מעבדה 2 – Hough וזיהוי פינות

(1) גילוי קוים ישרים – lab2b

close ALL

%x3=imread('septagon.tif') ;

x3=imread('hex.jpg') ;

x=x3(:,:,1) ;

img=edge(x,'canny',0.4) ;

[acc,theta,r]=hough(img,'RhoResolution',2,'Theta',-90:89);

NUMPEAKS=10 ;

PEAKS = houghpeaks(acc,NUMPEAKS,'Threshold',80,'NHoodSize',[9 9]) ;

[m,n]=size(PEAKS);

for i=1:m % marking peaks

PEAKS(i,3)=acc(PEAKS(i,1),PEAKS(i,2))

acc(PEAKS(i,1)-1:PEAKS(i,1)+1,PEAKS(i,2)-1:PEAKS(i,2)+1)=64 ;

end

figure, imshow(x)

figure, imshow(img)

figure, imshow(acc,[],'XData',theta,'YData',r,...

'InitialMagnification','fit');

xlabel('\theta'), ylabel('\rho');

axis on, axis normal ;

x1 = theta(PEAKS(:,2));

y1 = r(PEAKS(:,1));

figure, plot(x1,y1,'s','color','black');

חזור והרץ את הקוד עם התמונה מהקובץ hex.jpg . שנה את הסף בפונקציה houghpeaks ל-80

(2) (א) – מציאת מעגלים עם תצוגה של מערך ההתמרה –lab2d

close ALL

%x3=imread('coins.png') ;

x3=imread('coins.jpg') ;

x=x3(:,:,1) ;

img=edge(x,'canny',0.4) ;

%[centers, radii, metric] = imfindcircles(img,[20 50]);

[acc, centers, radii] = CircularHough\_Grd(x,[20 50],10,20);

centers=uint16(centers) ;

[centers radii]

[m,n]=size(centers)

for i=1:m

img(centers(i,2)-1:centers(i,2)+1,centers(i,1)-1:centers(i,1)+1)=255 ;

end

figure, imshow(x,'InitialMagnification','fit');

figure, imshow(img,'InitialMagnification','fit');

figure, imshow(acc,[],'InitialMagnification','fit');

%figure(2); imagesc(acc); colormap('gray'); axis image;

% hold on;

% plot(circen(:,1), circen(:,2), 'r+');

% for k = 1 : size(circen, 1),

% DrawCircle(circen(k,1), circen(k,2), cirrad(k), 32, 'b-');

% end

% hold off;

% title(['Raw Image with Circles Detected ', ...

% '(center positions and radii marked)']);

הרץ את הקוד גם עם התמונה coins.png

(ב) מציאת מעגלים (ללא מערך התמרה כמוצא) – lab2c

close ALL

x3=imread('coins.png') ;

x=x3(:,:,1) ;

img=edge(x,'canny',0.4) ;

[centers, radii, metric] = imfindcircles(img,[20 50]);

[centers radii]

centers=round(centers) ;

[m,n]=size(centers) ;

for i=1:m

img(centers(i,2),centers(i,1))=255 ;

end

figure, imshow(x,'InitialMagnification','fit');

figure, imshow(img,'InitialMagnification','fit');

% axis on, axis normal ;

(3)(א) מציאת פינות בשיטת Moravec : -lab2e

clear

close ALL

f=double(imread('sq.bmp')) ;

[m, n] = size(f) ;

for i = 2:m-1

for j = 2:n-1

acc=0 ;

for k=-1:1

for l=-1:1

acc= acc+abs(f(i,j)-f(i-k,j-l)) ;

end

end

acc=acc/8 ;

g(i,j) = acc ;

end

end

g1=g ;

mx=max(max(g)) ;

g2=(g==mx) ;

n\_corners=sum(sum(g2))

nc=sum(sum(g2))

for i = 2:m-1

for j = 2:n-1

if g(i,j) == mx

g(i,j)=0 ;

g1(i-1:i+1,j-1:j+1)=255 ;

end

end

end

figure(1)

subplot(221), image(255\*f)

colormap(gray(256))

subplot(222),image(255\*g)

colormap(gray(256))

figure(2)

image(g1)

colormap(gray(256))

