Final project script 2

Step 2. Measure features for each coin using a so-called "filter bank" of matching filters.

In step 1 we found the coins, but we still need to classify them. In step 3 we will perform classifcation. The cl collect in this step. A "filter bank" of matching filters can be constructed, with distinct filters for quarters, nicke coin in **msk_dil_erd**, the results of which generate 3 features that can be used by k-means. in the next step.

- In a separate MATLAB grader problem, you are asked to create function `MakeCircleMatchingFi` ***finished***. Embed the resulting function in the bottom of the script with the other Helper Functions
- Use the `MakeCircleMatchingFilter` function with the default variables provided in the script tc **nickelfilter**, and **quarterfilter**.
- For each *i*th coin centroid found in step 1, compute the *j*th (j=1, 2, and 3 should correspond to the di computing the correlation between the matching filter and the local region of image pixels that fall w centroid. Store the result in **D(i,j).**

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=[];

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

```
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 variables
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.)
N = cc.NumObjects % cc- connected components
[dimefilter, ~, ~] = MakeCircleMatchingFilter(dimediameter, filtsize);
[nickelfilter, ~, ~] = MakeCircleMatchingFilter(nickeldiameter, 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
filtsize
D = zeros(N,3); % Feature Matrix
for i = 1 : N
    local_region = msk_dil_erd(centroid(i,2)-filtsizeh:centroid(i,2)+filtsi
    D(i,1) = corr(dimefilter(:), local_region(:));
```

```
    D(i,1) = corr(dimefilter(:), local_region(:));
    D(i,2) = corr(nickelfilter(:), local_region(:));
    D(i,3) = corr(quarterfilter(:), local_region(:));
end

D

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

function [filter,xc,yc] = MakeCircleMatchingFilter(diameter,W)
% Initialize filter
    filter = zeros(W,W);
% Define coordinates for the center of the W x W filter
    xc = (1 + W) / 2;
    yc = (1 + W) / 2;
% Use nested for loops to check if each pixel lies in the foreground of the
    for i = 1 : W
        for j = 1 : W
            if sqrt((j - xc)*(j - xc) + (i - yc)*(i - yc)) <= (diameter/2)
                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

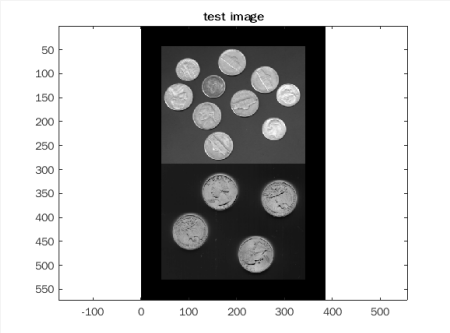
Assessment: All Tests Passed

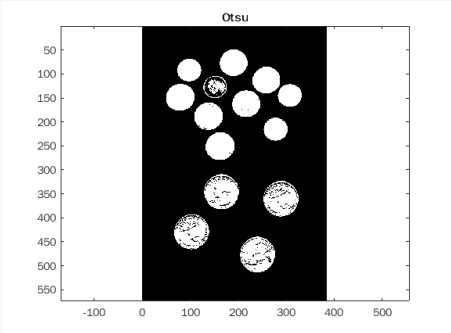
Submit

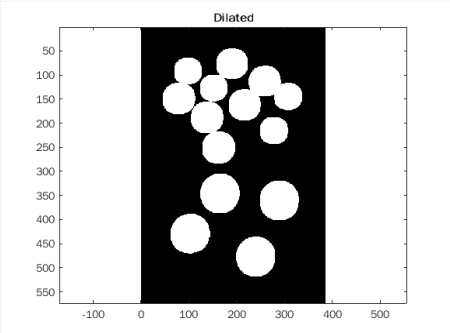
- ✔ **Is nickelfilter correct?**
- ✔ **Is dimefilter correct?**
- ✔ **Is quarterfilter correct?**
- ✔ **Is D correct?**

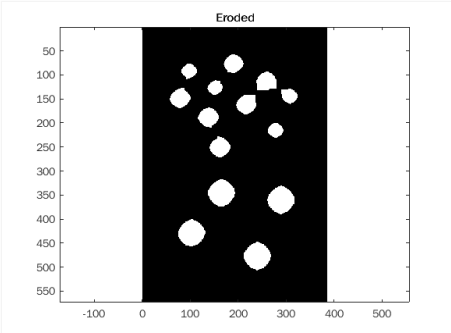
Output

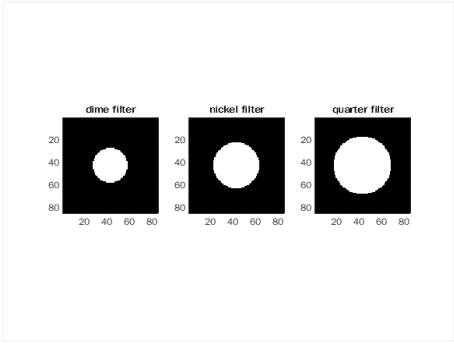
```
N =

    14


filtsize =

    85


D =

    0.7237    0.9180    0.7013
    0.4948    0.6857    0.9160
    0.8499    0.6177    0.4439
    0.7157    0.9260    0.7044
    0.7801    0.5502    0.3817
    0.4932    0.6835    0.9138
    0.7196    0.9573    0.7453
    0.7392    0.8675    0.6413
    0.6415    0.8135    0.6234
    0.4858    0.6732    0.9001
    0.5697    0.7251    0.5555
    0.4857    0.6730    0.8998
    0.9610    0.7130    0.5333
    0.7024    0.5428    0.3613
```











© 2020 The MathWorks, Inc.