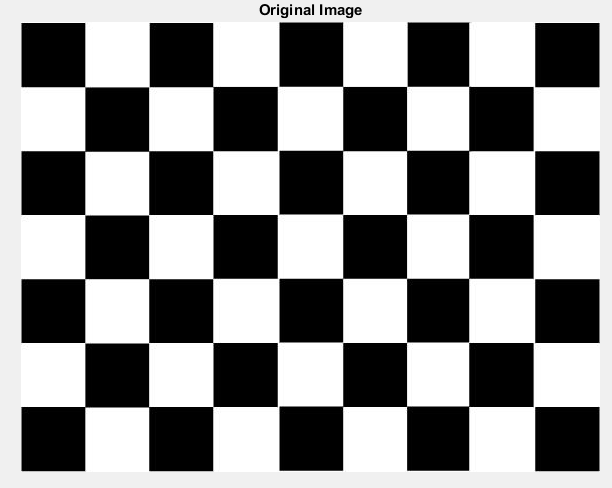
**COMPUTER VISION**

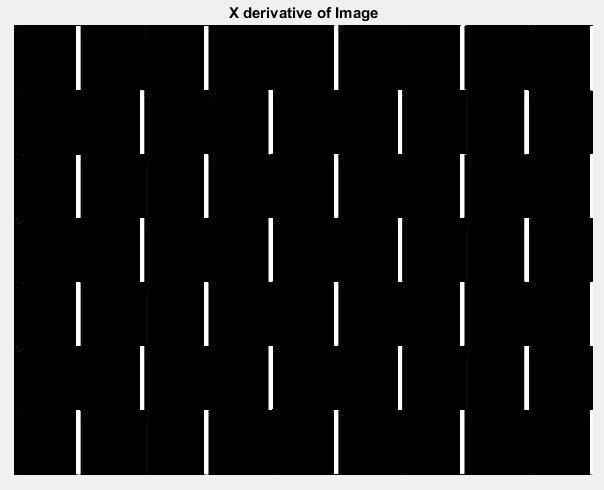
Homework 5

Answer 1

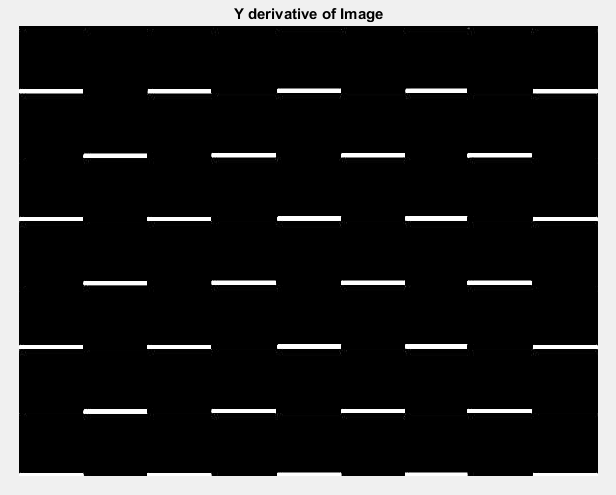
1. The original image is:



