```
% Compattezza


%% Reset

clear
close all

%% Legge immagine

info = imfinfo('pillsetc.png')

MyImage = imread('pillsetc.png');
figure('Name','MyImage')
title('MyImage')
imshow(MyImage)

imwrite(MyImage,'MyImage.png','png')

%% Converte in scala di grigi

GrayImage = rgb2gray(MyImage);
figure('Name','GrayImage')
title('GrayImage')
imshow(GrayImage)

imwrite(GrayImage,'GrayImage.png','png')

%% Miglioramento

GrayImageAdj = imadjust(GrayImage,[0.2 1]);
figure('Name','GrayImageAdj')
title('imadjust')
imshow(GrayImageAdj)

imwrite(GrayImageAdj,'GrayImageAdj.png','png')

% %%
% GrayImageAdj = histeq (GrayImageAdj);
% figure('Name','GrayImageAdj')
% title('histeq')
% imshow(GrayImageAdj)
%
% %%
% GrayImageAdj=adapthisteq(GrayImageAdj,'NumTiles',[3 3],'ClipLimit',0.1);
% figure('Name','GrayImageAdj')
% title('adapthisteq')
% imshow(GrayImageAdj)

%% Calcola soglia e converte in bianco e nero

threshold = graythresh(GrayImageAdj);
BWImage=im2bw(GrayImageAdj,threshold);
figure('Name','BWImage')
title('BWImage')
imshow(BWImage)

imwrite(BWImage,'BWImage.png','png')
```

```
%% Converte in negativo se necessario per avere sfondo nero

if mean2(BWImage) > 0.5
    BlackBackgrBWImage = imgnegative (BWImage); % BlackBackgrBWImage =
~BWImage;
else
    BlackBackgrBWImage = BWImage;
end

figure('Name','BlackBackgrBWImage')
title('BlackBackgrBWImage')
imshow(BlackBackgrBWImage)

imwrite(BlackBackgrBWImage,'BlackBackgrBWImage.png','png')

%% Rimuove tutti gli oggetti contenenti meno di 50 pixel

BlackBackgrBWImageMod1 = bwareaopen(BlackBackgrBWImage,50);

figure('Name','BlackBackgrBWImageMod1 (bwareopen)')
title('BlackBackgrBWImageMod1')
imshow(BlackBackgrBWImageMod1)

imwrite(BlackBackgrBWImageMod1,'BlackBackgrBWImageMod1 (bwareopen).png','png')

%% Chiusura morfologica

se = strel('disk',2);
BlackBackgrBWImageMod2 = imclose(BlackBackgrBWImageMod1,se);

figure('Name','BlackBackgrBWImageMod2 (imclose)')
title('BlackBackgrBWImageMod2')
imshow(BlackBackgrBWImageMod2)

imwrite(BlackBackgrBWImageMod2,'BlackBackgrBWImageMod2 (imclose).png','png')

%% Riempimento dell' interno

reply = '';
while ~strcmpi(reply,'i') && ~strcmpi(reply,'o')
    reply = input('Want you consider inner (holes) or outer contours? I/O [O
default]: ','s');
    if isempty(reply)
        reply = 'O';
    end
end

if reply == 'O' || reply == 'o'
    BlackBackgrBWImageMod3 = imfill(BlackBackgrBWImageMod2,'holes');
else
    BlackBackgrBWImageMod3 = BlackBackgrBWImageMod2;
end

figure('Name','BlackBackgrBWImageMod3 (imfill)')
title('BlackBackgrBWImageMod3')
imshow(BlackBackgrBWImageMod3)
```

```
imwrite(BlackBackgrBWImageMod3,'BlackBackgrBWImageMod3 (imfill).png','png')

%% Trova i contorni

if reply == 'O' || reply == 'o'
    [B,L] = bwboundaries(BlackBackgrBWImageMod3,'noholes');
else
    [B,L] = bwboundaries(BlackBackgrBWImageMod3);
end
LabelImage=label2rgb(L);
figure('Name','Figura finale')
title('LabelImage')
imshow(LabelImage)
axis image

imwrite(LabelImage,'LabelImage.png','png');

%% Traccia i contorni

hold on
for k = 1:length(B)
    boundary = B{k};
    plot(boundary(:,2), boundary(:,1), 'y', 'LineWidth', 2)
end
title('ContourImage')

print -dpng -r300 Contour

%% Determina quali oggetti sono rotondi

stats = regionprops(L,'Area','Centroid','EquivDiameter','Perimeter'); % stats è
un vettore di struct
threshold = 0.88;

for k = 1:length(B) % length(B)= no. oggetti

