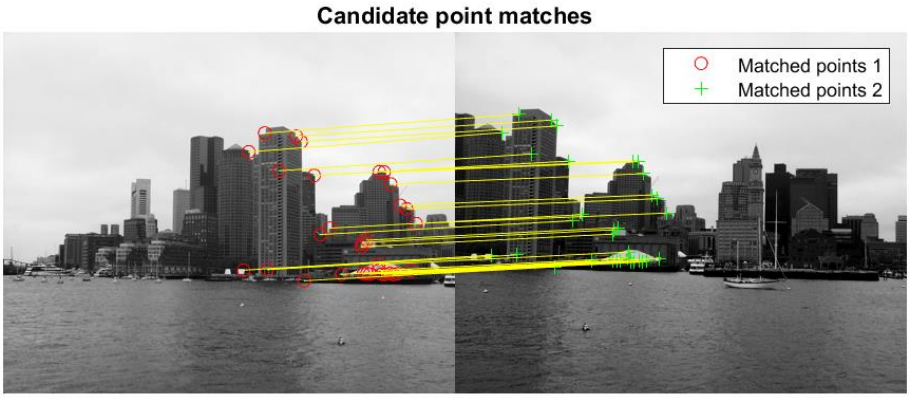
CS-E4850 Computer Vision Exercise Round 4

