ECE415 -- Homework 5

Name: Wang Yunjuan

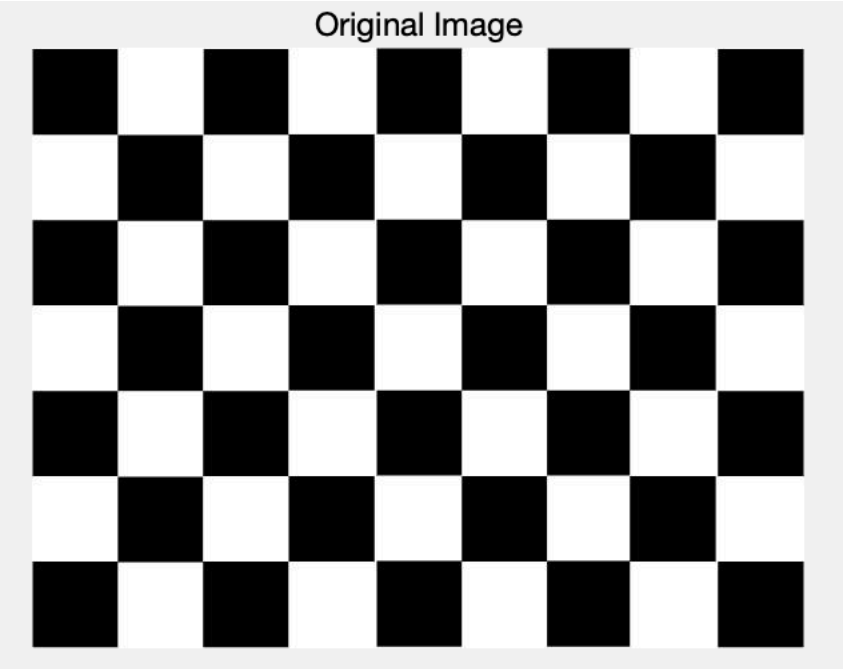
UIN:661180568

Problem 1

Detect features in the image ‘checkerboard.jpg’ using the Harris detector.

1) Display the image

```
im=imread('checkerboard.jpg');
img=im(:,:,1);
figure(1)
imshow(im)
title('Original Image')
```



2) Detect the features using Harris detector. Assume that the patches are of size 5x5 pixels.

a) Display the x-derivative, y-derivative.

I define convolute function ahead of time.

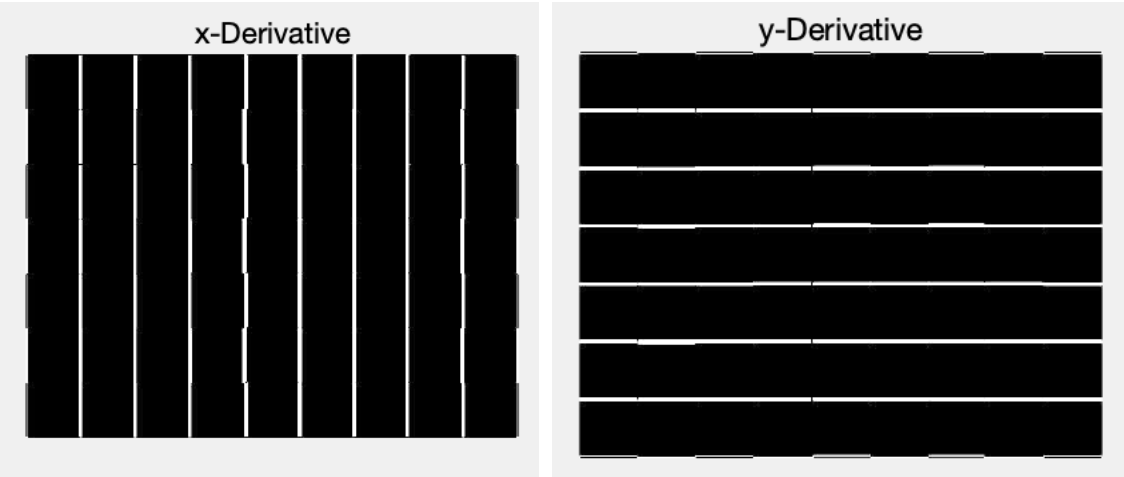
```
function resultimg = convolute(kernel,img)
[k_size_x,k_size_y]=size(kernel);
[img_x,img_y]=size(img);
img=double(img);
pad_x=(k_size_x-1)/2;
pad_y=(k_size_y-1)/2;
img=[zeros(pad_x,img_y);img;zeros(pad_x,img_y)];
img=[zeros(img_x+2*pad_x,pad_y) img zeros(img_x+2*pad_x,pad_y)];
```

```
resultimg=zeros(img_x,img_y);
for i=1:img_x
    for j=1:img_y
        for k=1:k_size_x
            for l=1:k_size_y
                resultimg(i,j)=resultimg(i,j)+kernel(k_size_x+1-k,k_size_y+1-
l)*img(i+k-1,j+l-1);
            end
        end
        resultimg(i,j)=abs(resultimg(i,j));
    end
end
```

Then I use this function to calculate the derivative.

```
%% Q2(a)
%assume that the kernel is kernel [-2 -1 0 1 2], use zero-padding
dd=[-2 -1 0 1 2];
dx=convolute(dd,img);
dy=convolute(dd',img);

dx=uint8(dx);
dy=uint8(dy);
figure(2)
subplot(2,1,1);
imshow(dx);
title('x-Derivative');
subplot(2,1,2);
imshow(dy);
title('y-Derivative');
```



b) Use Forstner-Harris metric to measure usefulness of the features.