    boundary = B{k}; % boundary matrix dim = length(B{k})*2

    % Perimetro
    if reply == 'O' || reply == 'o'
        delta_sq = diff(boundary).^2; % .^ potenza elem per elem
        perimeter = sum(sqrt(sum(delta_sq,2))); % sum (,2) somma righe
        corr = sqrt((boundary(1,2)-boundary(length(boundary),2))^2 +
(boundary(1,1)-boundary(length(boundary),1))^2);
        perimeter = perimeter + corr;
    else
        perimeter = stats(k).Perimeter;
    end

    % Area
    area = stats(k).Area;

    % Metrica
    metric = 4*pi*area/perimeter^2;

    % Centro
    centroid = stats(k).Centroid;
```

```
    % Raggio
    radius = stats(k).EquivDiameter/2;

    % Risultati
    metric_string = sprintf('%2.2f',metric);

    center_string = sprintf('Centro: (%3.2f,%3.2f)',centroid);

    radius_string = sprintf('Raggio: %3.2f',radius);

    string = {center_string radius_string};

    % Segna oggetti rotondi
    if metric > threshold
        centroid = stats(k).Centroid;
        plot(centroid(1),centroid(2),'ko');
        text(centroid(1),centroid(2)+16,string,'HorizontalAlignment','center','
Color','k',...
            'FontSize',6,'FontWeight','bold');
    end

    text(boundary(1,2)-20,boundary(1,1)-10,metric_string,'Color','k',...
        'FontSize',8,'FontWeight','bold');

end

title('Metrics closer to 1 indicate that the object is approximately round')
print -dpng -r300 Contour2

axis on
print -dpng -r300 Contour3

%% Mostra tutti i passaggi

scrsz = get(0,'ScreenSize');
figure('Name','Sintesi','Position',scrsz);
subplot(3,4,1); imshow (MyImage);
subplot(3,4,2); imshow (GrayImage);
subplot(3,4,3); imshow (GrayImageAdj);
subplot(3,4,4); imshow (BWImage);
subplot(3,4,5); imshow (BlackBackgrBWImage);
subplot(3,4,6); imshow (BlackBackgrBWImageMod1);
subplot(3,4,7); imshow (BlackBackgrBWImageMod2);
subplot(3,4,8); imshow (BlackBackgrBWImageMod3);
subplot(3,4,9); imshow (LabelImage);
subplot(3,4,10); imshow ('Contour.png');
subplot(3,4,11); imshow ('Contour2.png');
subplot(3,4,12); imshow ('Contour3.png');

print -dpng -r900 Sintesi

% %% Chiude finestre
%
% disp('Premere un tasto per chiudere tutte le finestre');
% pause;
% close all;
```

```
%% Trasformata di Hough


%% Reset

clear
close all

%% Dimensioni immagine

Xmax = 100;
Ymax = 50;

%% Circle

X1=75; % X<Xmax-R
Y1=30;  % Y<Ymax-R
R1=15;  % R<min(Xmax,Ymax)

X2=25; % X<Xmax-R
Y2=15;  % Y<Ymax-R
R2=5;  % R<min(Xmax,Ymax)

%Circles
theta = 0:(2*pi)/100:2*pi;

t1 = X1 + R1*cos(theta);
u1 = Y1 + R1*sin(theta);

t2 = X2 + R2*cos(theta);
u2 = Y2 + R2*sin(theta);

% Square
t3 = [35*ones(1,10) 35:45 45*ones(1,10) 45:-1:35];
u3 = [25:35 35*ones(1,10) 35:-1:25 25*ones(1,10)];

figure ('Name','Hough Transform')
plot (t1,u1,'ko',t2,u2,'ko',t3,u3,'ko')
title('Image')

axis equal

print -dpng -r300 Start

B = [[u1;t1]';[u2;t2]';[u3;t3]'];

%% Hough

[XYR,time1,A] = houghfunc(Xmax,Ymax,B);

%%

tic

hold on
theta = 0:(2*pi)/100:2*pi;
for i=1:size(XYR,1)
    if XYR(i,1)~=0 && XYR(i,2)~=0 && XYR(i,3)~=0
```

```
        x = XYR(i,1)+XYR(i,3)*cos(theta);
        y = XYR(i,2)+XYR(i,3)*sin(theta);
        plot (x,y,'r','LineWidth',2);
        center_string = sprintf('Centro: (%u,%u)',[XYR(i,1) XYR(i,2)]);
        radius_string = sprintf('Raggio: %u',XYR(i,3));
        string = {center_string radius_string};
        plot(XYR(i,1),XYR(i,2),'ko');
        text(XYR(i,1),XYR(i,2)-
(4/3)*XYR(i,3),string,'HorizontalAlignment','center','Color','k',...