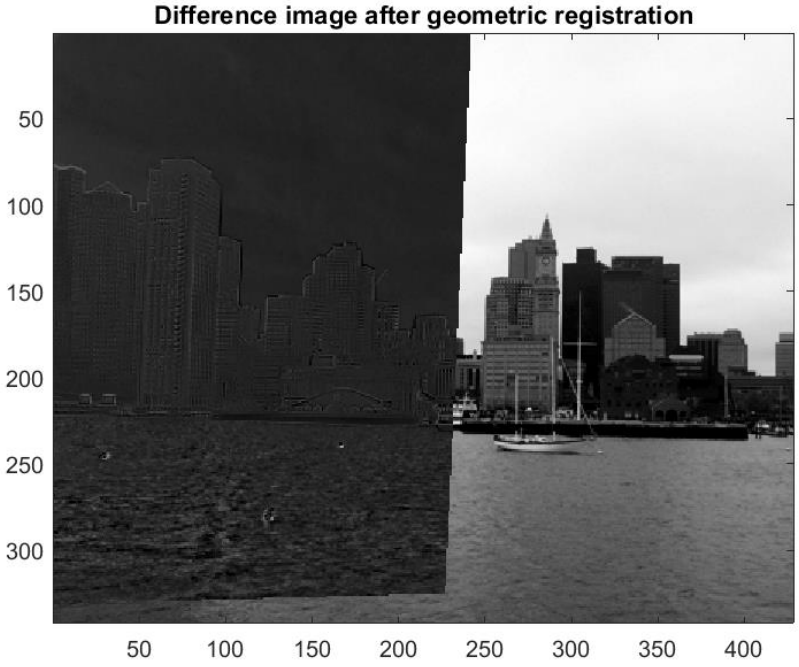
Laura Alejandra Encinar

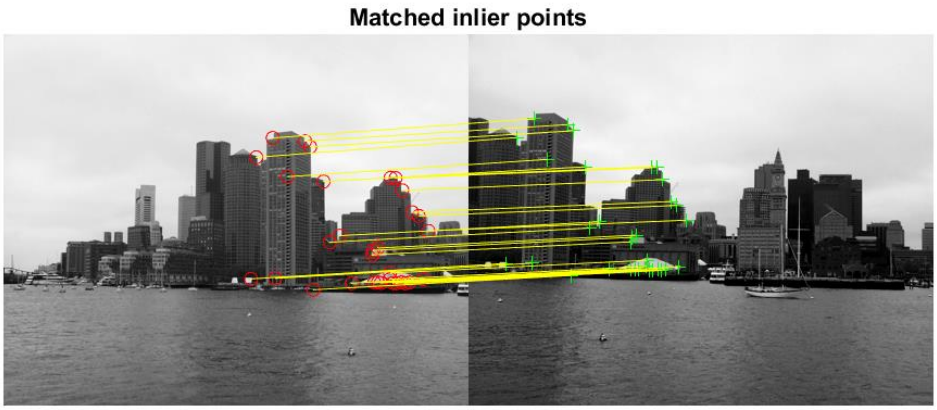
28/09/2023

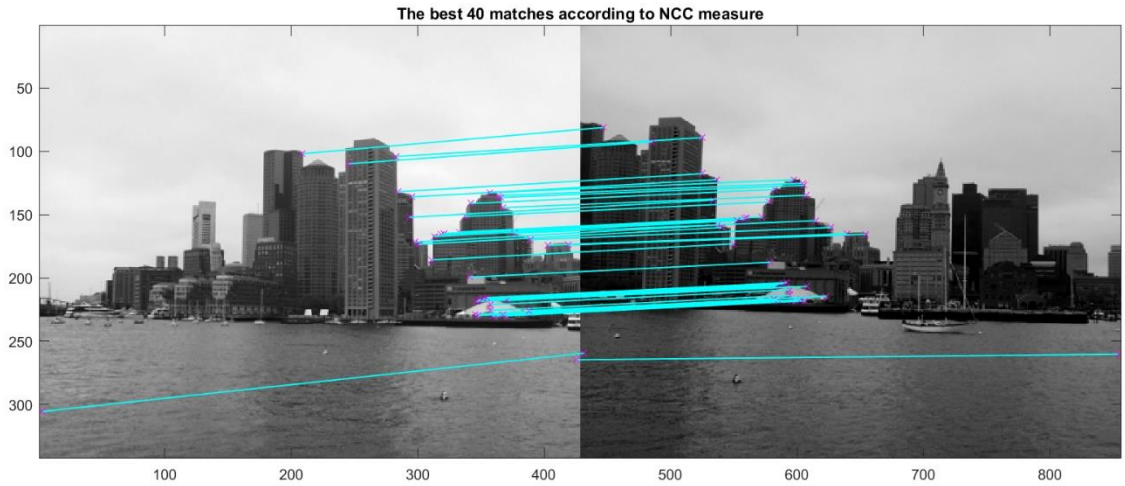
Exercise 1. Matching Harris corner points

a) Results after substituting SSD for NCC:



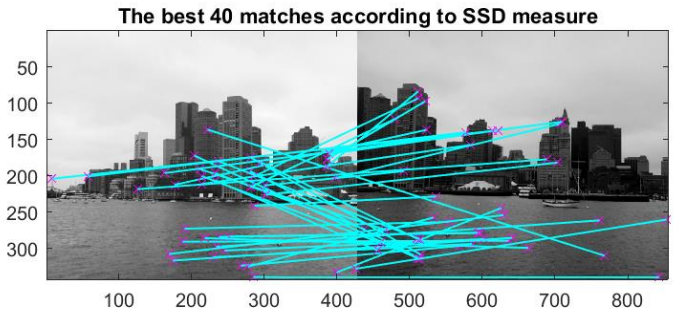






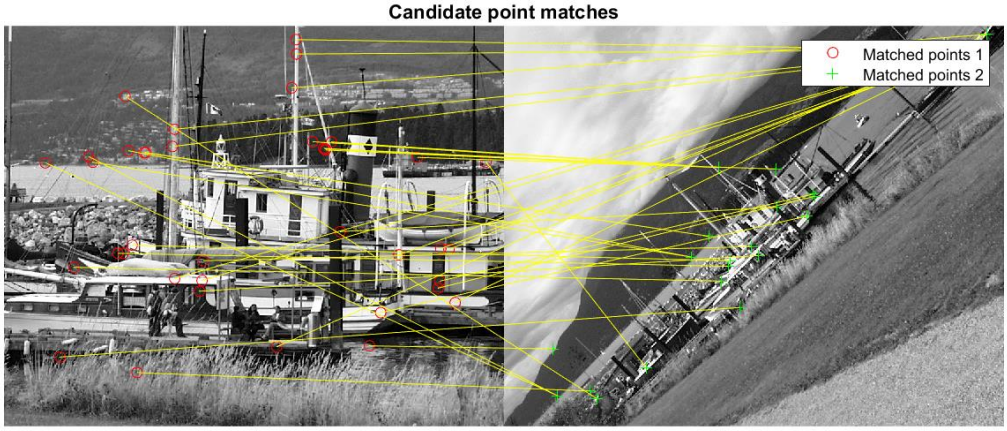
b) Using NCC I got **295** correct correspondences, whereas using SSD I got only 49.

