HOME ASSIGNMENT FOR MACHINE VISION COURSE (3621518)

Project: Counting empty bottles in bottle crates

Student name: LAHARI SENGUPTA

Student number: 277146

Abstract: Image segmentation and morphological image operations are very powerful tool for image analysis and processing. In this project, these two methods are mainly used to implement a code to identify the number of empty bottles in bottle crates. In the following sections, all the techniques which are used for implementation and the results are described. From the result, it is found that the technique is powerful enough for the objective, but sometimes, extra noise has created few errors. This implementation can be used as a part of whole software.

Technical Discussion:

In the original images, we have bottle reflection components and some background components. First we need to segregate background and bottle components. So, image segmentation techniques are required to extract out the background and bottle components individually. Threshold process is one of the techniques of segmentation. The bottle components are extracted out from the image by threshold.

Threshold: It is the simplest method of segmentation and commonly used to convert a grayscale image into a binary one. In this process, the intensity values of image pixels are set to either 0 or 1 based on a threshold value, i.e. the intensity values less than the threshold value are set to 0 and greater than the threshold value are set to 1.

The background components could also be extracted by threshold techniques. But I found better result by using edge detection technique. In these images, the background components are mostly thin components (almost like edge) and probably that is the reason of giving edge detected result better. I tried with "Sobel", "Prewitt", "Laplacian of Gaussian" techniques and found optimum result for "Prewitt". So, I have used the "Prewitt" edge detection technique for it.

Edge detection: This is another kind of segmentation approach generally used to detect discontinuities in intensity values of an image. If a set of connected pixels in an image is the boundary of two different gray scale values, then this set of connected pixels is an edge and this discontinuity can be detected by edge detection technique. Prewitt is one kind of different masks those are used for implementing the operation on the images.

Now the background and bottle components are separated into two images. Morphological opening operation is done on background components and this background components are now subtracted from the bottle component image now. So, finally an image without the unnecessary background is achieved. As opening finds out the components distinguishably so, it helps to recover the background.

Morphological image opening: Morphological operations on image involves with changing the form and structure of the image. Opening smoothes the contour of an object, breaks the sharp and narrow corners. Opening is combination of image erosion and then followed by dilation.
Erosion and Dilation: Erosion shrinks an image and dilation expands.

After removing the background, there are still noisy areas which can be classified as clutter. This clutter is removed based on the size. The bottle sizes are lesser than the clutter. So, here the bigger sized components are removed. To find the component, "bwlabel" operation is performed which finds the connected components in image in MATLAB. Here, I have used 4-connectivity to find the connected components. Then "regionprops", which measures the properties of image regions, finds the area of each components. After finding the areas, the clutter parts are removed.

The image without the clutter is now again opened to smooth it and to make the circular like shaped bottle components to more circular. There are also some small noises and those are removed by "bwareopen" operation by selecting the size of small noises.

Now, I have dilated the image a few so that some non-continuous bottle openings can expand a little.

Finally, I have the small noise-free, clutter-free image. It may contain the circular bottle candidates as well as some large non-circular components. So, eccentricity is measured for each component to find out the circles only. This extracted image contains some circular holes, which is now filled with “imfill”. Now by area difference, I have discriminated the wrong placement i.e. upside down placement. In few images, I have found the structure of the bottle crate takes as a circle where there is no bottle. So, these are also eliminated from my final detection by size difference of connected components. Lastly I have detected the bottles by green and orange circles as given in example code.

Results:

There are total 24 images for bottle crates. I have executed my solution for all the 24 images and found following results:

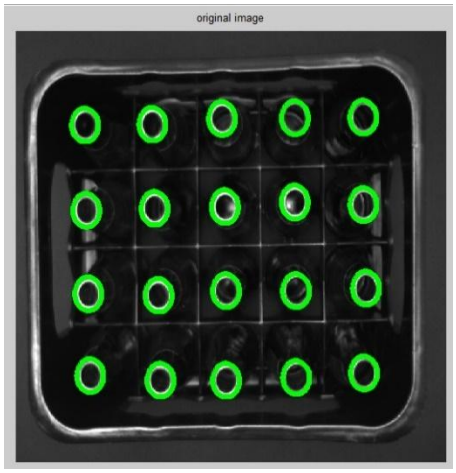

Figure 1: Bottle_crate_01

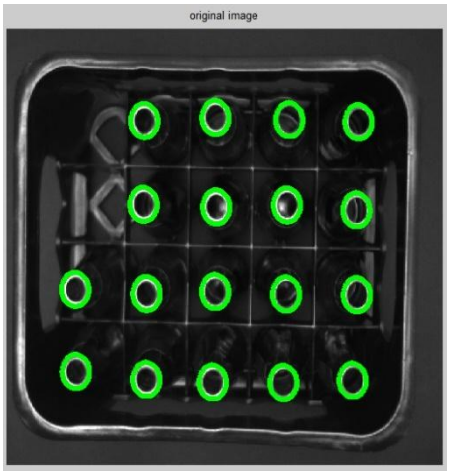

Figure 2: Bottle_crate_02

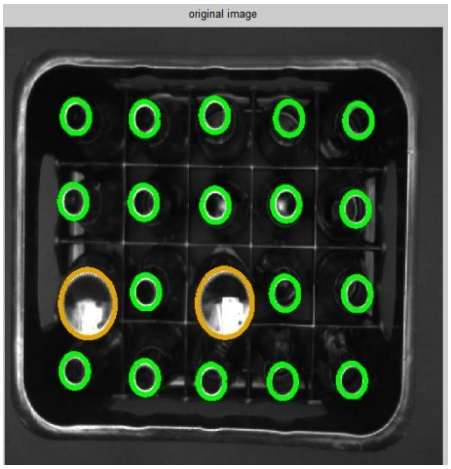

Figure 3: Bottle_crate_03

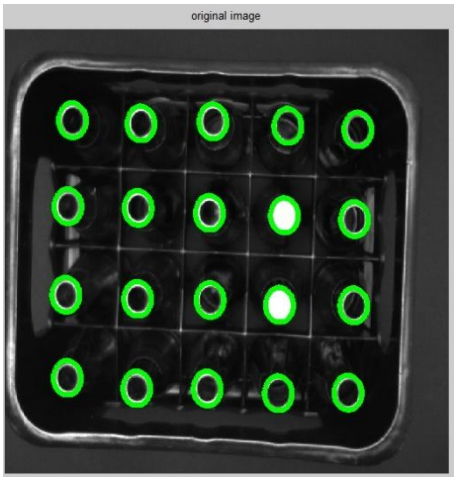

Figure 4: Bottle_crate_04

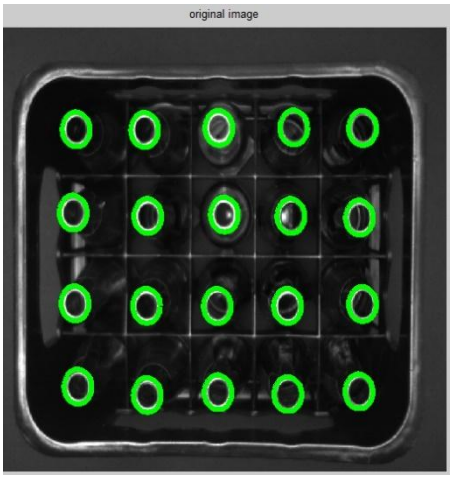

Figure 5: Bottle_crate_05

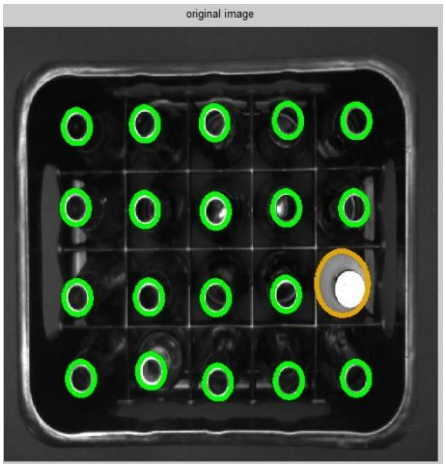

Figure 6: Bottle_crate_06

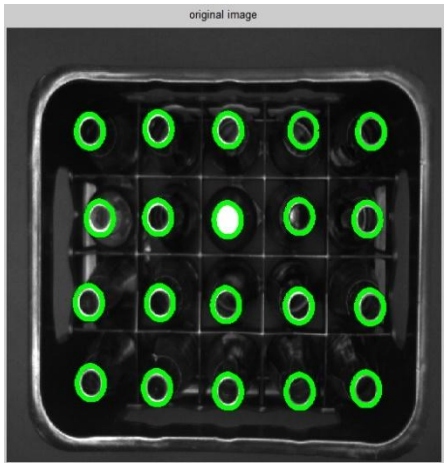

Figure 7: Bottle_crate_07