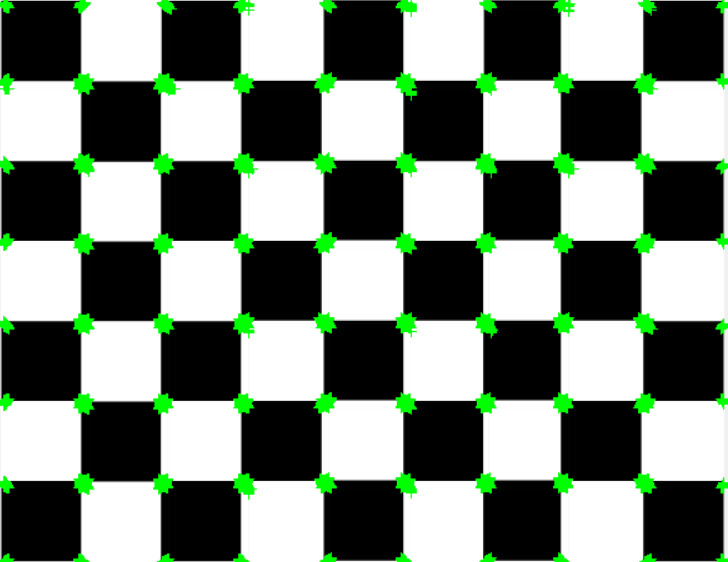
1. X--derivative of the image



Y- derivative of the image

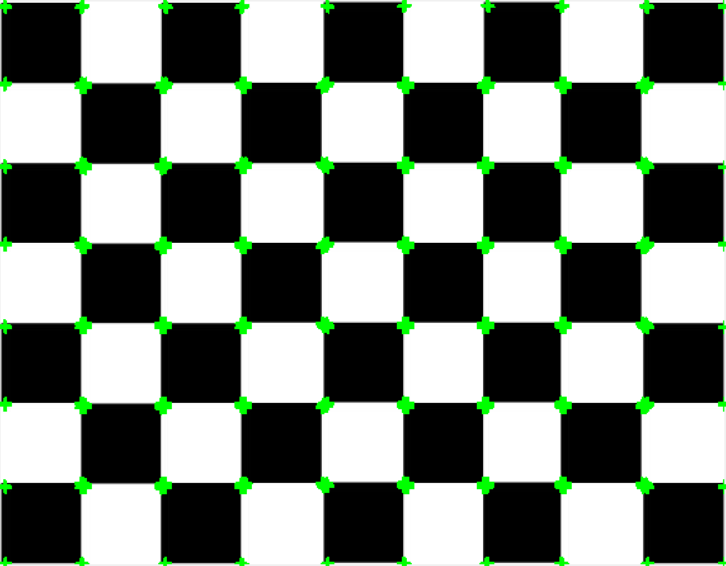


1. Image with marked locations of the selected features

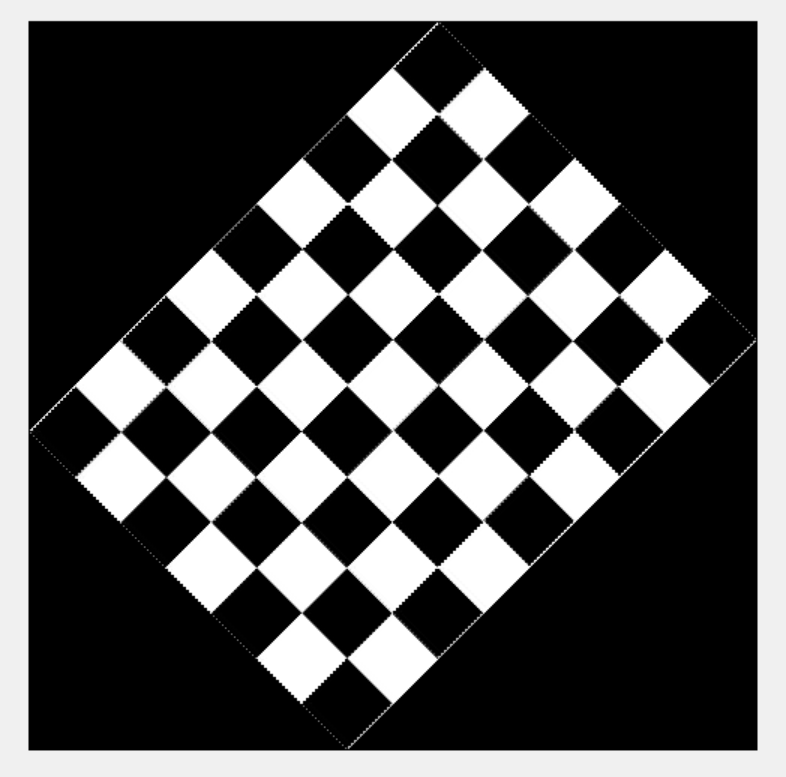


Threshold value is 85000

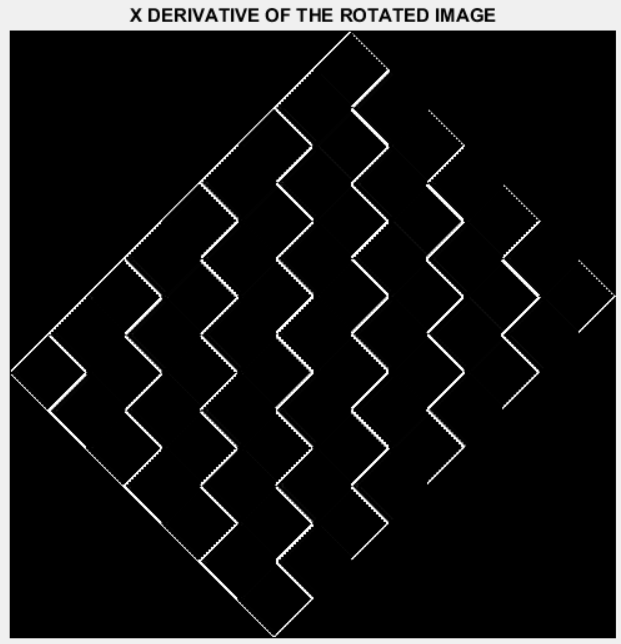
1. After Thresholding, the image with marked locations of the selected features



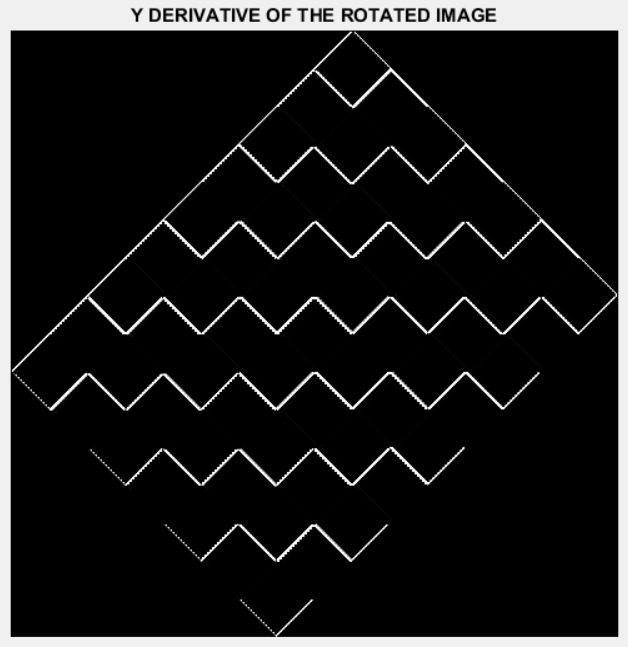
We can say that the adaptive method compares the uneven distribution of image in a better way than the Harris Detector. This is because the Harris Detector uses a threshold value. SO, even if there is an uneven distribution of the features at the corners, it oes feature matching at corners where there is high intensity. This is not the case with adaptive method, as it matches features within a selected radius