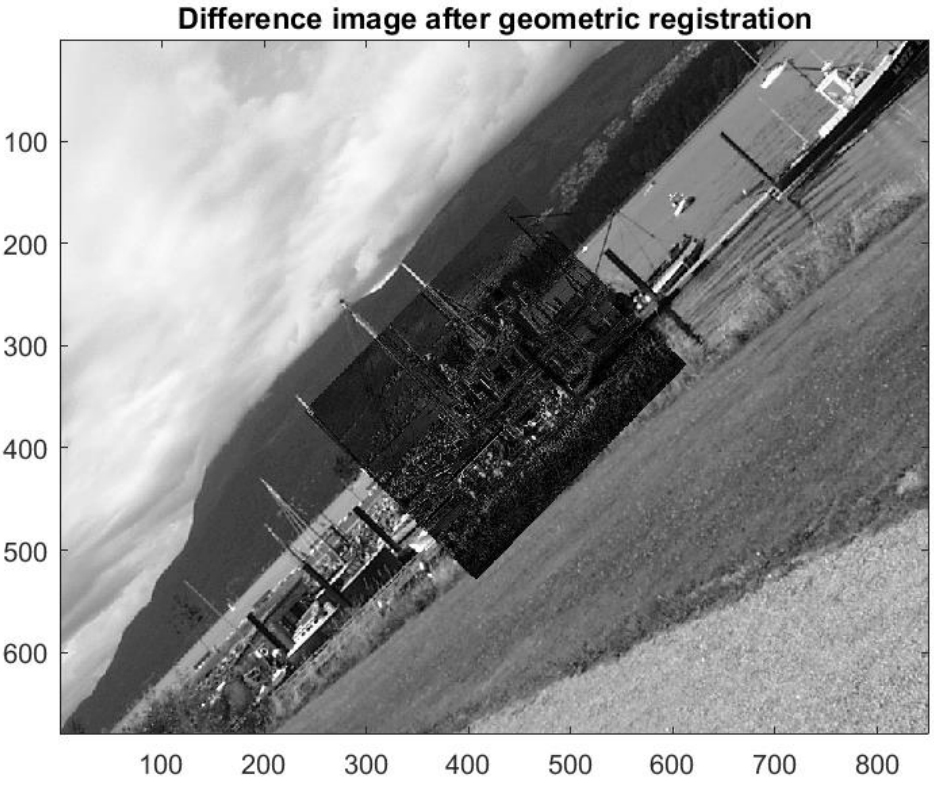
c) In this case, we can conclude that NCC performs better than SSD since the number of correct correspondences is greater. In addition, if we compare both best 40 matches, we can appreciate that the results of using NCC are more accurate.