(ב) מציאת פינות בשיטת Harris :

clear

close ALL

I = imread('hex.bmp');

corners = detectHarrisFeatures(I);

figure, imshow(I); hold on;

plot(corners.selectStrongest(6)); % keep 6 strongest corners

C=corner(I,6) ;

figure, imshow(I); hold on

plot(C(:,1), C(:,2), 'r\*');

(ג) מציאת פינות

I = imread('cameraman.tif');

**% Find and extract corner features.**

corners = detectHarrisFeatures(I);

[features, valid\_corners] = extractFeatures(I, corners);

figure; imshow(I); hold on

plot(valid\_corners);

(ד) ממוש אלגוריתם Harris למציאת פינות – detaile\_harris\_corners

close ALL

%img = imread('septagon.tif');

img = imread('hex.jpg');

[ x, y, scores, Ix, Iy ] = harris\_corners( img );

figure, imshow(img)

hold on

for i = 1:scores

plot(x(i), y(i), 'r+');

end

hold off

function [ x, y, scores, Ix, Iy ] = harris\_corners(img)

% HARRIS\_CORNERS Extracts points with a high degree of 'cornerness' from

% RGB image matrix of type uint8

% Input - image = NxMx3 RGB image matrix

% Output - x = nx1 vector denoting the x location of each of n

% detected corners

% y = nx1 vector denoting the y location of each of n

% detected corners

% scores = number of corners found

% Ix = A matrix with the same number of rows and columns as the

% input image, storing the gradients in the x-direction at each pixel

% Iy = A matrix with the same nuimber of rwos and columns as the

% input image, storing the gradients in the y-direction at each pixel

G = rgb2gray(img);

% convert to double

G2 = im2double(G);

% create X and Y Sobel filters

horizontal\_filter = [1 0 -1; 2 0 -2; 1 0 -1];

vertical\_filter = [1 2 1; 0 0 0 ; -1 -2 -1];

% using imfilter to get our gradient in each direction

filtered\_x = imfilter(G2, horizontal\_filter);

filtered\_y = imfilter(G2, vertical\_filter);

% store the values in our output variables, for clarity

Ix = filtered\_x;

Iy = filtered\_y;

f = fspecial('gaussian');

Ix2 = imfilter(Ix.^2, f);

Iy2 = imfilter(Iy.^2, f);

Ixy = imfilter(Ix.\*Iy, f);

% set empirical constant between 0.04-0.06

k = 0.04;

num\_rows = size(img,1);

num\_cols = size(img,2);

% create a matrix to hold the Harris values

H = zeros(num\_rows, num\_cols);

% % get our matrix M for each pixel

for y = 6:size(img,1)-6 % avoid edges

for x = 6:size(img,2)-6 % avoid edges

Ix2\_matrix = Ix2(y-2:y+2,x-2:x+2);

Ix2\_mean = sum(Ix2\_matrix(:));

% Iy2 mean

Iy2\_matrix = Iy2(y-2:y+2,x-2:x+2);

Iy2\_mean = sum(Iy2\_matrix(:));

% Ixy mean

Ixy\_matrix = Ixy(y-2:y+2,x-2:x+2);

Ixy\_mean = sum(Ixy\_matrix(:));

% compute R, using te matrix we just created

Matrix = [Ix2\_mean, Ixy\_mean;

Ixy\_mean, Iy2\_mean];

R1 = det(Matrix) - (k \* trace(Matrix)^2);

% store the R values in our Harris Matrix

H(y,x) = R1;

end

end

% set threshold of 'cornerness' relative to the average R score

avg\_r = mean(mean(H));

threshold = abs(100 \* avg\_r);

[row, col] = find(H > threshold);

y = row;

x = col;

scores=length(x)

end