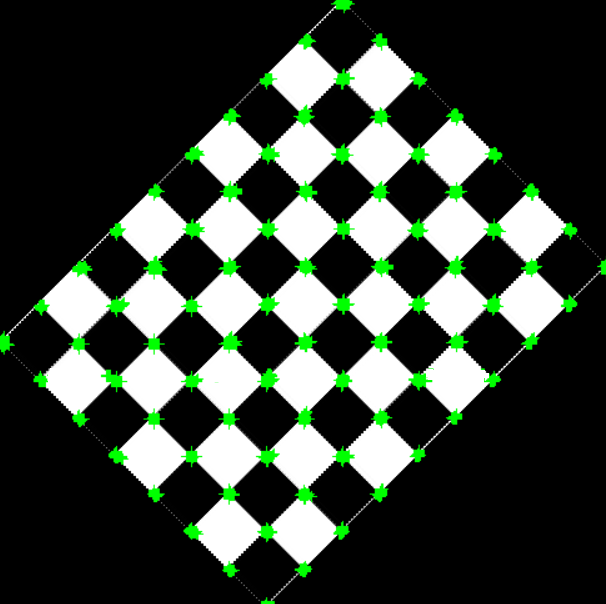
1. X- derivative of the image



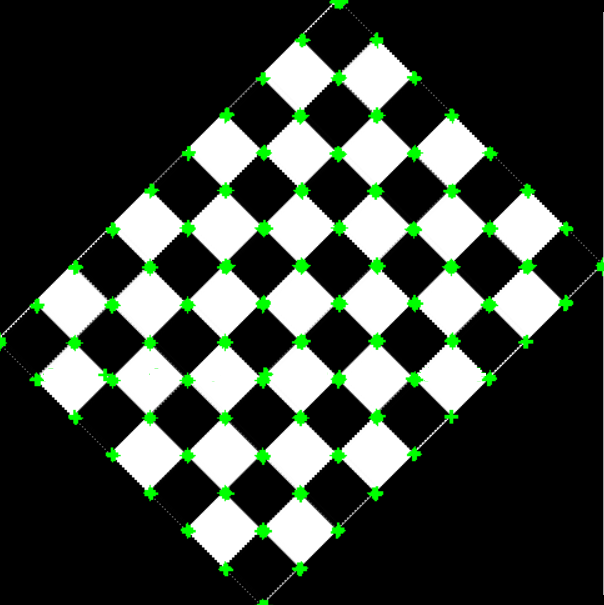
1. Y-derivative of the rotated image



1. Marked locations of the selected features on the rotated image

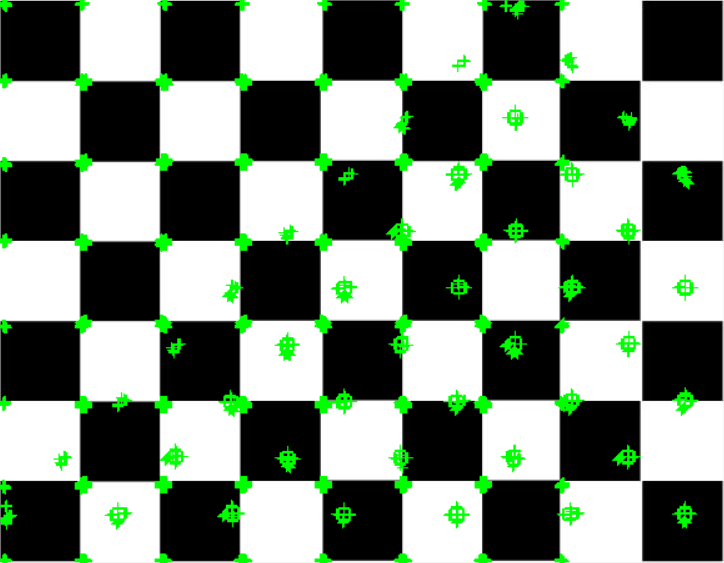


Threshold value = 85000



1. Final image with matched features marked as green dots

A total of 1163 features were matched

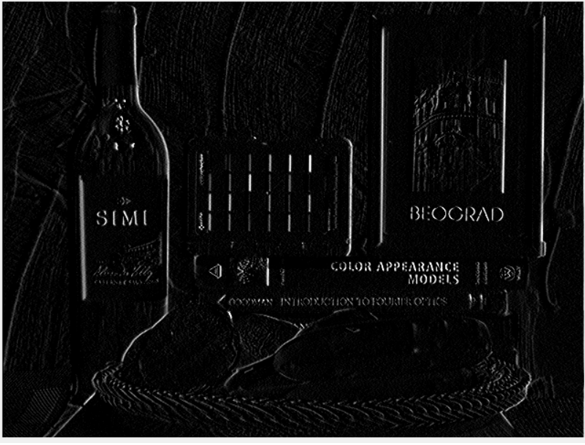


Using the same code and applying the steps to Image.bmp

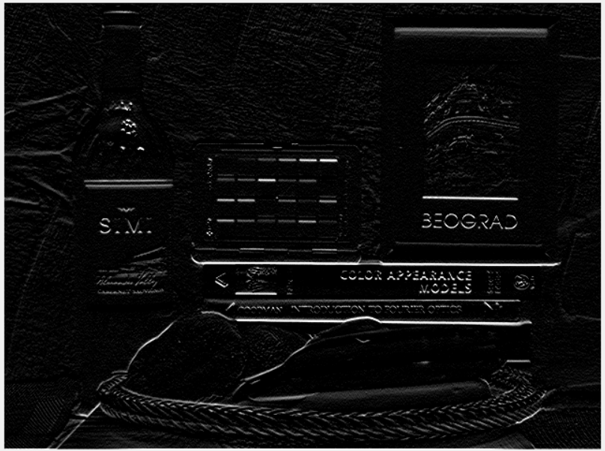