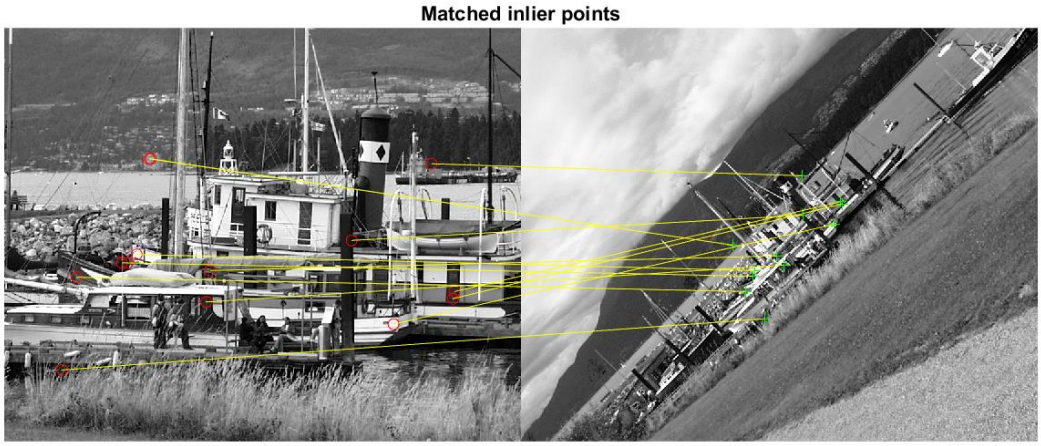


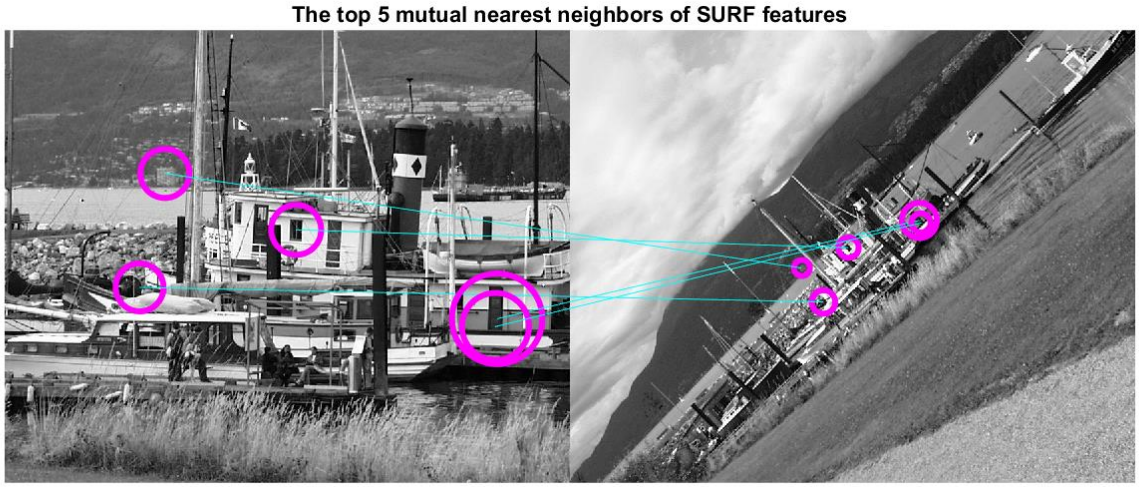
Exercise 2. Matching SURF regions.

a) Results after substituting nearest neighbour distance for nearest neighbour distance (NNDR).

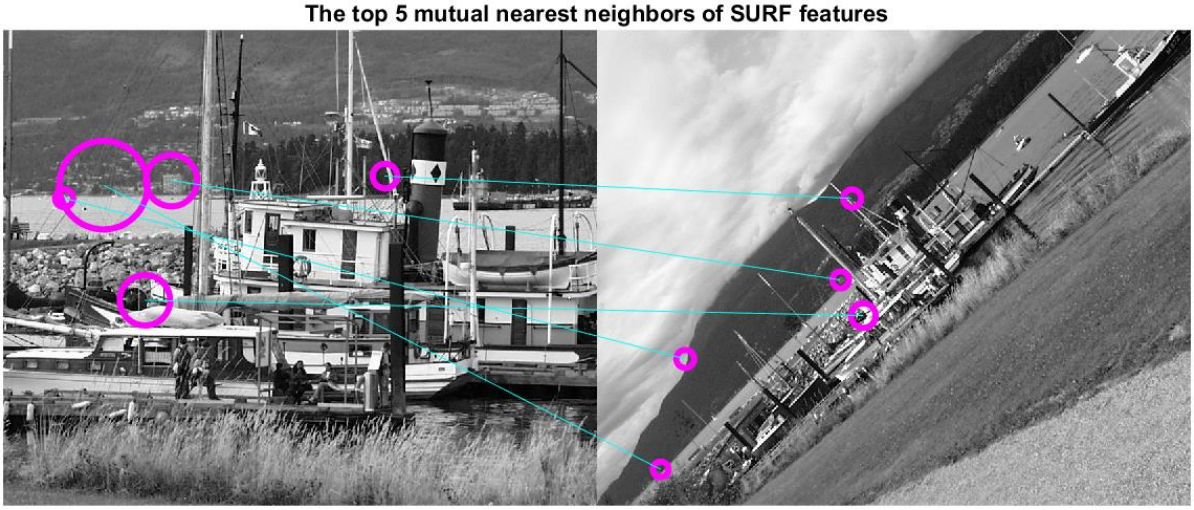








The number of correct correspondences is 5 out of 5, whereas if we look at the results based on nearest neighbour distance, we found out that only 2 of the correspondences are correct.



b) The matching approach employed in task 1 (Harris corners and NCC-based matching) is not suitable for the example images in task 2 due to significant scale and rotation variations. In task 1, the image featured a building landscape with prominent corners that were relatively easy to detect using Harris corners. However, task 2 involves an image with more intricate details, including textured regions, and a rotation factor. Harris corners and NCC-based matching are not inherently designed to handle these transformations effectively.

In summary, SURF regions offer greater versatility and robustness in addressing scale and rotation variations, making them a more suitable choice for matching tasks in scenarios like task 2. Nonetheless, Harris corners may still find relevance when the primary objective is corner detection, and scale and rotation invariance are not critical considerations.

