Homework 3 Solution

Contents

1(a)

I will add extra zeros (1 rows on each edge, since our window is 3x3) to frame the original image, then loop through the pixels that correspond to the original image and compute the median in all 3x3 window.

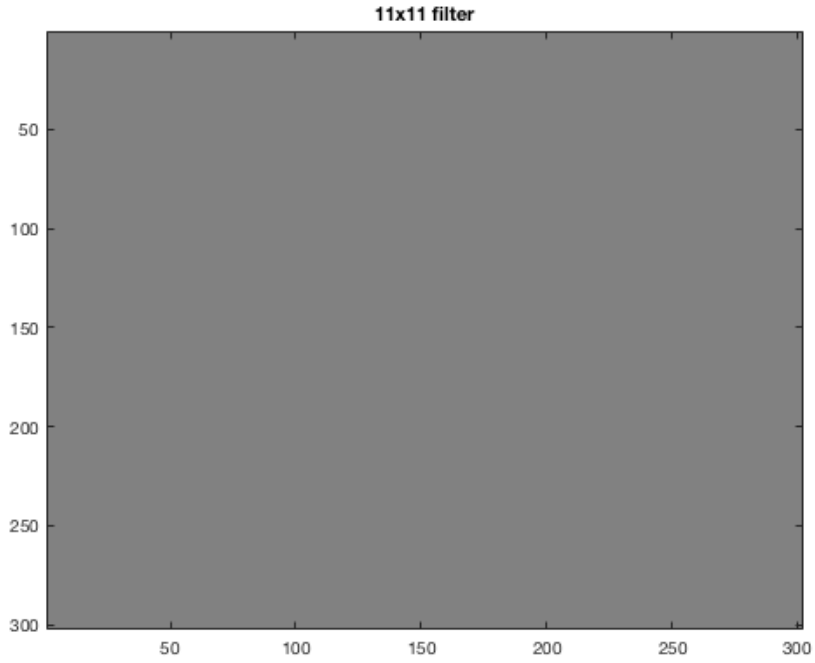
```
load('/Users/jeanettemumford/Dropbox/Research/Teaching/BMI_567_2018/Homework/Assignment3/medfilt_problem_dat.mat')
img_dim = size(dat);
pad_zeros = zeros(img_dim+2);
pad_zeros(2:(end-1), 2:(end-1)) = dat;

med_filt_img = 0*dat;
for i=2:(1+img_dim(1))
    for j=2:(1+img_dim(2))
        dat_loop = pad_zeros((i-1):(i+1), (j-1):(j+1));
        med_filt_img(i-1, j-1) = median(dat_loop(:));
    end
end

% Compare to medfilt2 using a difference of images
medfilt2_result = medfilt2(dat, [3,3]);

diff = med_filt_img-medfilt2_result;
min(diff(:))
max(diff(:))
% Images match since difference is all zeros
```

```
ans =

     0


ans =

     0
```

1b

By inspection, I found and 11x11 filter removed all of the noise

```
filt_img9x9 = medfilt2(dat, [9,9]);
figure
colormap(gray)
imagesc(filt_img9x9)
title('9x9 filter')
filt_img11x11 = medfilt2(dat, [11,11]);
colormap(gray)
imagesc(filt_img11x11)
title('11x11 filter')
```



1c

If the noise is 5x5, I need to make sure my median filter window size is large enought that the median is 0. This would mean more than half the values in the window would need to be 0 to guarantee the median is 0. Since the noise contains 25 nonzero values, we'd need the window to have at least 50 pixels in it. Since it is square and the dimension needs to be odd, we just need find the closest odd number above the square root of 50.

```
sqrt(50)
```

```
ans =

    7.0711
```

Since the square root of 50 is just above 7, the window must be 9x9 to guarantee removal of isolated bits of noise. The reason I needed to use an 11x11 filter above is because of instances where noise pieces were not isolated, but were located near other pieces of noise in the image.