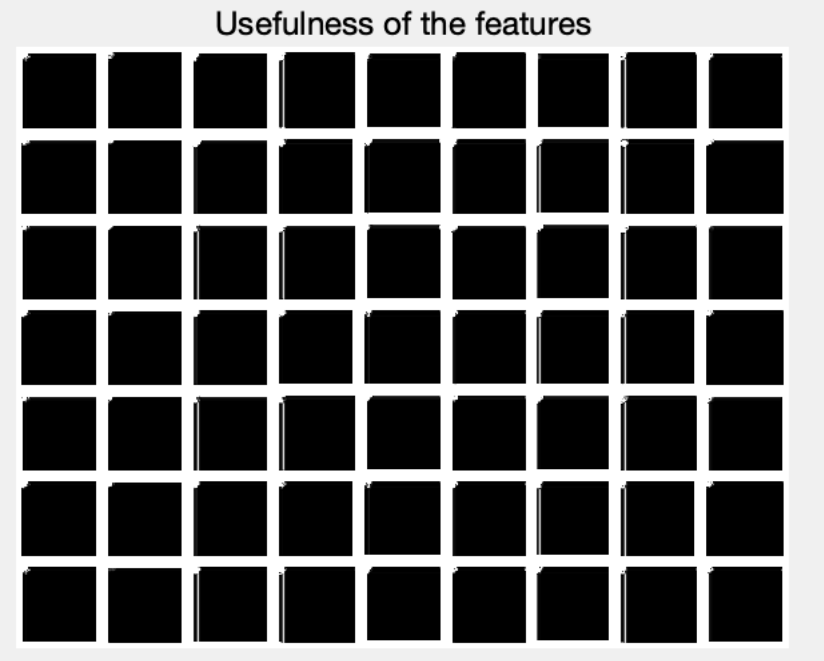
Define 5*5 Gaussian kernel to sommoth Ixx, Ixy, Iyy.

Calculate R based on auto-correlated matrix.

Turn R matrix into unit8 and output R.

```
%% Q2(b)
dx=double(dx);
dy=double(dy);
Ixx=dx.*dx;
Iyy=dy.*dy;
Ixy=dx.*dy;

%use 5*5 Gaussian filter to smoothen A
G = [exp(-8), exp(-5), exp(-4), exp(-5), exp(-8)
    exp(-5), exp(-2), exp(-1), exp(-2), exp(-5)
    exp(-4), exp(-1), 1, exp(-1), exp(-4)
    exp(-5), exp(-2), exp(-1), exp(-2), exp(-5)
    exp(-8), exp(-5), exp(-4), exp(-5), exp(-8)];
G = 1/(sum(sum(G))).* G;
Ixx=convolute(G,Ixx);
Ixy=convolute(G,Ixy);
Iyy=convolute(G,Iyy);
k = 0.06;
R = (Ixx.*Iyy-Ixy.*Ixy)-(k*(Ixx+Iyy).^2);
RR = abs(R);
RR=double(uint8(RR));
RR=(RR-min(RR(:))./(max(RR(:))-min(RR(:))).*255;
RR=uint8(RR);
figure(3)
imshow(RR)
title('Usefulness of the features');
```



c) Select features based on the value of local maximums. What threshold did you use?

I define the number of features as Num. Num=80. Then I sort all the R(:) descend. The R(Num) is the threshold.

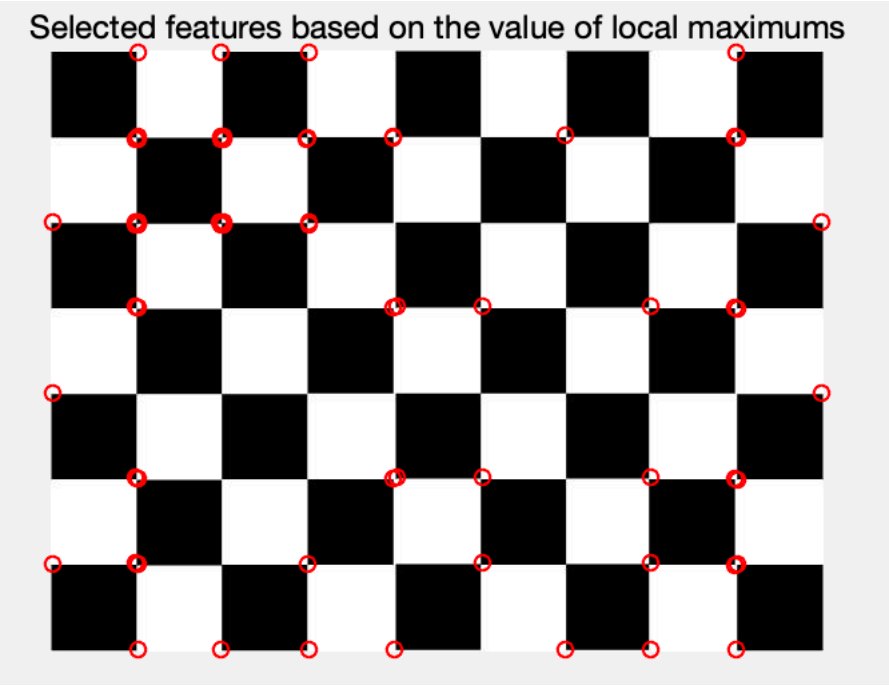
Threshold= 1.772938912293658e+10

Display image with marked locations of the selected features.

```
Num=80;
temp=sort(R(:),'descend');
temp=temp(Num);

figure(4)

imshow(img)
for i=1:row
    for j=1:col
        if R(i,j)>=temp
            hold on
            plot(j,i,'ro')
        end
    end
end
title('Selected features based on the value of local maximums')
```



3) Use adaptive non-maximal suppression to select features.

a) Select same number of features as you had in part 3.

The number of features I selected is 80.

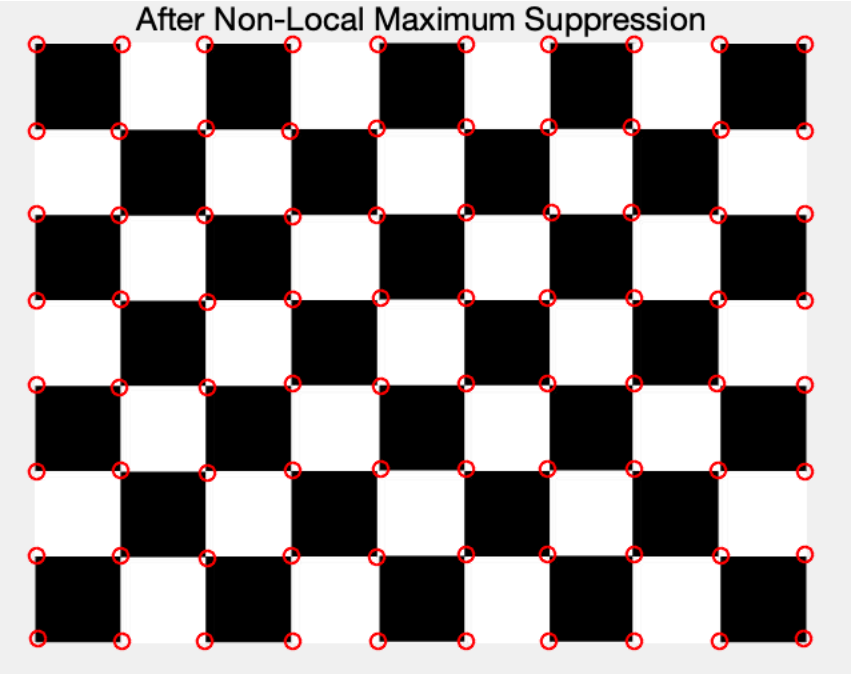
b) Display image with marked locations of the selected features.