1. Displaying the image



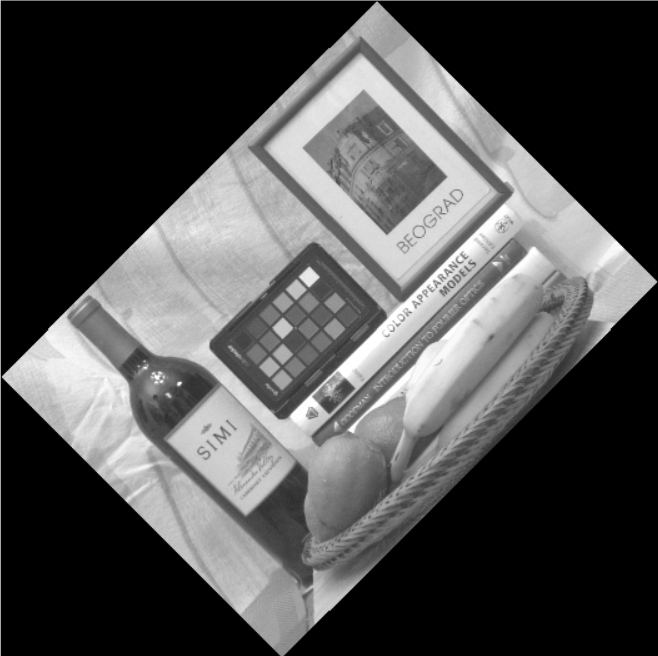
1. X-derivative of the image



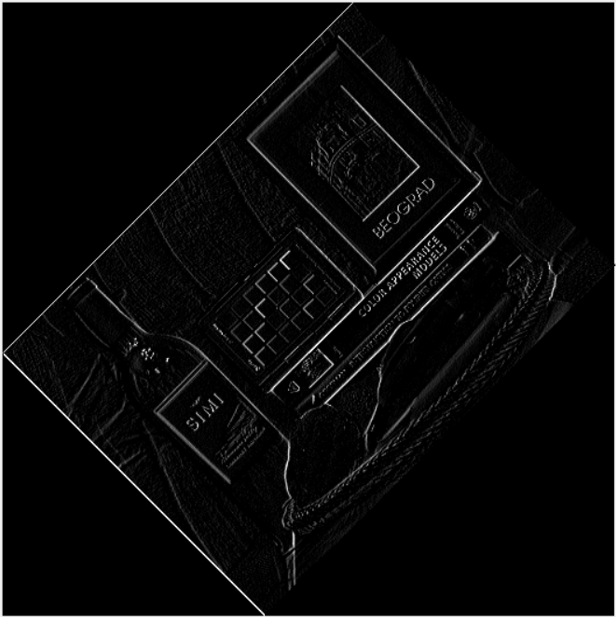
1. Y-derivative of the image



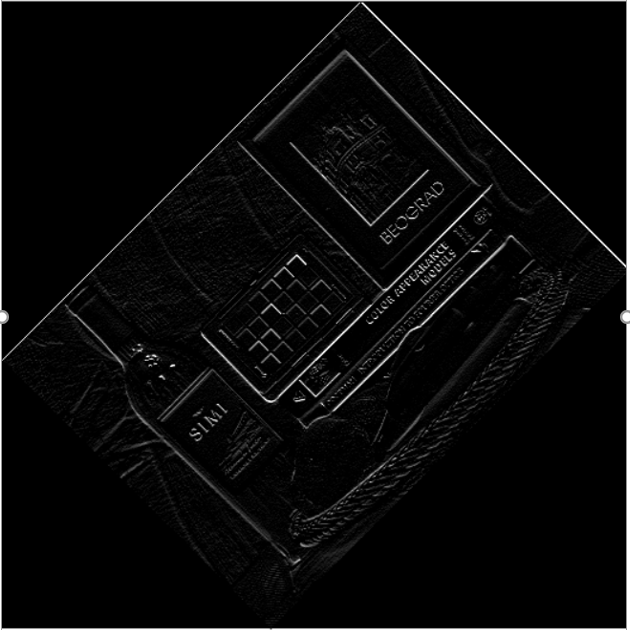
1. Display rotated image



1. X-derivative of the rotated image



1. Y-derivative of the rotated image



Code:

clc

close all

clear all

original\_image = imread('checkerboard.jpg');

x\_derivative = [-2 -1 0 1 2];

y\_derivative = x\_derivative';

original\_image = original\_image(:,:,1);