Exercise 3. Scale-space blob detection

a) Generate a Laplacian of Gaussian filter. (You can set σ = 0.5.)

```
%Generate a Laplacian of Gaussian filter
[g,gx,gy,gxx,gyy,gxy]=gaussian2(sigmas(i)^2);
scaleNormalizedLaplacian{i}=sigmas(i)^2*(gxx+gyy);
```

b) Build a Laplacian scale space, starting with some initial scale and going for n iterations.

```
%filter image with scale-normalized Laplacian at current scale
%save square of Laplacian response for current level of scale space in
%tmpgxx and tmpgyy

tmpxx = conv2(img, gxx, 'same');
tmpyy = conv2(img, gyy, 'same');

scalespace(:,:,i)=(sigmas(i)^2*(tmpxx+tmpyy)).^2;

%increases scale i by factor k
sigmas(i)=k^(i-1)*sigma0;
```

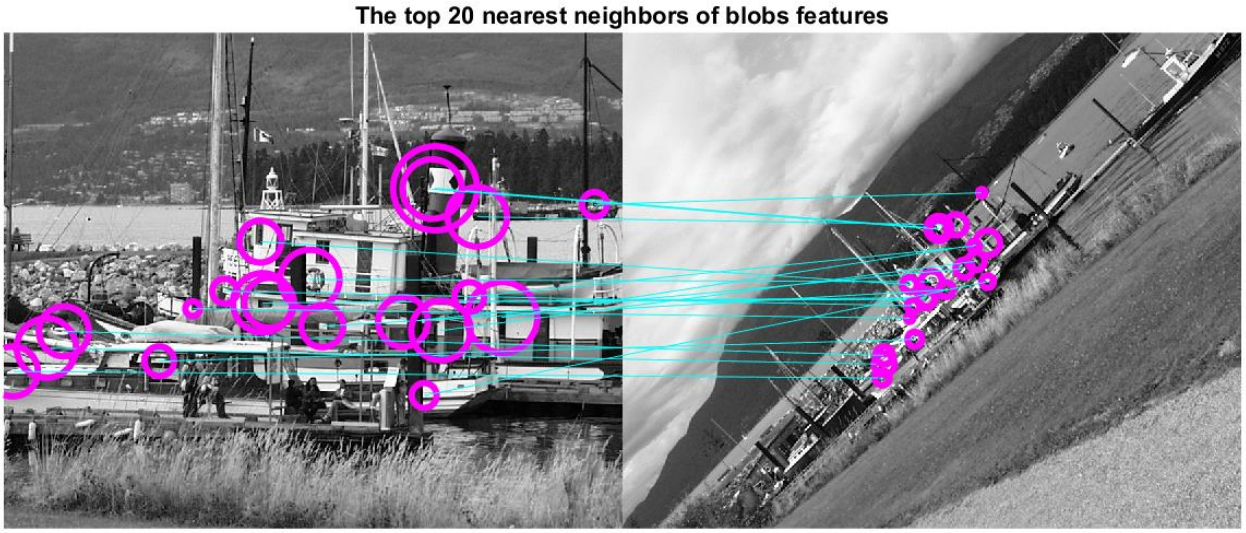
c) Perform non-maximum suppression in scale space.

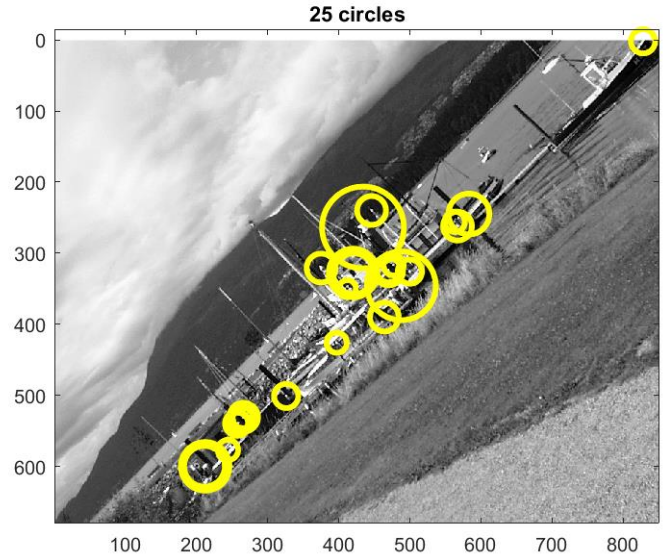
```
%Perform non-maximum suppression in scale space.
localmaxima=zeros(size(scalespace));
for i=1:Nscales
    maxi=colfilt(scalespace(:,:,i),[3 3],'sliding','max([x(1:4,:);x(6:9,:)])');
    localmaxima(:,:,i)=scalespace(:,:,i)>maxi;
end
```