I choose the patch size r is 11.

```
r=11; %radius of local maximum using adaptive non-maximal suppression
corner=zeros(row,col);
R=[zeros((r-1)/2,col);R;zeros((r-1)/2,col)];
R=[zeros(row+(r-1),(r-1)/2) R zeros(row+(r-1),(r-1)/2)];

t=zeros(r,r);
count=0;
for i=1:row
    for j=1:col
        t=R(i:i+r-1,j:j+r-1);
        tmax=sort(t(:),'descend');
        if(tmax(1)==R(i+(r-1)/2,j+(r-1)/2)&&tmax(1)~=tmax(2))
            corner(i,j)=1;
            count=count+1;
        end
    end
end

R=R((r-1)/2+1:end-(r-1)/2,(r-1)/2+1:end-(r-1)/2);
s=R.*corner;
t=sort(s(:),'descend');
t=t(Num);
figure(5)
imshow(img)
for i=1:row
    for j=1:col
        if s(i,j)>=t
            hold on
            plot(j,i,'ro')
        end
    end
end
title("After Non-Local Maximum Suppression")
```

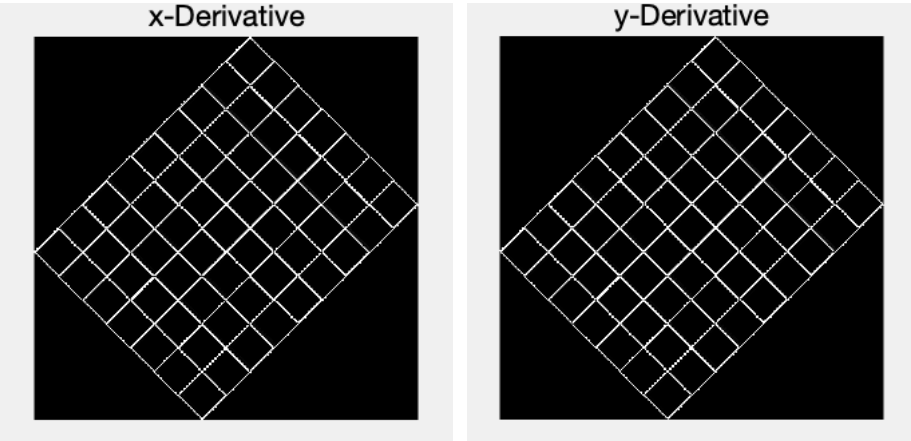


c) Comment on the location of features in part 2 and part 3.

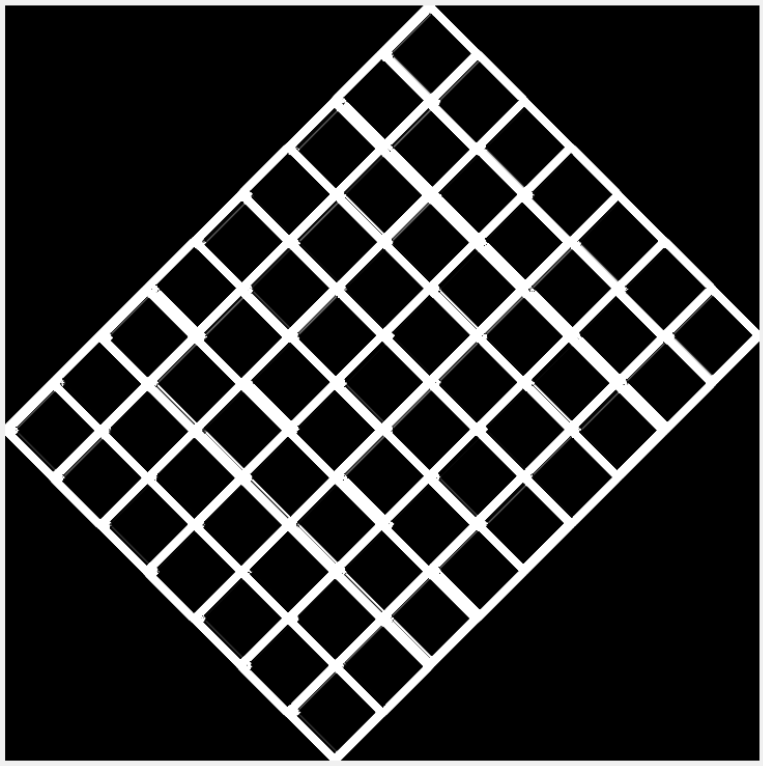
We can easily find that the result of non-local maximum suppression is far better than select features based on the value of local maximums. Because if we only look for local maxima in the interest function, this can lead to an uneven distribution of feature points across the image. With non-maximum suppression, detected corners are more spread out.

4) Rotate the original image by 45 degrees using built-in Matlab function imrotate(). Display the rotated image.

