```
for i=1:length(centroid)
    D(i,1) = corr(dimefilter(:),reshape(msk_dil_erd(centroid(i,2)-filtsizeh
        centroid(i,2)+filtsizeh,centroid(i,1)-filtsizeh:centroid(i,1)+filts
    D(i,2) = corr(nickelfilter(:),reshape(msk_dil_erd(centroid(i,2)-filtsiz
        centroid(i,2)+filtsizeh,centroid(i,1)-filtsizeh:centroid(i,1)+filts
    D(i,3) = corr(quarterfilter(:),reshape(msk_dil_erd(centroid(i,2)-filtsi
        centroid(i,2)+filtsizeh,centroid(i,1)-filtsizeh:centroid(i,1)+filts
end

figure;
subplot(1,3,1); imagesc(dimefilter); colormap(gray); title('dime filter');
subplot(1,3,2); imagesc(nickelfilter); colormap(gray); title('nickel filter
subplot(1,3,3); imagesc(quarterfilter); colormap(gray); title('quarter filt

%%%%% 3. Perform k-means clustering of features for unsupervised learning c
rng(0);
cls_init=[]; cls=[]; totcount=[];

N = cc.NumObjects % connected components
[cls_init,C] = kmeans(D, 3);


cls_init
% relabel centroid classes based on average size of the objects in each cla

dist1 = sqrt((centroid(:,1)-C(1,1)).^2 + (centroid(:,2)-C(1,2)).^2);
dist2 = sqrt((centroid(:,1)-C(2,1)).^2 + (centroid(:,2)-C(2,2)).^2);
dist3 = sqrt((centroid(:,1)-C(3,1)).^2 + (centroid(:,2)-C(3,2)).^2);

class_average_object_size = zeros(3,1);
class_average_object_size(1) = mean(component_size((dist1 < dist2) & (dist1
class_average_object_size(2) = mean(component_size((dist2 < dist1) & (dist2
class_average_object_size(3) = mean(component_size((dist3 < dist1) & (dist3

[~,classmap] = sort(class_average_object_size);

cls = zeros(length(centroid),1);
for i = 1 : length(centroid)
    if (cls_init(i) == classmap(1))
        cls(i) = 1;
    elseif (cls_init(i) == classmap(2))
        cls(i) = 2;
    elseif (cls_init(i) == classmap(3))
        cls(i) = 3;
    end
end


cls
% Visualize the result
figure; imagesc(im);colormap(gray);title('test image');hold on;axis equal;
```

```
% plot circles around each coin with different color/diameter unique to eac

totcount = 0;
for i = 1 : length(centroid)
    rng(0);
    [coinvalue,~,~,~] = AddCoinToPlotAndCount(centroid(i,1), centroid(i,2),
    hold on;
    totcount = totcount + coinvalue;
end


title([num2str(totcount),' cents'])


%%%%%%%%%%%%%%%%%%%% Helper Functions %%%%%%%%%%%%%%%%%%%%

function [coinvalue,x_plot,y_plot,col] = AddCoinToPlotAndCount(x,y,cls)
    % Initialize radians for defining x_plot and y_plot using cos and sin f
    rads = 0 : 2*pi/32 : 2*pi;

    % Initialize parameters for radius and color of circle for each type of
    DimeRadius = 22; NickelRadius = 30; QuarterRadius = 40;
    DimeValue = 10; NickelValue = 5; QuarterValue = 25;

    % Use if-elseif statement to define x_plot, y_plot, col
    % When cls is 1, we found a dime
    if (cls == 1)
        coinvalue = DimeValue;
        x_plot = x + DimeRadius*cos(rads);
        y_plot = y + DimeRadius*sin(rads);
        col = 'r';
    % When cls is 2, we found a nickel
    elseif (cls == 2)
        coinvalue = NickelValue;
        x_plot = x + NickelRadius*cos(rads);
        y_plot = y + NickelRadius*sin(rads);
        col = 'g';
    % When cls is 3, we found a quarter
    elseif (cls == 3)
        coinvalue = QuarterValue;
        x_plot = x + QuarterRadius*cos(rads);
        y_plot = y + QuarterRadius*sin(rads);
        col = 'm';
    end

    plot(x_plot,y_plot,col);
end

function filter = MakeCircleMatchingFilter(diameter,filtsize)
filter = zeros(filtsize,filtsize);
radius = diameter/2;
```

```
radius = diameter/2;
c = (filtsize+1)/2;
for i=1:filtsize
    for j=1:filtsize
        if (i-c)*(i-c) + (j-c)*(j-c) <= radius*radius
            filter(i,j) = 1;
        end
    end
end
end

function [msk,thrsh] = OtsuThreshold(im)
hst = imhist(im);
res = otsuthresh(hst);
thrsh = res*255;
msk = im>thrsh;
end