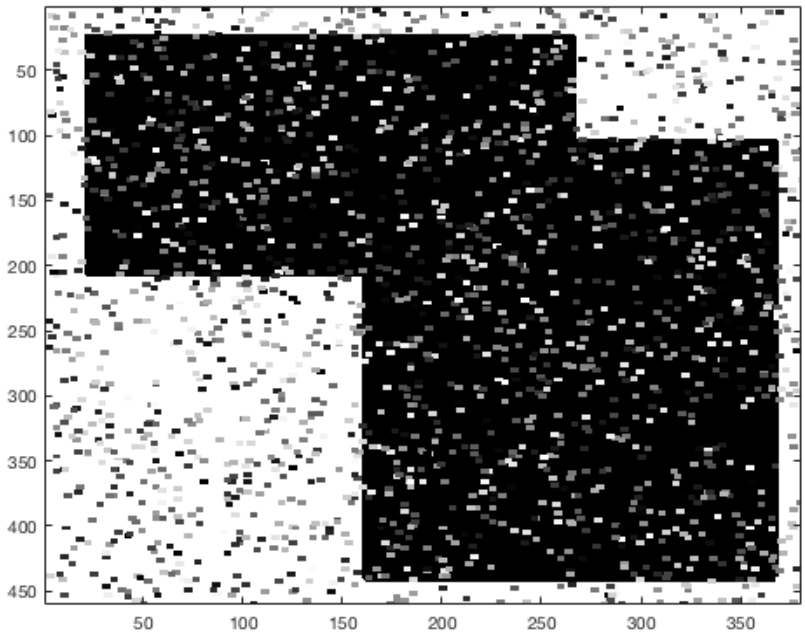
2

I'm using addpath to point it to where the houg_trans.m function lives

```
addpath ~/Dropbox/BMI_567_2017/ClassMaterials/Week7_2_28_3_2/
```

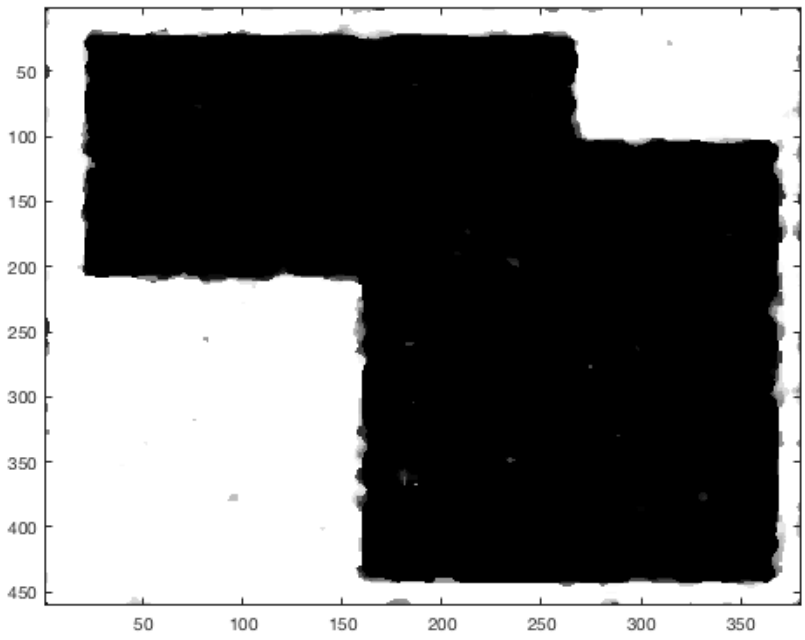
Changing into the directory where the data live

```
load('/Users/jeanettemumford/Dropbox/Research/Teaching/BMI_567_2018/Homework/Assignment3/dat_hough.mat')
img_hwk = dat_hough;

figure
imagesc(img_hwk)
colormap(gray)
```



The median filter can remove this type of noise. I found a window size of 11 worked pretty well. If it is too large, the corners become too rounded.