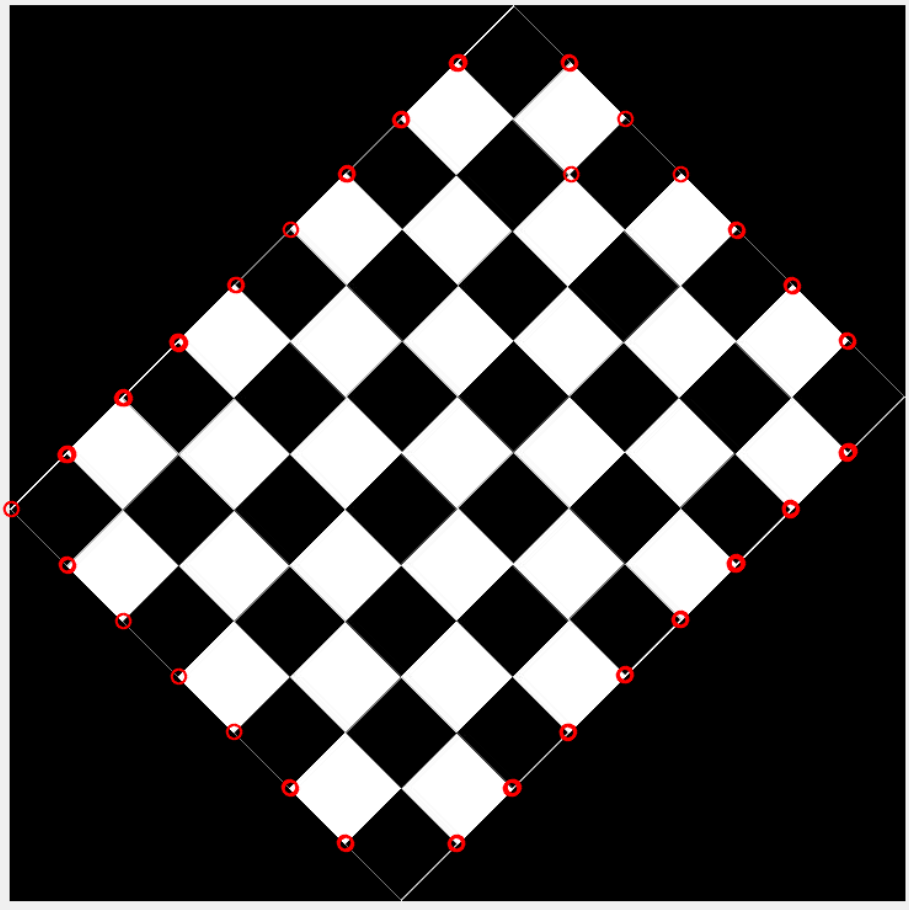
5) Repeat steps 2 and 3 for the rotated image.

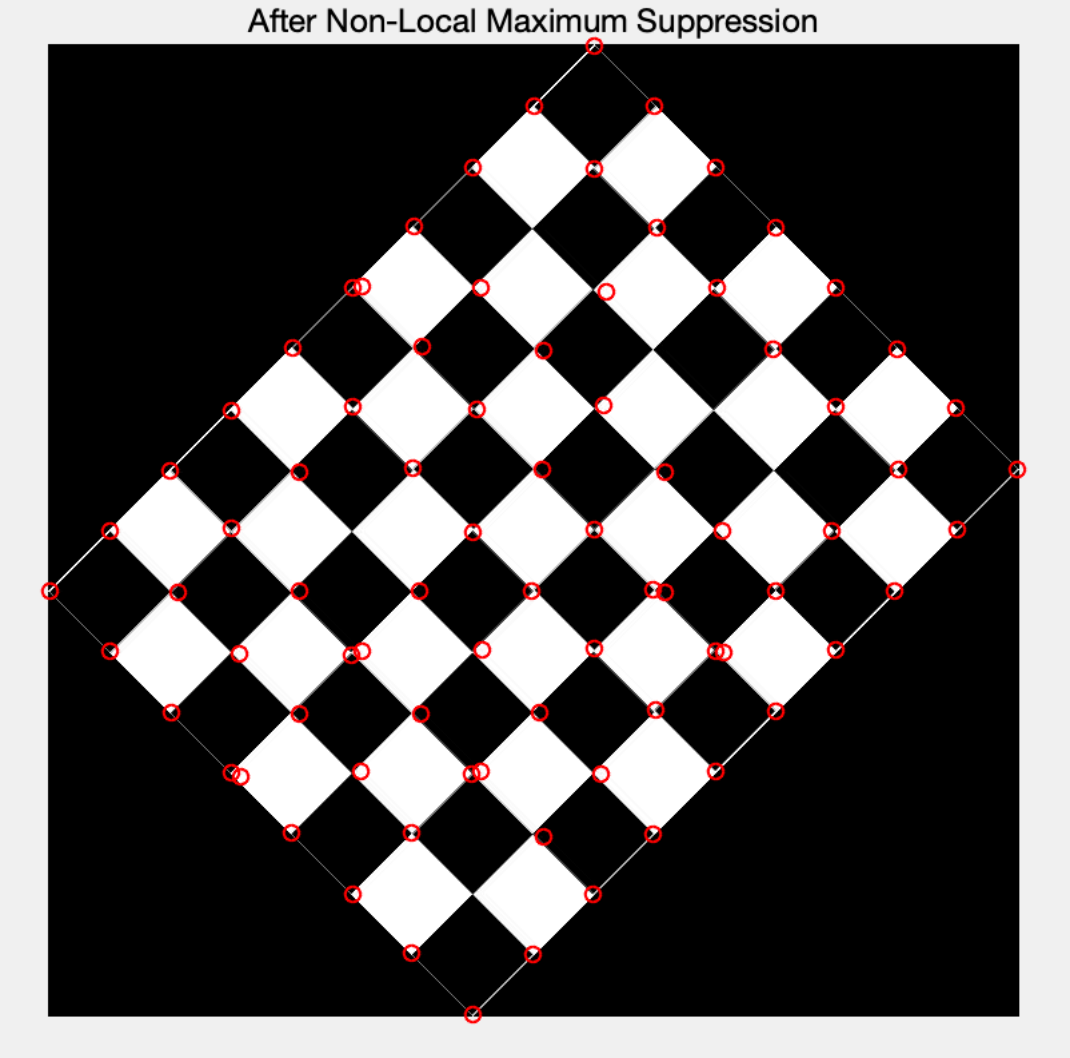


Usefulness of the features



Selected features based on the value of local maximums





6) Match the features found using adaptive non-maximal suppression in the original image to the ones found in the same way in the rotated image. For comparison use sum of squared differences between patches. Keep in mind that before matching two features you have to rotate both of them to align their orientation. How many features were matched? Display the original image with marked locations of the matched features.