AxDerivative1 = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

AyDerivative1 = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

AxDerivative3 = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

AyDerivative3 = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

A\_y\_der\_3 = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

base = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

kernel = [0.0183,0.0821,0.1353,0.0821,0.0183;0.0821,0.3679,0.6065,0.3679,0.0821;0.1353,0.6065,1,0.6065,0.1353;0.0821,0.3679,0.6065,0.3670,0.0821;0.0183,0.0821,0.1353,0.0821,0.0183];

figure;

imshow(original\_image)

title('Original Image');

AxDerivative = padarray (original\_image, [1 1], 255);

AxDerivative = padarray (AxDerivative, [1 1], 255);

AyDerivative = padarray (original\_image, [1 1 ], 255);

AyDerivative = padarray (AyDerivative, [1 1 ], 255);

for i = 3 : size(AxDerivative,1) - 2

for j = 3 : size(AxDerivative,2) - 2

temp = double(AxDerivative(i, j-2:j+2));

AxDerivative1(i,j) = temp \* x\_derivative';

end

end

finalAxDerivative = AxDerivative1(3:size(AxDerivative,1)-2,3:size(AxDerivative,2)-2);

figure

imshow(uint8(finalAxDerivative));

title('X derivative of Image');

for i = 3 : size(AyDerivative,1) - 2

for j = 3 : size(AyDerivative,2) - 2

temp = double(AyDerivative(i-2: i+2, j));

AyDerivative1(i,j) = temp' \* y\_derivative;

end

end

finalAyDerivative= AyDerivative1(3:size(AyDerivative,1)-2, 3:size(AyDerivative,2)-2);

figure

imshow(uint8(finalAyDerivative));

title('Y derivative of Image');

A\_x\_der\_2 = AxDerivative1 .^2;

A\_y\_der\_2 = AyDerivative1 .^2;

A\_x\_y\_der = AxDerivative1 .\* AyDerivative1 ;

for i = 3 :size(AxDerivative1, 1) - 2

for j = 3 :size(AxDerivative1, 2) - 2

temp = A\_x\_der\_2(i-2: i+2, j-2: j+2);

AxDerivative3(i, j) = sum(sum(kernel .\* temp));

temp = A\_y\_der\_2(i-2: i+2, j-2: j+2);

A\_y\_der\_3(i, j) = sum(sum(kernel .\* temp));

temp = A\_x\_y\_der(i-2: i+2, j-2: j+2);

AyDerivative3(i, j) = sum(sum(kernel .\* temp));

end

end

for i = 3:size(AxDerivative1, 1) - 2

for j = 3: size(AxDerivative1, 2) - 2

A\_pixel = double([AxDerivative3(i,j) AyDerivative3(i,j) ; AyDerivative3(i,j) A\_y\_der\_3(i, j)]);

base(i, j) = det(A\_pixel) - 0.06 \* ((trace(A\_pixel))^2);

end

end

threshold\_value = 85000;

r = 1;

for i = 3:size(AxDerivative1, 1) - 2

for j = 3:size(AxDerivative1, 2) - 2

if( base (i, j)>threshold\_value)

base1(i,j) = base(i , j);

x\_1(r) = i;

y\_1(r) = j;

r = r+1;

else

base1(i , j) = 0;

end

end

end

figure

imshowpair(original\_image, uint8(base1(3:size(base,1)-2, 3:size(base,2)-2)), 'montage')

title('Original Image and Thresholded Image');

figure

imshow(original\_image);

hold on

plot(y\_1(:,:), x\_1(:,:), '+g')

q=1;

for i=1:size(base1,1)

for j=1:size(base1,2)

if(base1(i,j)~=0)

base1Array(q)=base1(i,j);

x\_2(q)=i;

y\_2(q)=j;

q=q+1;

end

end

end

[R,a]=sort(base1Array,'descend');

for i=1:size(a,2)

x\_3(i)=x\_2(a(i));

y\_3(i)=y\_2(a(i));

end

k=15;