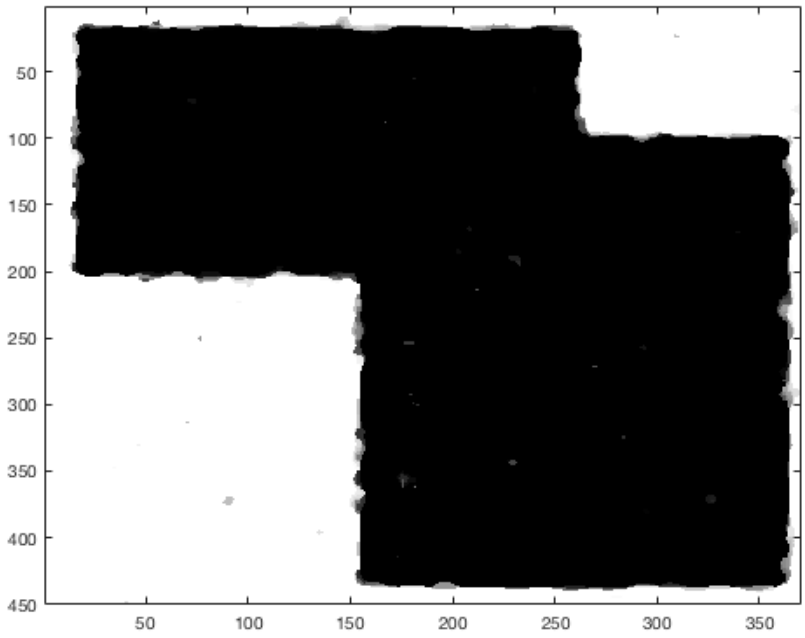
```
med_img_hwk = medfilt2(img_hwk, [11,11]);
figure
colormap(gray)
imagesc(med_img_hwk)
```



Remove exactly the amount of edges that would be impacted by the median filter edge artifact Since my filter is 11 wide, I need to trim 5 off all edges. Some of you may not have done this, but then your result will pick up lines along the edges of the image. That's okay.

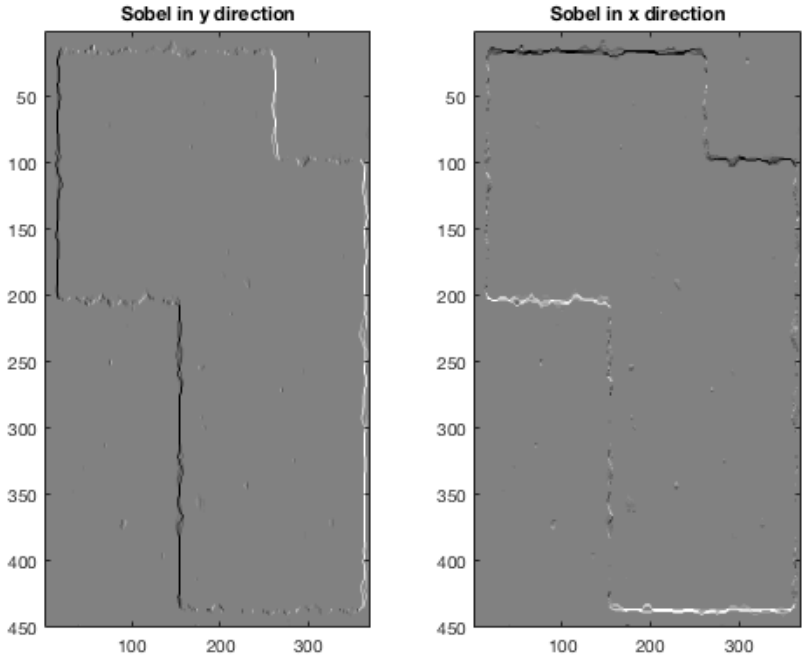
```
med_img_hwk = med_img_hwk(6:(end-5), 6:(end-5));
figure
```

```
imagesc(med_img_hwk)
colormap(gray)
```



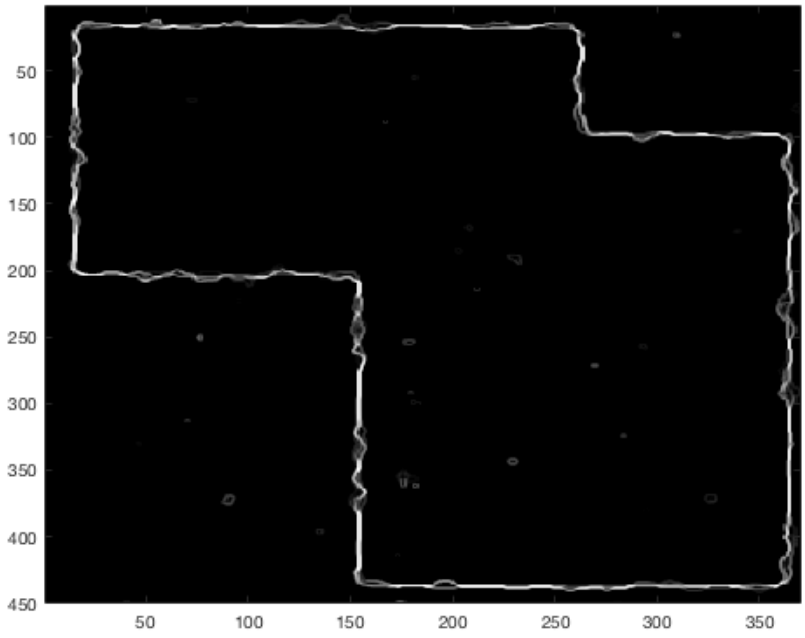
Applying the Sobel filter using conv2 to find the edges.

```
k_dx = [1 2 1; 0 0 0 ; -1 -2 -1];
k_dy = [1 0 -1; 2 0 -2 ; 1 0 -1];

conv_dy = conv2(med_img_hwk, k_dy, 'same');
conv_dx = conv2(med_img_hwk, k_dx, 'same');
figure
subplot(1,2,1)
imagesc(conv_dy)
title('Sobel in y direction')
colormap(gray)
subplot(1,2,2)
imagesc(conv_dx)
title('Sobel in x direction')
```