Figure 8: Bottle_crate_08

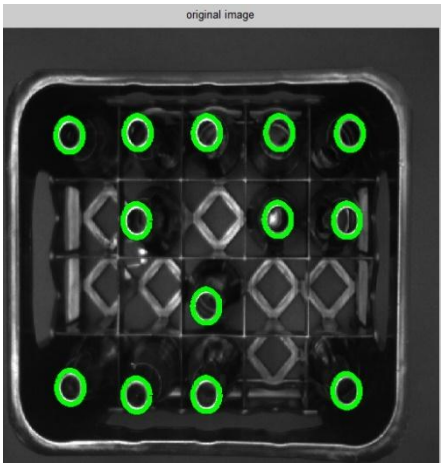

Figure 9: Bottle_crate_09

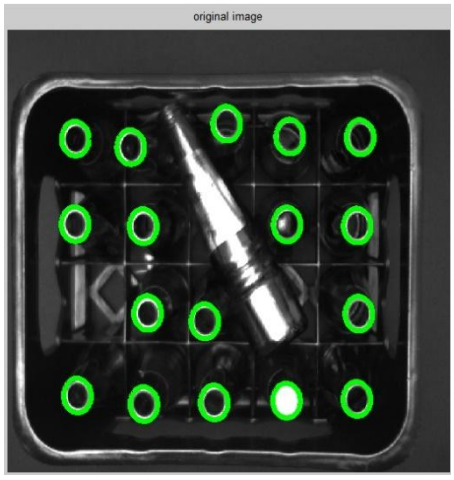

Figure 10: Bottle_crate_10

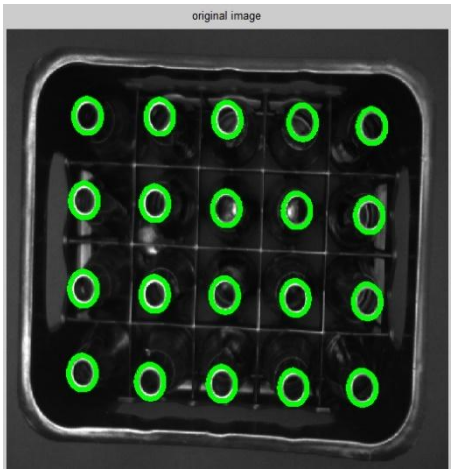

Figure 11: Bottle_crate_11

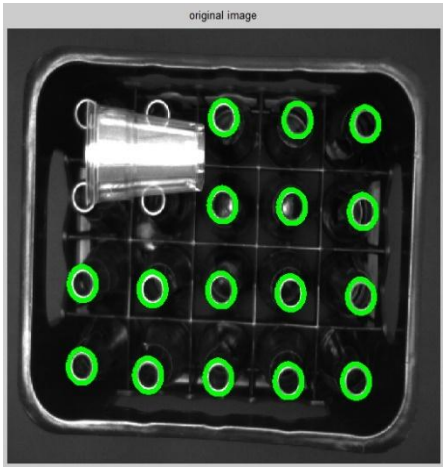

Figure 12: Bottle_crate_12

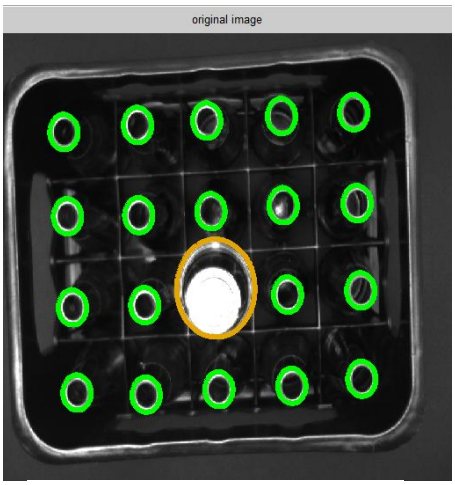

Figure 13: Bottle_crate_13

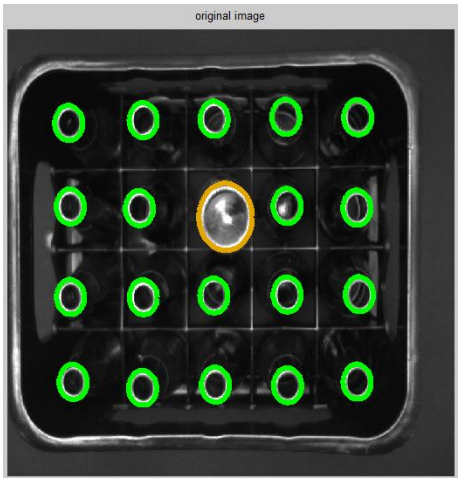

Figure 14: Bottle_crate_14