            'FontSize',6,'FontWeight','bold');
        XYR (i,1)
        XYR (i,2)
        XYR (i,3)
    end
end

time2=toc;

time1
time2

title('Hough Transform')
print -dpng -r300 Result
```

```
function [XYR,time,A] = houghfunc(Xmax,Ymax,B)

%% Parameters

maxvalue = 25 % soglia ricerca massimo nella matrice A
n = 5; % soglia tolleranza cerchio r = r +- n
minpoints = 20 % numero minimo di punti che devo trovarsi nella corona
circolare per potere considerarli appartenenti a un cerchi

%% Hough Transform

tic

maxc1 = Xmax;
maxc2 = Ymax;
maxc3 = min(Xmax,Ymax)/2;

A = zeros (maxc1,maxc2,maxc3,'uint32');

boundary_row_num = size(B,1);

for p = 1:boundary_row_num

    p
    xi = B(p,2);
    yi = B(p,1);

    for c1 = 1:maxc1
        for c2 = 1:maxc2

            formula = round(sqrt((xi-c1)^2+(yi-c2)^2));

            if formula<=maxc3 && formula>=1
                c3 = formula;
                A(c1,c2,c3)=A(c1,c2,c3)+1;

            end
        end
    end
end

%% Find maxima

XYR=zeros(10,3);

p = 1

while max(A(:)) >= maxvalue

    maximum = max(A(:));
    count = 0;

    for i = 1:maxc1
        for j = 1:maxc2
            for k = 1:maxc3
                if A(i,j,k) == maximum && count == 0
                    XYR(p,1) = i;
                    XYR(p,2) = j;
```

```
                    XYR(p,3) = k;
                    count = 1;
                end
            end
        end
    end

    ivect=max(1,XYR(p,1)-XYR(p,3)):min(XYR(p,1)+XYR(p,3),maxc1);
    jvect=max(1,XYR(p,2)-XYR(p,3)):min(XYR(p,2)+XYR(p,3),maxc2);

    for i=ivect
        for j=jvect
            for k=1:maxc3
                if sqrt((XYR(p,1)-i)^2+(XYR(p,2)-j)^2) <= XYR(p,3)
                    A(i,j,k)=0;
                end
            end
        end
    end

    p = p + 1

end

%% Delete false cirlces

XYRrow=size(XYR,1);
k1=0;
for i=1:XYRrow
    if XYR(i,1)~=0 && XYR(i,2)~=0 && XYR(i,3)~=0
        k1=k1+1;
    end
end

for p = 1:k1

    count = 0;
    for i = 1:size(B,1)
        xi = B(i,2);
        yi = B(i,1);
        if (xi-XYR(p,1))^2 + (yi-XYR(p,2))^2 <= XYR(p,3)^2+n && (xi-XYR(p,1))^2
+ (yi-XYR(p,2))^2 >= XYR(p,3)^2-n
            count = count + 1;
        end
    end

    if count < minpoints
        XYR(p,1)=0;
        XYR(p,2)=0;
        XYR(p,3)=0;
    end

end

time=toc;
```