**Circle detection & Classification (Matlab)**

Results: 89 tomatoes found





Code in Matlab:

%main tomatoes

clc

clear all

close all

I\_color=imread('assorted\_tomatoes.jpg');

%find grayscale image

I\_gray=rgb2gray(I\_color);

[hight, width]=size(I\_gray);

%perform edge detection with Canny,do it separately for Grayscale image,Red channel,

%Green channel & Blue channel

I\_edge=edge(I\_gray,'Canny',[0.05,0.15]);

I\_edgeRed=edge(I\_color(:,:,1),'Canny',[0.05,0.15]);

I\_edgeGreen=edge(I\_color(:,:,2),'Canny',[0.05,0.15]);

I\_edgeBlue=edge(I\_color(:,:,3),'Canny',[0.05,0.15]);

%enhance edges with dilation (connectivity 4)

SE=[0 1 0; 1 1 1; 0 1 0];

I\_edge = imdilate(I\_edge,SE);

I\_edgeRed = imdilate(I\_edgeRed,SE);

I\_edgeGreen = imdilate(I\_edgeGreen,SE);

I\_edgeBlue = imdilate(I\_edgeBlue,SE);

%find circles in edge images: Grayscale,Red, Green & Blue

%channels, get circles centers and radiuses

[centersE,radiiE] = imfindcircles(I\_edge,[30 300],'ObjectPolarity','dark','Sensitivity',0.9);%max(hight, width)]);

[centersR,radiiR] = imfindcircles(I\_edgeRed,[30 200],'ObjectPolarity','dark','Sensitivity',0.91);

[centersG,radiiG] = imfindcircles(I\_edgeGreen,[30 200],'ObjectPolarity','dark','Sensitivity',0.915);

[centersB,radiiB] = imfindcircles(I\_edgeBlue,[30 200],'Sensitivity',0.9);

%Group the circles from all different edge images, and take care of

%overlappings

%first initialize by checking for overlaps in grayscale image circles

[centersE,radiiE]=Add\_new\_circles([],[],centersE,radiiE,I\_color);

%add non-overlaping circles from all color edge images

[centers,radii]=Add\_new\_circles(centersE,radiiE,centersR,radiiR,I\_color);

[centers,radii]=Add\_new\_circles(centers,radii,centersG,radiiG,I\_color);

[centers,radii]=Add\_new\_circles(centers,radii,centersB,radiiB,I\_color);

%for each circle find average color of its area in original image

colors=find\_circle\_color(I\_color,centers,radii);

%Define Reference colors

Colors\_label = {'Red'; 'Brown'; 'Green'; 'Yelow'};

Ref\_Colors=[200,40,40; 140,42,42; 130,160,60; 200,155,20];

%Classify by matching each color in 'colors' to Reference colors

color\_match = classify\_color(Ref\_Colors,colors);

%plot Graph of tomatoes color vs. radius

figure;

plot(color\_match,radii,'o')

set(gca,'xtick',[1:4],'xticklabel',Colors\_label)

title('Tomatoe color vs. radiu');

%show final result with all circles and classifications

figure;

imshow(I\_color);

hold on

viscircles(centers,radii,'EdgeColor','b');title(['Final Result- ' num2str(length(radii)) ' tomatoes found']);

hold on

for i=1:length(color\_match)

text(centers(i,1),centers(i,2),Colors\_label(color\_match(i)));

text(centers(i,1)-25,centers(i,2)+15,['(',num2str(round(colors(i,1))),',',num2str(round(colors(i,2))),',',num2str(round(colors(i,3))),')']);

hold on

end

Additional functions:

function [centers,radii]=Add\_new\_circles(centersA,radiiA,centersB,radiiB,I)

%adds non-overlapping circles parameters from list B to list A

if(isempty(radiiA))%initialize if we receive an empty list

if(~isempty(radiiB))

radiiA(1,:)=radiiB(1,:);%copy first circle parameters to A

centersA(1,:)=centersB(1,:);

else%both lists are empty

msg='empty input arguments';

error(msg);

end

end

if(isrow(radiiA))

radiiA=radiiA';

end

max\_color\_dist=40;

Max\_circle\_distance=10;

Max\_radius\_distance=20;

A\_length=length(radiiA);

B\_length=length(radiiB);

%Initialize output lists

centers=centersA;

radii=radiiA;

added\_circles=0;

for i=1:B\_length%for each circle in list B

if(isrow(radii))

radii=radii';

end

move\_on=0;

%find distance between the center of the candidate circle to all

%current circle centers on list A

xdist=centers(:,1)-centersB(i,1);

ydist=centers(:,2)-centersB(i,2);

center\_distances=sqrt(xdist.^2+ydist.^2);

%same for radius distances

rad\_distances=abs(radii-radiiB(i));