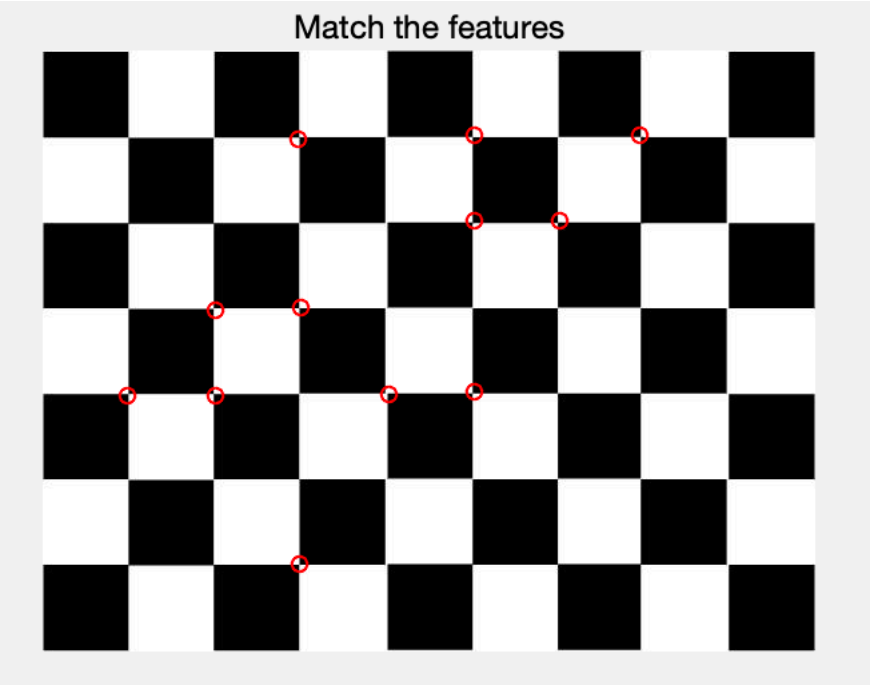
I define what I have done above into a function for simplicity in the following coding.

```
function [corner,dx,dy,R]=detHarrisCorners(img,Num)
```

I define function rotatefeature to calculate the dominate orientation and rotate them align to x-arix.

I delete the features on the boundary and match the remaining features. I choose the threshold=1.5*e11.

40 features were matched. According to the image, we can get the conclusion that some features are mismatching.




```
function mask=rotatefeature(feature,deri_x,deri_y,patch)
[size_x,size_y]=size(feature);
x=zeros(1,8);
for i=1:size_x
    for j=1:size_y

        angle=atan(deri_y(i,j)/deri_x(i,j))/2/pi*360;
        mag=sqrt(deri_x(i,j)^2+deri_y(i,j)^2);
        if angle>=0&&angle<45
            x(1)=x(1)+mag;
            continue
        end
        if angle>=45&&angle<90
            x(2)=x(2)+mag;
            continue
        end
        if angle>=90&&angle<135
            x(3)=x(3)+mag;
            continue
        end
        if angle>=135&&angle<180
            x(4)=x(4)+mag;
            continue
        end
        if angle>=180&&angle<225
            x(5)=x(5)+mag;
```

```
            continue
        end
        if angle>=225&&angle<270
            x(6)=x(6)+mag;
            continue
        end
        if angle>=270&&angle<315
            x(7)=x(7)+mag;
            continue
        end
        if angle>=315&&angle<360
            x(8)=x(8)+mag;
            continue
        end
    end
end
[value,index]=max(x);
rotateangle=(index-1)*45;
mask=imrotate(feature,-rotateangle);
[mask_x,mask_y]=size(mask);
if mask_x==patch
    mask=mask((patch+1)/2:(patch*3-1)/2,(patch+1)/2:(patch*3-1)/2);
else
    mask=mask((patch+1)/2:(patch*3-1)/2,(patch+1)/2:(patch*3-1)/2);
end
```

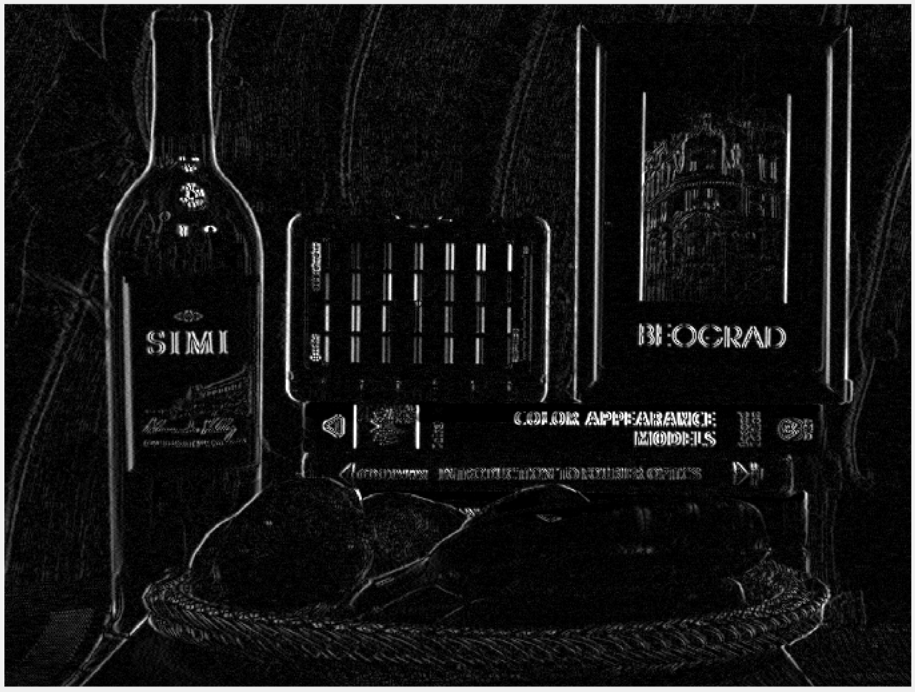
7) Repeat the above for the image ‘Image.bmp’