Combine to show the edges.

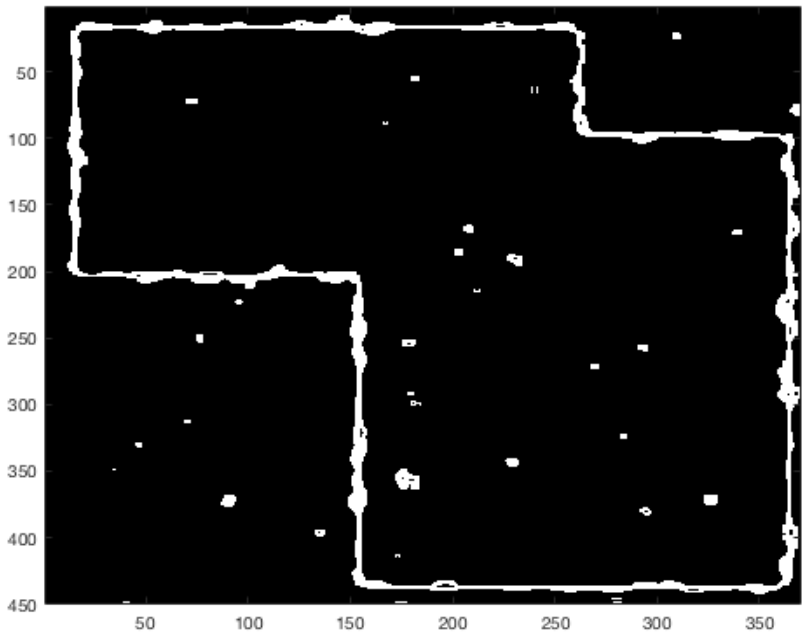
```
diff_img = sqrt(conv_dy.^2 + conv_dx.^2);
figure
imagesc(diff_img)
colormap(gray)
```



Binarize the output from the Sobel filter. Note, it is important that the edges are set to 1. I found 3 to be a decent looking threshold. If you try too hard to remove all the speckle artifact you end up losing some of the border.

```
bin_img = (diff_img>3)*1;
figure
```