```

▶ Run Script

Assessment: All Tests Passed

Submit

Is cls_init correct?

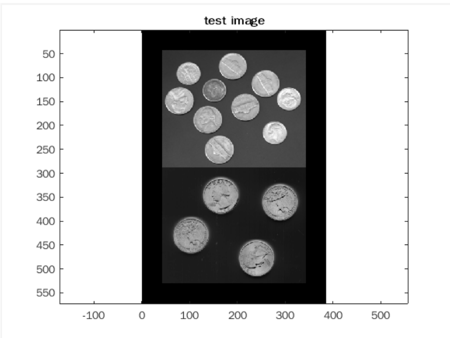
Is cls correct?

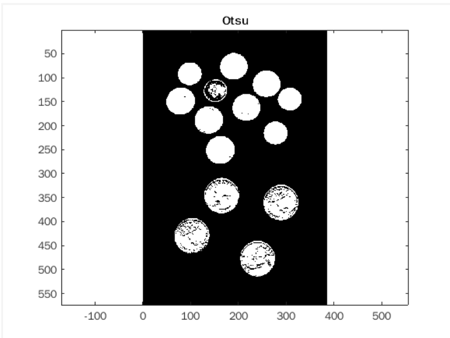
Is totcount correct?

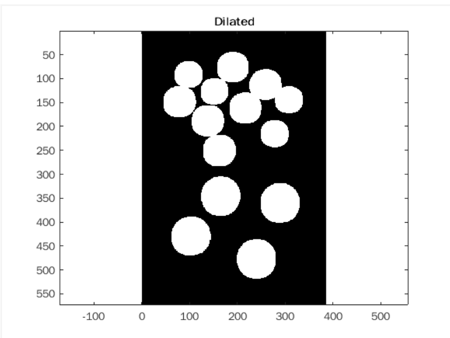
Output

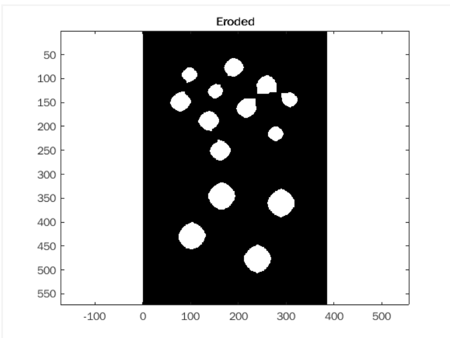
```
N =

    14


cls_init =

     3
     1
     2
     3
     2
```

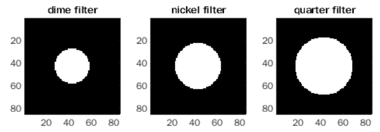
```
     1
     3
     3
     3
     1
     3
     1
     2
     2


cls =

     2
     3
     1
     2
     1
     3
     2
     2
     2
     3
     2
     3
     1
     1
```

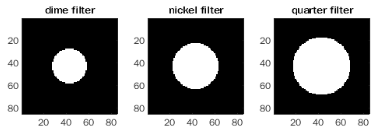


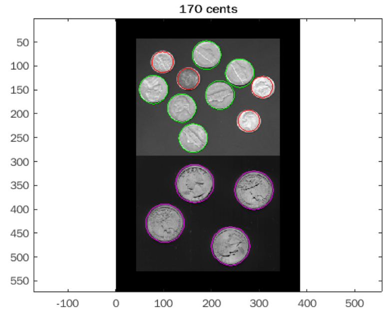







dime filter
nickel filter
quarter filter


dime filter
nickel filter
quarter filter


170 cents