1) Display the image

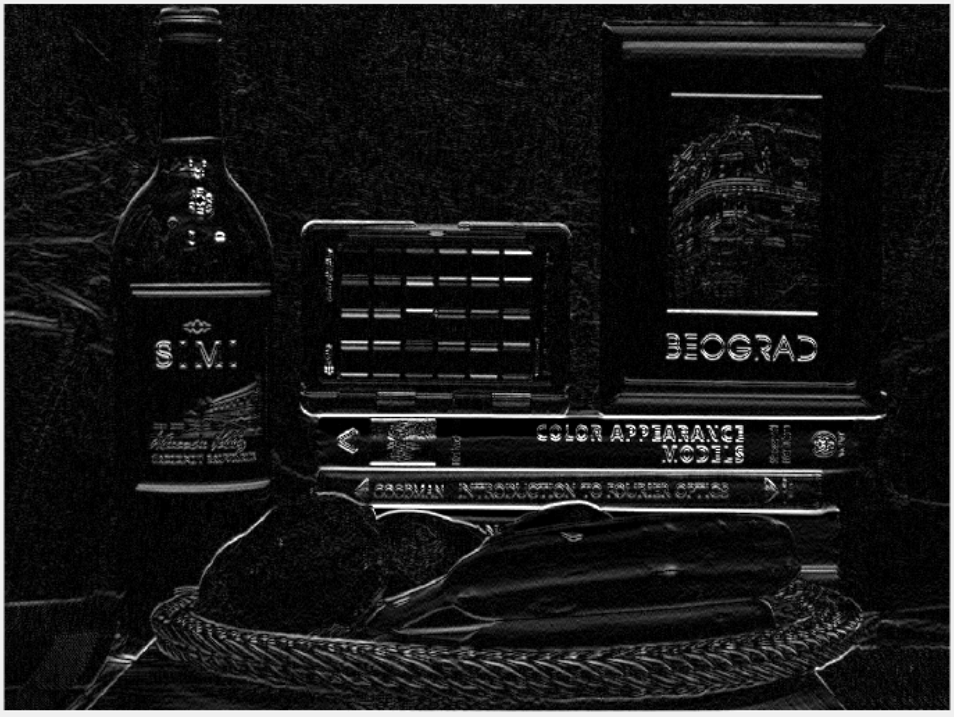
2) Detect the features using Harris detector. Assume that the patches are of size 5x5 pixels.

a) Display the x-derivative, y-derivative.

x-Derivative



y-Derivative



b) Use Forstner-Harris metric to measure usefulness of the features.

Calculate R based on auto-correlated matrix

c) Select features based on the value of local maximums. What threshold did you use?

I define the number of features as Num. Then I sort all the R(:) descend. The R(Num) is the threshold.

When Num=500, Threshold= 9.696344477546471e+07

3) Use adaptive non-maximal suppression to select features.

d) Select same number of features as you had in part 3.

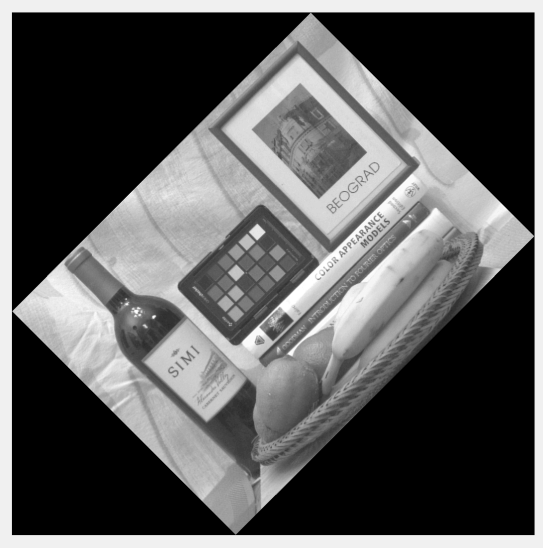
Num=500

e) Display image with marked locations of the selected features.

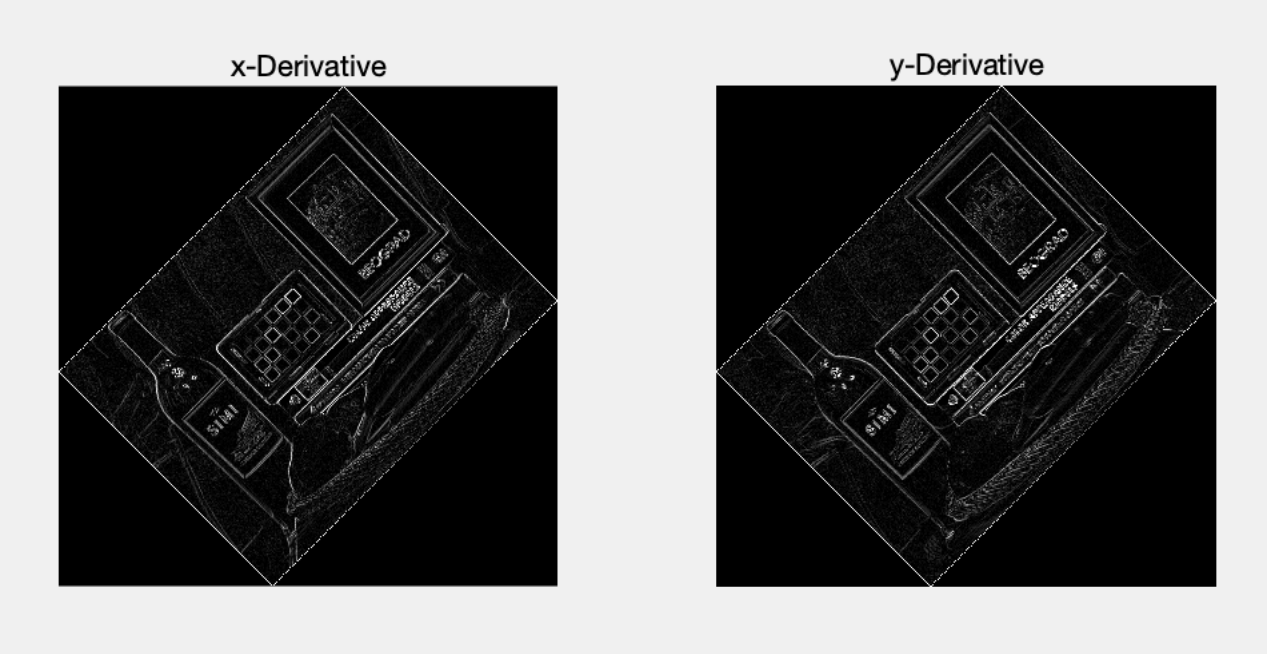
f) Comment on the location of features in part 2 and part 3.