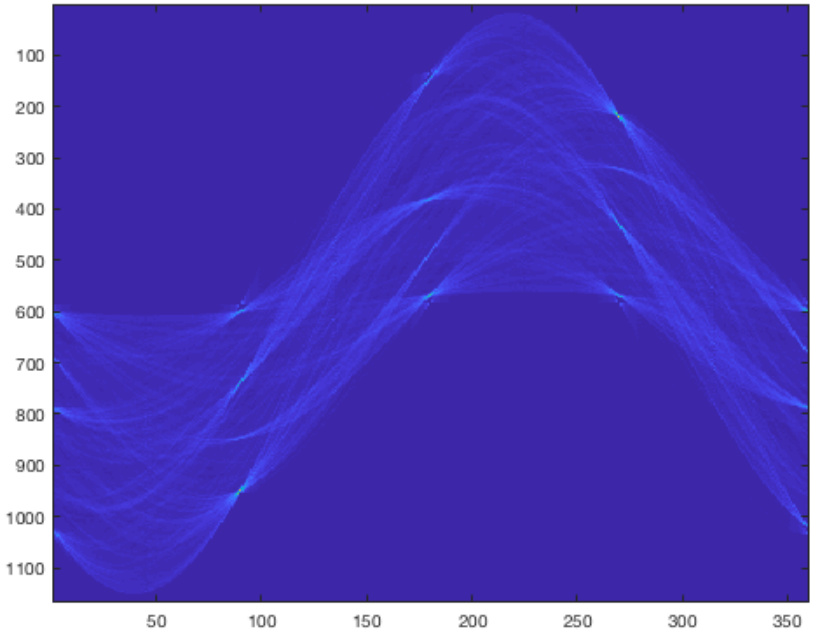
```
imagesc(bin_img)
colormap(gray)
```



2 (e)

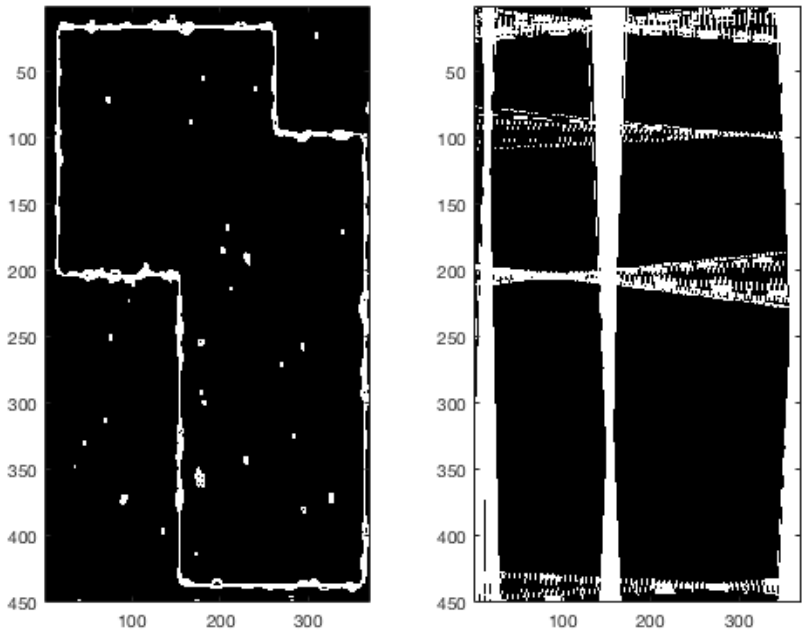
Run the Hough transform using the function from class.

```
[himg, theta, r] = hough_trans(bin_img);

figure
imagesc(himg)

maxint=max(max(himg));
```



A much lower cut point is necessary in this image, compared to the one we did in class. That is because the class example had really thick edges. 20% of the max worked okay for me. This will vary based on the median filter you used and the threshold you used to binarize the image.

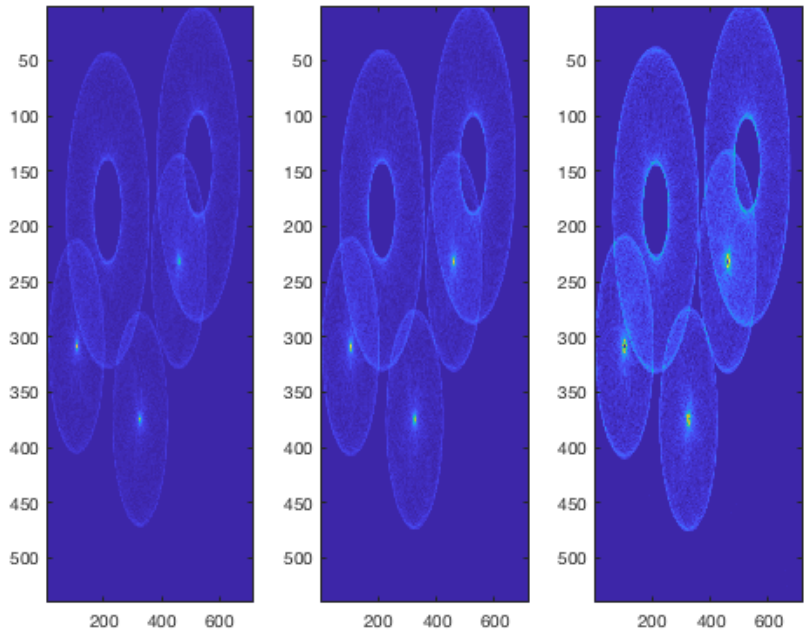
```
[i, j] = find(himg > 0.2*maxint);

theta_rad = theta*pi/180;

slopes = -1*cos(theta_rad(j))./sin(theta_rad(j));
intercepts = r(i)./sin(theta_rad(j));

img_lines = zeros(size(bin_img));
n_lines = length(slopes);
for i = 1:n_lines
  xval = [1:0.1:size(bin_img,1)]';
  yval = intercepts(i) + xval*slopes(i);
  xval = round(xval);
  yval = round(yval);
  keep = yval< size(bin_img,2) & yval>0;
  xval_keep = xval(keep);
  yval_keep = yval(keep);
  for index = 1:length(xval_keep)
     img_lines(xval_keep(index), yval_keep(index)) = 1;
  end
end

figure
subplot(1, 2, 1)
imagesc(bin_img)
subplot(1, 2, 2)
imagesc(img_lines)
colormap(gray)
```