%look for overlapings (we consider overlap when distance between centers<max(R\_a,R\_b))

overlap\_center\_index=find(center\_distances<max(radii,radiiB(i)));

if(~isempty(overlap\_center\_index))%we found center overlap

%check if one of the overlapping circles in the list is very similar to

%the the new circle - if so, dont add the circle and move to next circle in B

for j=1:length(overlap\_center\_index)

if (center\_distances(overlap\_center\_index(j))<Max\_circle\_distance && rad\_distances(overlap\_center\_index(j))<Max\_radius\_distance)

move\_on=1;

end

end

%check total overlap - if one of the circles in overlaping\_list is containing the circle from B or

%containd by him (it happens when d<|R\_a-R\_b| - leave only the

%largest

if(~move\_on)

for j=1:length(overlap\_center\_index)

if(center\_distances(overlap\_center\_index(j))<rad\_distances(overlap\_center\_index(j)))

move\_on=1;

if(radii(overlap\_center\_index(j))<radiiB(i))%if the circle in B is larger

centers(overlap\_center\_index(j),:)=centersB(i,:);%Replace circle A with B

radii(overlap\_center\_index(j))=radiiB(i);

end

end

end

end

%overlapping is not total but large enough it could be an elipce

if(~move\_on)

D=0;

for j=1:length(overlap\_center\_index) %for each overlaping circle

%check average color inside the two circles.

centers\_to\_check=[centers(overlap\_center\_index(j),:);centersB(i,:)];

radiuses\_to\_check=[radii(overlap\_center\_index(j));radiiB(i)];

colors=find\_circle\_color(I,centers\_to\_check,radiuses\_to\_check);

% calculate L2 distance between the two average colors

D(j) = sqrt(sum((colors(1,:) - colors(2,:)) .^ 2));

end

if (min(D)<max\_color\_dist) %the two overlaping circles has similar color - we decide its elipce

%average the radius and centers

idx=find(min(D));

radii(overlap\_center\_index(idx))=mean([radii(overlap\_center\_index(idx)) radiiB(i)]);

centers(overlap\_center\_index(idx),:)=(centers(overlap\_center\_index(idx),:)+centersB(i,:))./2;

else %overlaping circles are with different colors - add the new circle to the list

added\_circles=added\_circles+1;

centers(A\_length+added\_circles,:)=centersB(i,:);

radii(A\_length+added\_circles)=radiiB(i);

end

end

else

%There is no overlap at all so we can add the new circle to the list

added\_circles=added\_circles+1;

centers(A\_length+added\_circles,:)=centersB(i,:);

radii(A\_length+added\_circles)=radiiB(i);

end

end

end

function colors=find\_circle\_color(I,centers,radii)

%find\_circle\_color(I,centers,radii) Returns a list of average colors of circled

%areas in image I according to circles center coordinates and radiuses

%initialize

Number\_of\_circles=length(radii);

colors=zeros(Number\_of\_circles,3);

[hight, width]=size(I(:,:,1));

for i=1:Number\_of\_circles%for each circle

%Get a binary circle image

[Binary\_circle,num\_pixels\_in\_tomato]=draw\_white\_circle(centers(i,:),radii(i),hight,width);

%multiply binary tomato with original image per color channel

single\_tomato\_image(:,:,1)=Binary\_circle.\*double(I(:,:,1));

single\_tomato\_image(:,:,2)=Binary\_circle.\*double(I(:,:,2));

single\_tomato\_image(:,:,3)=Binary\_circle.\*double(I(:,:,3));

%for each color channel find average value inside the colored circle

colors(i,:)=sum(sum(single\_tomato\_image))./num\_pixels\_in\_tomato;

end

end

function [Binary\_circle,num\_pixels\_in\_tomato]=draw\_white\_circle(center,radius,hight,width)

%Returns a binary image with size hight X width X 3 , for each color channel all pixels inside of a

%the circle specified with its center coordinats and radius are equal 1 and

%0 otherwise

Binary\_circle=zeros(hight,width);

num\_pixels\_in\_tomato=0;

for i=1:hight

for j=1:width

if ((j-center(1))^2+(i-center(2))^2<radius^2)

Binary\_circle(i,j)=1;%fill with 1 each color channel if we inside the circle

num\_pixels\_in\_tomato=num\_pixels\_in\_tomato+1;

end

end

end

end

function Idx = classify\_color(Ref\_Colors,colors\_list)

%Returns the matching index in Ref\_Colors for each color in colors\_list

Num\_of\_colors=length(colors\_list);

Num\_of\_ref\_colors=length(Ref\_Colors);

D=zeros(length(colors\_list),length(Ref\_Colors));

Idx=zeros(length(colors\_list),1);

for i=1:Num\_of\_colors

for j=1:Num\_of\_ref\_colors

%calculate L2 norm distances between each color to each reference color

D(i,j) = sqrt(sum((colors\_list(i,:) - Ref\_Colors(j,:)) .^ 2));

end

%Classify according to the minimum norm distance

Idx(i)=find(D(i,:)==min(D(i,:)));

end