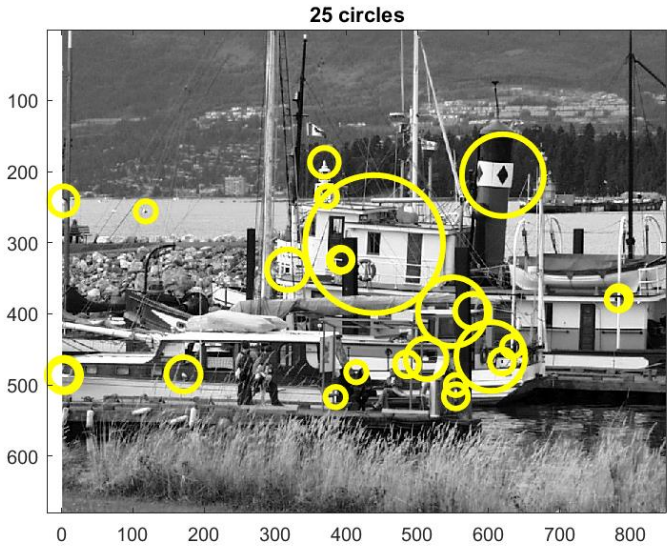
d) Display resulting circles at their characteristic scales.

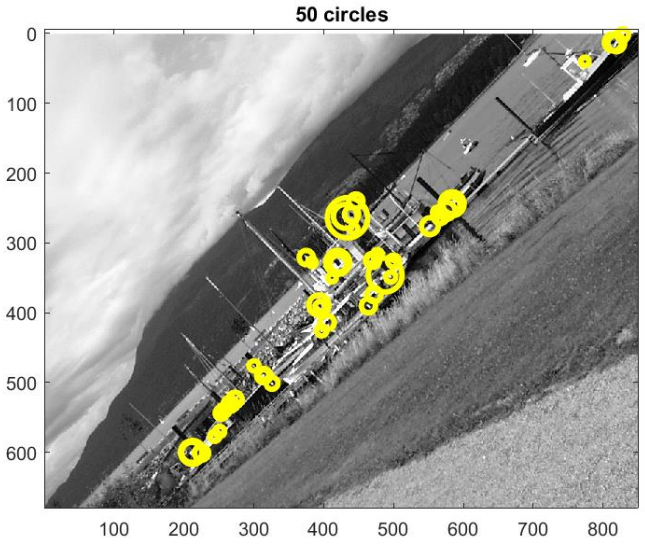
```
%visualization
show_all_circles(img, sblobs(1:NVIS,1), sblobs(1:NVIS,2), 6*sqrt(2)*sblobs(1:NVIS,3), 'y', 3);
```

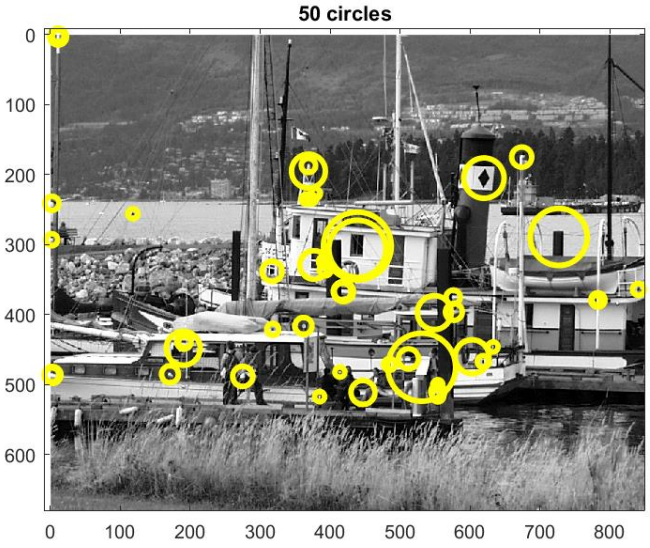
Results after running the script.








50 circles
0
100
200
300
400
500
600
100
200
300
400
500
600
700
800


50 circles
0
100
200
300
400
500
600
0
100
200
300
400
500
600
700
800