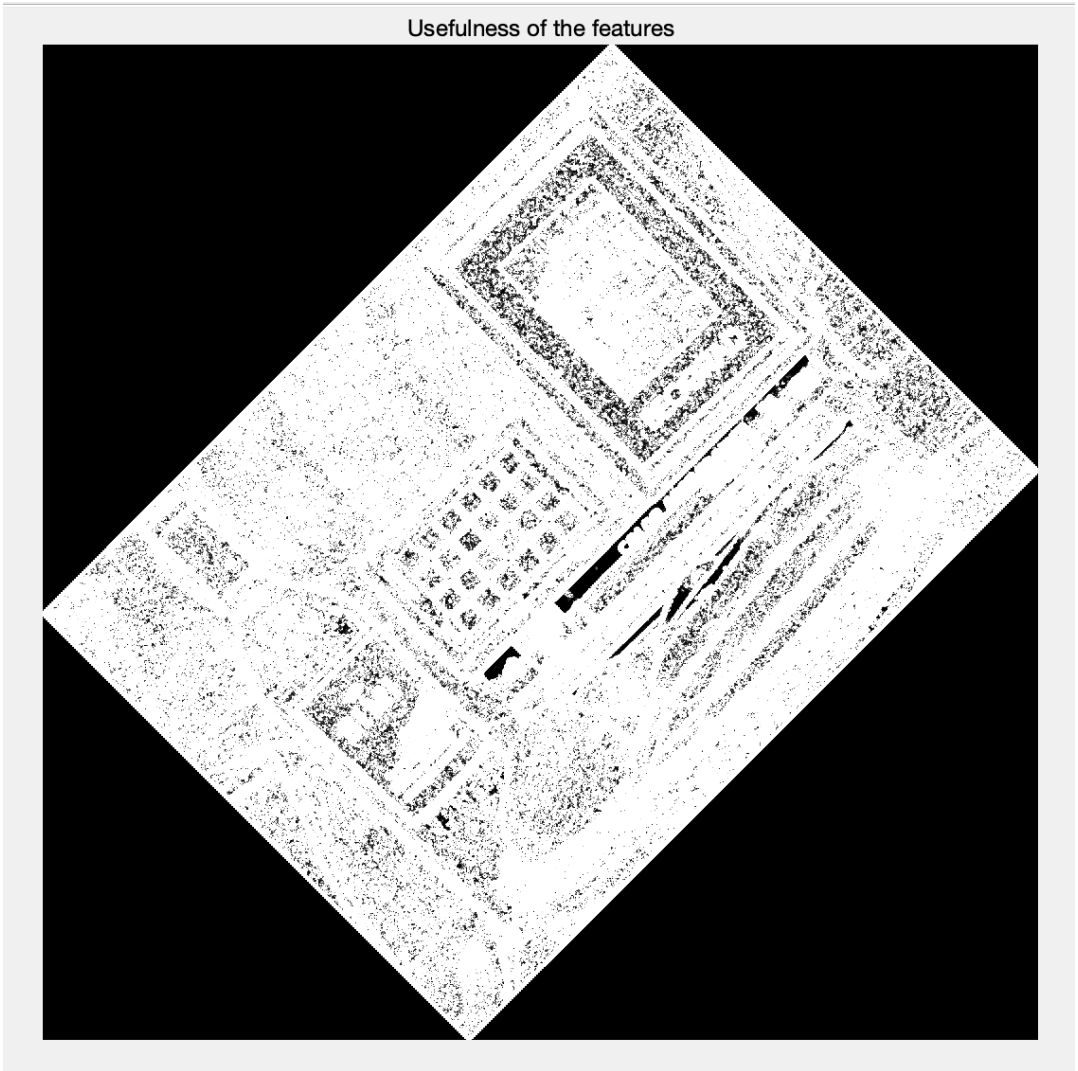
We can easily find out that the features in part 3 is better than in part 2. To be specific, the real feature number found in part 3 is more than in part 2.

4) Rotate the original image by 45 degrees using built-in Matlab function imrotate(). Display the rotated image.

