```
%%Visione artificiale

clearvars
close all
clc
syms x diag1(x) diag2(x)

%% Lettura immagine
% imgRGB=imread('triangolo.jpg');
% imgRGB=imread('rondella.jpg');
imgRGB=imread('bottiglia.jpg');

prefRes=[480,640];
width = size(imgRGB, 2);    % Larghezza iniziale dell'immagine
height = size(imgRGB, 1);   % Altezza iniziale dell'immagine
%% Ridimensionamento Immagine
if (height > prefRes(1))
    imgRGB = imresize(imgRGB, [prefRes(1) NaN]);
end
if (width > prefRes(2))
    imgRGB = imresize(imgRGB, [NaN prefRes(2)]);
end
width = size(imgRGB, 2);    % Larghezza dell'immagine post-compressione
height = size(imgRGB, 1);   % Altezza dell'immagine post-compressione
res = width*height;         % Risoluzione dell'immagine post-compressione
center = [floor((width+1)/2), floor((height+1)/2)];
%% RGB to gray scale
I= rgb2gray(imgRGB);

%% grey scale to BW
BW = imbinarize(I);
BW1=BW;
%% Verifica della luminosità dell'immagine, se troppo chiara effettuo il complementare
% Somma righe e colonne dei pixel dell'immagine
nPixelW = sum(sum(BW1));
if(nPixelW >= (prefRes(1)*prefRes(2))/2)
 %%Complement Image from WB to->BW
    BW1 = ~BW1;
end
%% Rimozione rumore
% utile per eliminare artefatti grafici (es: ombre)
 pxToDel = 40;        % Soglia di pixel da considerare come rumore.
BW2 = bwareaopen(BW1,pxToDel);

%%  Maschere di Dilatazione ed Erosione
% Definisco operatore morfologico: Applico maschere quadrate
mask = strel('square', 10);
% imclose() applica operatori morfologici su immagini in b/n  o greyscale
BW3 = imclose(BW2, mask);
%% Eliminazione dei "buchi" dall'immagine
BW4 = imfill(BW3, 'holes');

%%

%traces the exterior boundaries of objects, as well as boundaries of holes
```

```
%inside these objects, in the binary image BW.
[B,L,n,A] = bwboundaries(BW4,'noholes');
%B:=exterior boundaries of the objects founds;
%L:= label matrix L where objects and holes are labeled.
%n:= number of object found
%A:= adiajency matrix of BW image
%regionprops returns measurements for the current identified region
stats = regionprops(L,'all');

 %%Proprieta dell'oggetto identificato:Area e Perimetro
      p=max([stats.Perimeter]);
      a=max([stats.Area]);
  %compute the roundness metric
  roundness = 4*pi*a/p^2;
  % Circolarità dell'oggetto se prossimo ad 1->cerchio
   circularity=roundness^(-1);
for k=1:size(stats,1)
    if stats(k).Area == a && stats(k).Perimeter==p

      boundary=B{k};
      apothem = a*2/p;
      epsilon=0.03;
     objShape = "Irregolare";

 fixed = [0.289,0.5,0.688,0.866];
polig = ["Triangolo", "Quadrilatero", "Pentagono", "Esagono"];
%In base all'apotema,perimetro ed area si identifica la forma dell'oggetto
for i=1:4
    numLati = i+2;
    disp(apothem/p*numLati)
    if((apothem/p*numLati) < (fixed(i)+epsilon) && (apothem/p*numLati) > (fixed(i)-↙
epsilon))
        objShape = polig(i);
        break;

    end
end
if (strcmp(objShape, "Irregolare") == 1)
    epsilon = 0.1;
   circularity=roundness^(-1);
    if( circularity< 1+epsilon && circularity > 1-epsilon )
            objShape = "Circolare";
    end
end

    end
end

%% rasformata di Radon per determinare baricentro ed orientamento
% Si calcolano le proiezioni radiali dell'oggeto , per identificare
%l'orientamento dell'oggeto e per tracciare le diagonali principali dell'oggetto
%appena riconosciuto. L'intersezione delle diagonali determina la posizione
%del baricentro
% Definiscl'intervallo in gradi su cui effettuare le proiezioni
theta = 0:179;
```

```
% Eseguo le proiezioni.