3

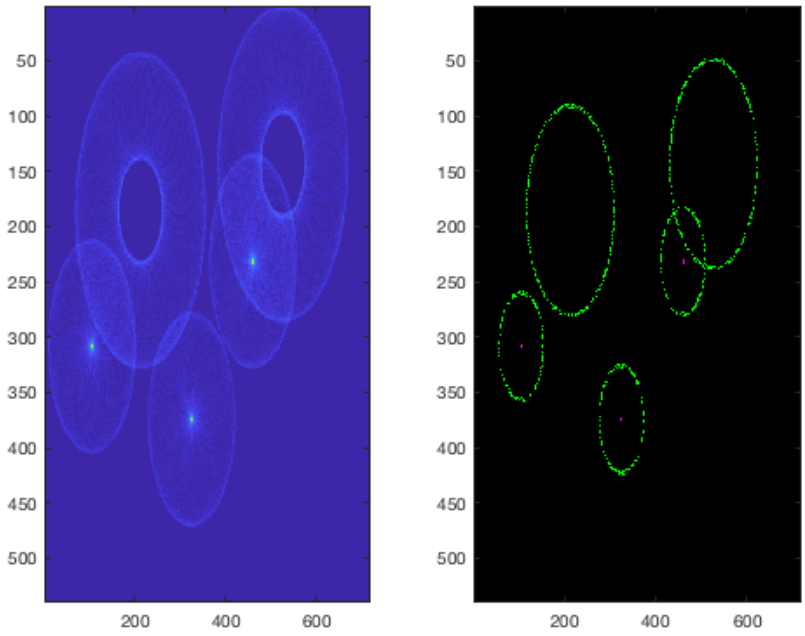
```
load('/Users/jeanettemumford/Dropbox/Research/Teaching/BMI_567_2018/Homework/Assignment3/img_circle.mat')

irng = size(img_circle,1);
jrng = size(img_circle, 2);

rad_vec = [48, 50, 52];
nrad = length(rad_vec)
centers = zeros(irng, jrng, nrad);
for rad=1:3
for i=1:irng
    for j = 1:jrng
        for ang = 1:360
            if img_circle(i,j)==1
                x_center = round(i - rad_vec(rad)*cos(ang*pi/180));
                y_center = round(j - rad_vec(rad)*sin(ang*pi/180));
                if (x_center>0 & x_center<irng) & (y_center>0 & y_center<jrng)
                    centers(x_center, y_center, rad) = centers(x_center, y_center, rad) + 1;
                end
            end
        end
    end
end
end




figure
subplot(1, 3, 1)
imagesc(centers(:,:,1))
subplot(1, 3, 2)
imagesc(centers(:,:,2))
subplot(1, 3, 3)
imagesc(centers(:,:,3))

max(max(centers(:,:,1)))
max(max(centers(:,:,2)))
max(max(centers(:,:,3)))
```

```
nrad =

     3


ans =

   204


ans =

   139


ans =

    81
```



The best radius is 48, since the accumulator matrix has the largest maximum.

```
img_centers = centers(:,:,1)>120;
fused_img = imfuse(img_circle, img_centers);
figure
subplot(1,2,1)
imagesc(centers(:,:,1))
subplot(1,2,2)
imagesc(fused_img)
```



Published with MATLAB® R2017b