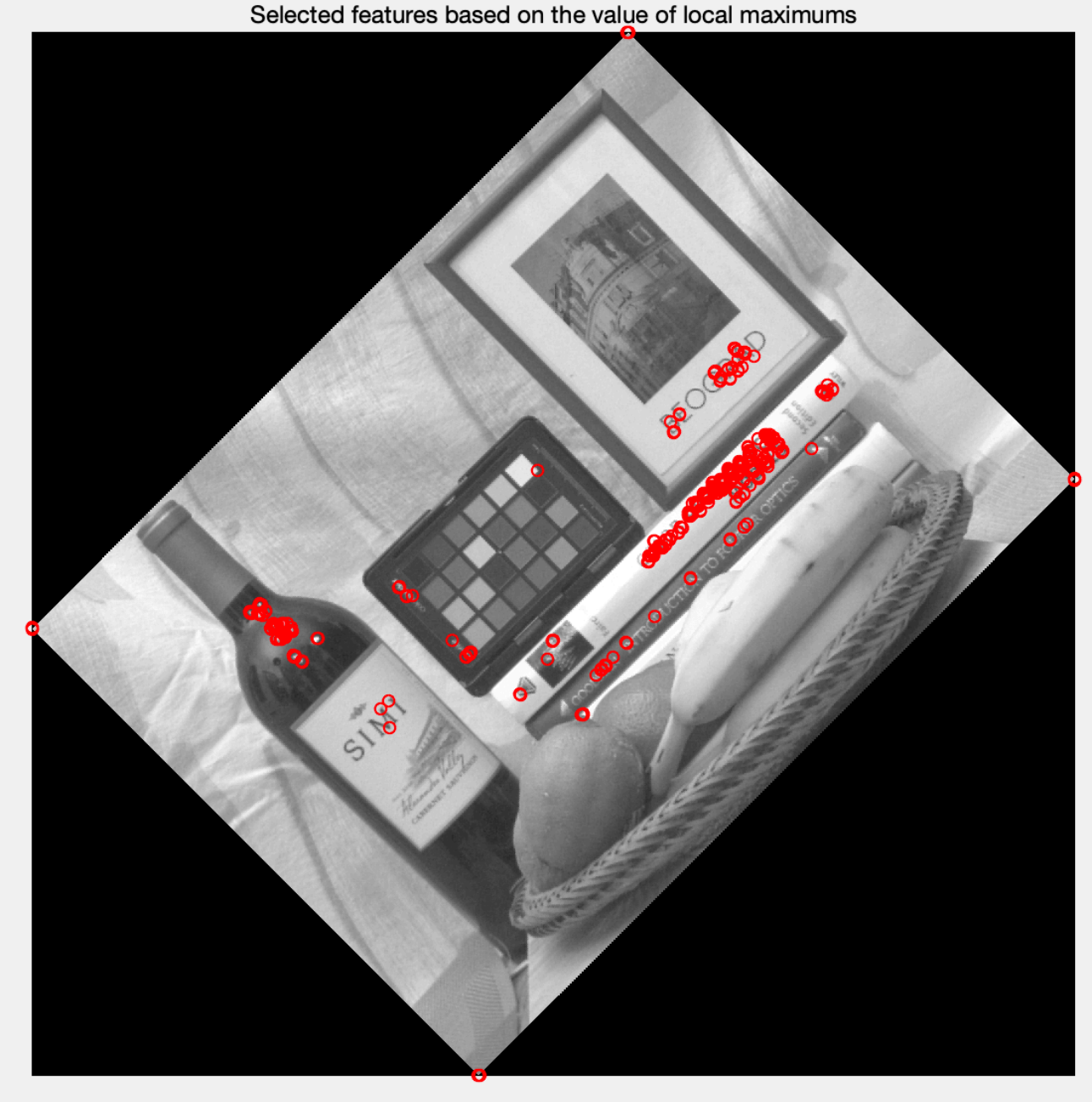


5) Repeat steps 2 and 3 for the rotated image.

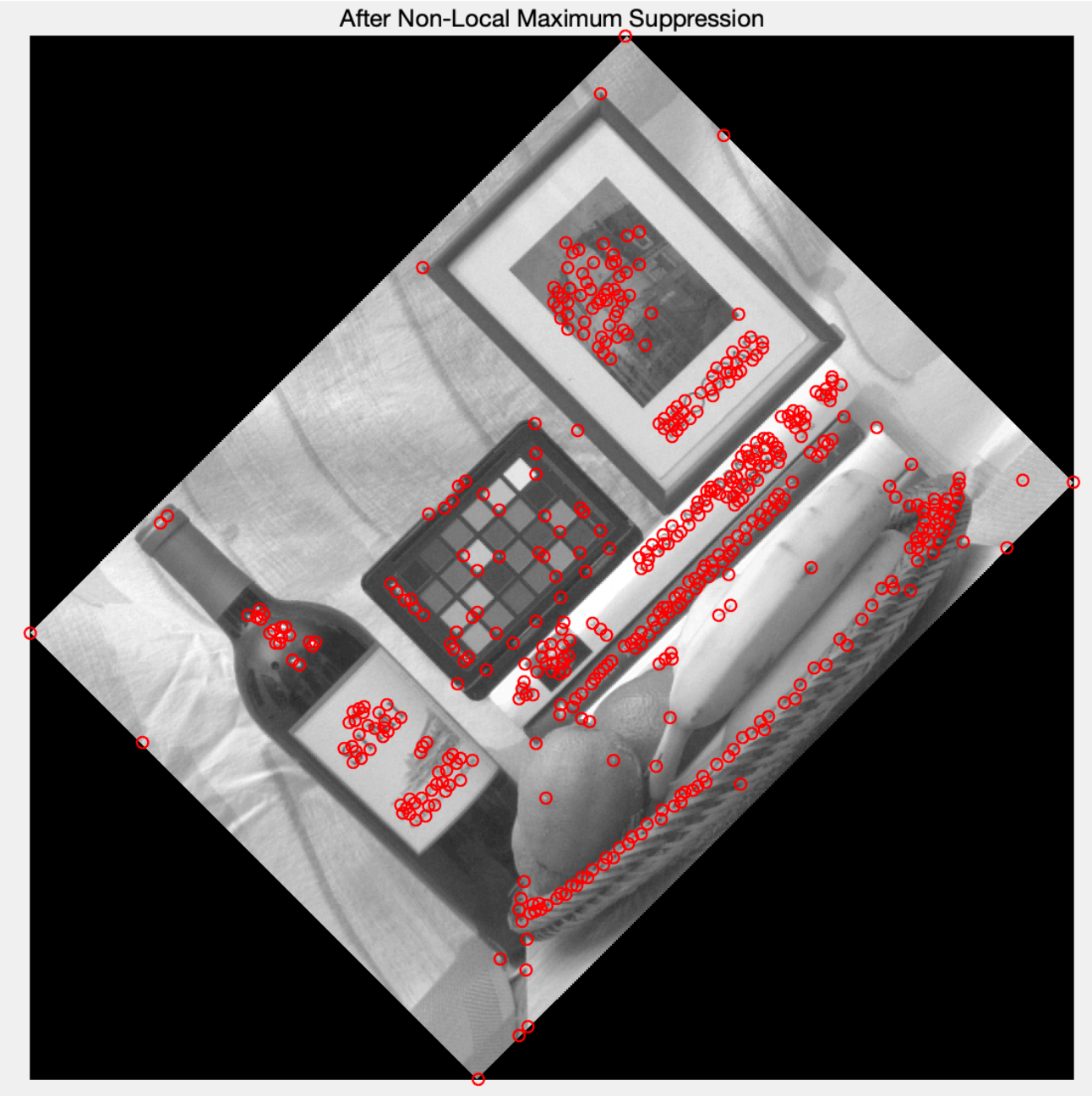




I decide that I select the number of features is 500. Thus the threshold= 8.055512465837336e+07



I define the patch size is 11 in non-local maximum suppression.



6) Match the features found using adaptive non-maximal suppression in the original image to the ones found in the same way in the rotated image. For comparison use sum of squared differences between patches. Keep in mind that before matching two features you have to rotate both of them to align their orientation. How many features were matched? Display the original image with marked locations of the matched features.

I delete the features on the boundary and match the remaining features. I choose the threshold=1.5*e11. 496 features were matched.

If I choose threshold=1*e11, 449 features were match.