R\_new=0;

for i=2:size(R,2)

minimum\_distance=100;

for j=i-1:-1:1

if(((R(j)-R(i))>(k\*R(i)/100)))

distance=sqrt((x\_3(i)-x\_3(j))^2+((y\_3(i)-y\_3(j))^2));

if (distance<minimum\_distance)

minimum\_distance=distance;

if(R\_new==0)

R\_new(1)=max(max(R));

radius(1)=450\*579;

x\_4(1)=x\_3(1);

y\_4(1)=y\_3(1);

end

R\_new(i)=R(j);

radius(i)=minimum\_distance;

x\_4(i)=x\_3(i);

y\_4(i)=y\_3(i);

end

end

end

end

top\_n=2000;

for i=top\_n:size(R\_new,2)

R\_new(i)=0;

end

final\_matrix=zeros(450,579);

for i=1:size(x\_3,2)

final\_matrix(x\_3(i),y\_3(i))=R\_new(i);

end

r=1;

for i = 1:size(final\_matrix,1)

for j =1:size(final\_matrix,2)

if(final\_matrix(i,j) ~= 0)

x\_5(r,1) = i;

y\_5(r,1) = j;

r = r+1;

end

end

end

figure

imshow(original\_image)

hold on

plot(y\_5(:,:), x\_5(:,:),'+g')

original\_image = imread('checkerboard.jpg');

original\_image=imrotate(original\_image,45);

x\_derivative = [-2 -1 0 1 2];

y\_derivative = x\_derivative';

original\_image = original\_image(:,:,1);

AxDerivative1 = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

AyDerivative1 = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

AxDerivative3 = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

AyDerivative3 = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

A\_y\_der\_3 = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

base = zeros(size(original\_image, 1)+ 4, size(original\_image, 2)+ 4);

kernel = [0.0183,0.0821,0.1353,0.0821,0.0183;0.0821,0.3679,0.6065,0.3679,0.0821;0.1353,0.6065,1,0.6065,0.1353;0.0821,0.3679,0.6065,0.3670,0.0821;0.0183,0.0821,0.1353,0.0821,0.0183];

figure

imshow(original\_image)

AxDerivative = padarray (original\_image, [1 1], 255);

AxDerivative = padarray (AxDerivative, [1 1], 255);

AyDerivative = padarray (original\_image, [1 1 ], 255);

AyDerivative = padarray (AyDerivative, [1 1 ], 255);

for i = 3 : size(AxDerivative,1) - 2

for j = 3 : size(AxDerivative,2) - 2

temp = double(AxDerivative(i, j-2:j+2));

AxDerivative1(i,j) = temp \* x\_derivative';

end

end

finalAxDerivative = AxDerivative1(3:size(AxDerivative,1)-2,3:size(AxDerivative,2)-2);

figure

imshow(uint8(finalAxDerivative));

title('X derivative of Rotated Image');

for i = 3 : size(AyDerivative,1) - 2

for j = 3 : size(AyDerivative,2) - 2

temp = double(AyDerivative(i-2: i+2, j));

AyDerivative1(i,j) = temp' \* y\_derivative;

end

end

finalAyDerivative= AyDerivative1(3:size(AyDerivative,1)-2, 3:size(AyDerivative,2)-2);

figure

imshow(uint8(finalAyDerivative));

title('Y derivative of Rotated Image');

A\_x\_der\_2 = AxDerivative1 .^2;

A\_y\_der\_2 = AyDerivative1 .^2;

A\_x\_y\_der = AxDerivative1 .\* AyDerivative1 ;

for i = 3 :size(AxDerivative1, 1) - 2

for j = 3 :size(AxDerivative1, 2) - 2

temp = A\_x\_der\_2(i-2: i+2, j-2: j+2);

AxDerivative3(i, j) = sum(sum(kernel .\* temp));

temp = A\_y\_der\_2(i-2: i+2, j-2: j+2);

A\_y\_der\_3(i, j) = sum(sum(kernel .\* temp));

temp = A\_x\_y\_der(i-2: i+2, j-2: j+2);