% R è la proiezione dell'oggetto, Xp è il vettore delle coordinate di ogni riga↙
dell'immagine.
[R, xp] = radon(BW4, theta);
% Determino la proiezione con altezza maggiore per determinare la diagonale
maxRadon = max(R);

%% Determino il massimo per identificare il punto più alto della
% proiezione con altezza maggiore.
% [pk, locs] = findpeaks() identifica il massimo locale del vettore dato in
% ingresso e l'indice di quel valore all'interno del vettore.
% SosrtStr specifica che i risultati andranno ordinati
% NPeaks specifica quanti massimi locali trovare nel vettore.
[pk, locs] = findpeaks(maxRadon, 'SortStr', 'descend', 'NPeaks', 2);

% Trovo angolo in radianti il cui indice corrisponde all'elemento di una
% delle colonne di R il cui valore è pari al picco individuato da pk
theta1 = locs(1);
offset1 = xp(R(:, locs(1)) == pk(1));
theta2 = locs(2);
offset2 = xp(R(:, locs(2)) == pk(2));

% Determino le diagonali
diag1(x) = tand(theta1+90) * (x - offset1*cosd(theta1)) + offset1*sind(theta1);
diag2(x) = tand(theta2+90) * (x - offset2*cosd(theta2)) + offset2*sind(theta2);

% Identifico il baricentro nel punto d'intersezione delle diagonali
x_bc = solve(diag1 == diag2);
y_bc = diag1(x_bc);

% Determino Orientamento a partire da due angoli identificati dalla
% trasformata di Radon

orient = (theta1+theta2)/2;
% Controllo se c'è discordanza tra i quadranti identificati dagli angoli
if(sign(sind(theta1)) ~= sign(sind(theta2)) || sign(cosd(theta1)) ~= sign(cosd↙
(theta2)) )
    orient = orient-90;
end
% Mi assicuro che l'angolo trovato sia tra -90° and 90°
orient = atand(sind(orient)/cosd(orient));
 %% Visualizzazione immagine pre e post processamento
montage({imgRGB,I,BW,BW1,BW2,BW3,BW4})
figure()
imshow(BW4)
title('Identificazione Oggetto');

hold on;

%% Grafico delle Diagonali
plot(center(1) + offset1*cosd(theta1), center(2) - offset1*sind(theta1), 'r*',↙
'MarkerSize', 10);
plot(center(1) + offset2*cosd(theta2), center(2) - offset2*sind(theta2), 'g*',↙
'MarkerSize', 10);
plot(center(1) + x_bc, center(2) - y_bc, 'bo', 'MarkerSize', 10);
```

```
diag1Plot = center(2) - tand(theta1+90) * (x - center(1) - offset1*cosd(theta1)) -↙
offset1*sind(theta1);
diag2Plot = center(2) - tand(theta2+90) * (x - center(1) - offset2*cosd(theta2)) -↙
offset2*sind(theta2);
lineOrient = center(2) - tand(orient) * (x - center(1) - x_bc) - y_bc;
fplot(diag1Plot, 'r', 'LineWidth', 1.5);
fplot(diag2Plot, 'g', 'LineWidth', 1.5);
fplot(lineOrient, 'y', 'LineWidth', 2);
%% Visualizzo il contorno dell'oggetto identificato
  text(boundary(1,1)+10,boundary(2,2)+10,objShape,'Color','magenta');
  plot(boundary(:,2),boundary(:,1),'red','LineWidth',2);
 txt=sprintf("Perimeter:%f\tArea:%f\nForma:%s\nRoundness:%d\nOrientation:%↙
f\nBaricentro=%d\t%d",p,a,objShape,roundness,orient,x_bc,y_bc);
  disp(txt);
```