AyDerivative3(i, j) = sum(sum(kernel .\* temp));

end

end

for i = 3:size(AxDerivative1, 1) - 2

for j = 3: size(AxDerivative1, 2) - 2

A\_pixel = double([AxDerivative3(i,j) AyDerivative3(i,j) ; AyDerivative3(i,j) A\_y\_der\_3(i, j)]);

base(i, j) = det(A\_pixel) - 0.06 \* ((trace(A\_pixel))^2);

end

end

threshold\_value = 900000;

r = 1;

for i = 3:size(AxDerivative1, 1) - 2

for j = 3:size(AxDerivative1, 2) - 2

if( base (i, j)>threshold\_value)

base1(i,j) = base(i , j);

x\_1(r) = i;

y\_1(r) = j;

r = r+1;

else

base1(i , j) = 0;

end

end

end

figure

imshowpair(original\_image, uint8(base1(3:size(base,1)-2, 3:size(base,2)-2)), 'montage')

title('Rotated Image and Thresholded Image');

figure

imshow(original\_image)

hold on

plot(y\_1(:,:), x\_1(:,:),'+g')

q=1;

for i=1:size(base1,1)

for j=1:size(base1,2)

if(base1(i,j)~=0)

base1Array(q)=base1(i,j);

x\_2(q)=i;

y\_2(q)=j;

q=q+1;

end

end

end

[R,a]=sort(base1Array,'descend');

for i=1:size(a,2)

x\_3(i)=x\_2(a(i));

y\_3(i)=y\_2(a(i));

end

k=15;

R\_new=0;

for i=2:size(R,2)

minimum\_distance=100;

for j=i-1:-1:1

if(((R(j)-R(i))>(k\*R(i)/100)))

distance=sqrt((x\_3(i)-x\_3(j))^2+((y\_3(i)-y\_3(j))^2));

if (distance<minimum\_distance)

minimum\_distance=distance;

if(R\_new==0)

R\_new(1)=max(max(R));

radius(1)=450\*579;

x\_4(1)=x\_3(1);

y\_4(1)=y\_3(1);

end

R\_new(i)=R(j);

radius(i)=minimum\_distance;

x\_4(i)=x\_3(i);

y\_4(i)=y\_3(i);

end

end

end

end

top\_n=2000;

for i=top\_n:size(R\_new,2)

R\_new(i)=0;

end

final\_mat\_rot=zeros(729,729);

for i=1:size(x\_3,2)

final\_mat\_rot(x\_3(i),y\_3(i))=R\_new(i);

end

r=1;

for i = 1:size(final\_mat\_rot,1)

for j =1:size(final\_mat\_rot,2)

if(final\_mat\_rot(i,j) ~= 0)

x\_5(r,1) = i;

y\_5(r,1) = j;

r = r+1;

end

end

end

figure

imshow(original\_image)

hold on

plot(y\_5(:,:), x\_5(:,:),'+g')

c=1;

for i=1:size(final\_matrix,1)

for j=1:size(final\_matrix,2)

if(final\_matrix(i,j)~=0)

array\_without\_rot(c)=final\_matrix(i,j);

x\_2(c)=i;

y\_2(c)=j;

c=c+1;

end

end

end

[R,a]=sort(array\_without\_rot,'descend');

for i=1:size(a,2)

x\_3(i)=x\_2(a(i));

y\_3(i)=y\_2(a(i));

end

c=1;

for i=1:size(final\_mat\_rot,1)

for j=1:size(final\_mat\_rot,2)

if(final\_mat\_rot(i,j)~=0)

array\_with\_rot(c)=final\_mat\_rot(i,j);

x\_cord\_rot(c)=i;

y\_cord\_rot(c)=j;

c=c+1;

end

end

end

[R\_rot,a\_rot]=sort(array\_with\_rot,'descend');

for i=1:size(a\_rot,2)

x\_cord\_rot\_new(i)=x\_cord\_rot(a\_rot(i));

y\_cord\_rot\_new(i)=y\_cord\_rot(a\_rot(i));

end

count=0;

for i=1:size(R,2)

for j=1:size(R,2)

if(sqrt((x\_3(i)-x\_cord\_rot\_new(j))^2+(y\_3(i)-y\_cord\_rot\_new(j))^2)<3)

count=count+1;

end

end

end

toc

tt=toc-tic;

original\_image = imread('checkerboard.jpg');

figure;imshow(original\_image);

hold on;

plot(x\_3, y\_3, '+g');