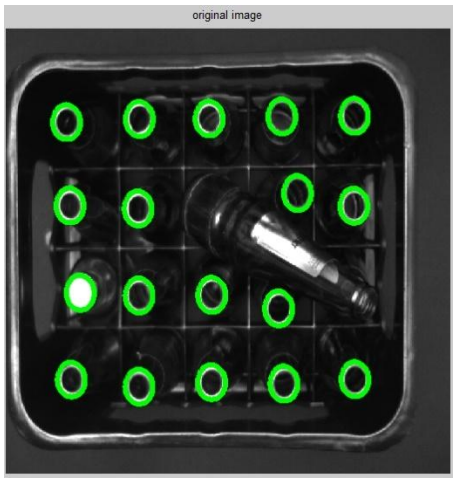

Figure 15: Bottle_crate_15

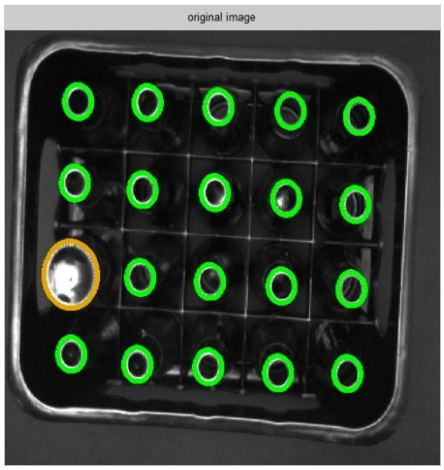

Figure 16: Bottle_crate_16

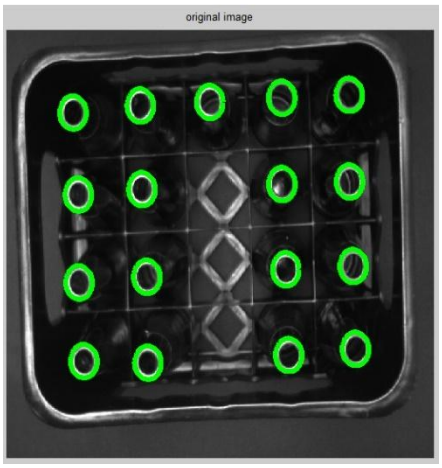

Figure 17: Bottle_crate_17

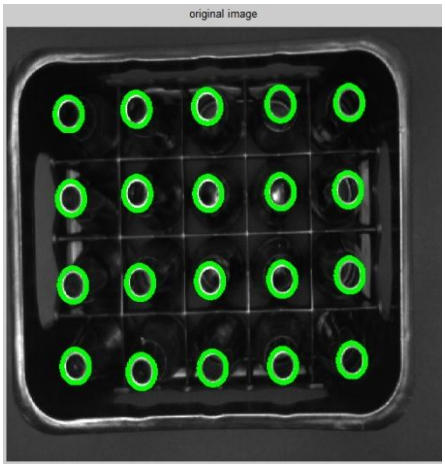

Figure 18: Bottle_crate_18

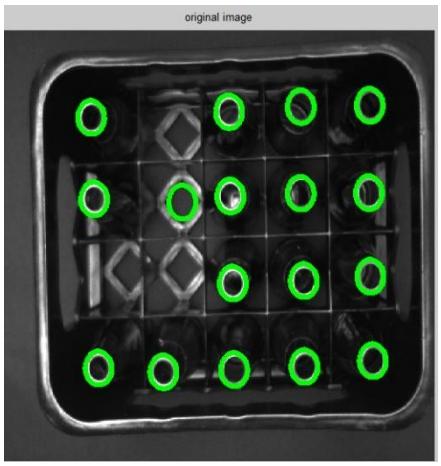

Figure 19: Bottle_crate_19

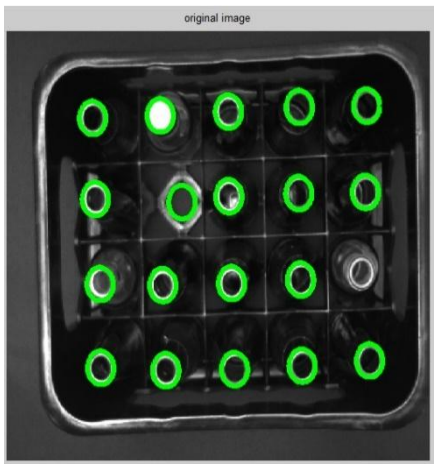

Figure 20: Bottle_crate_20

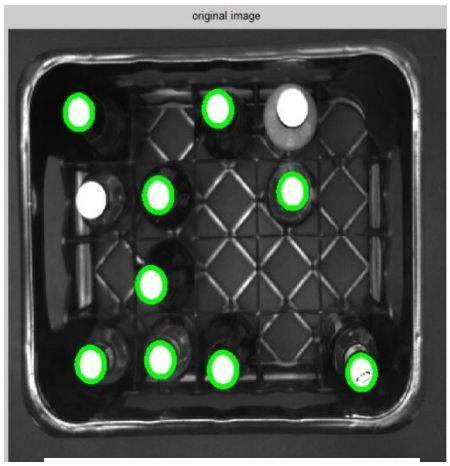

Figure 21: Bottle_crate_21

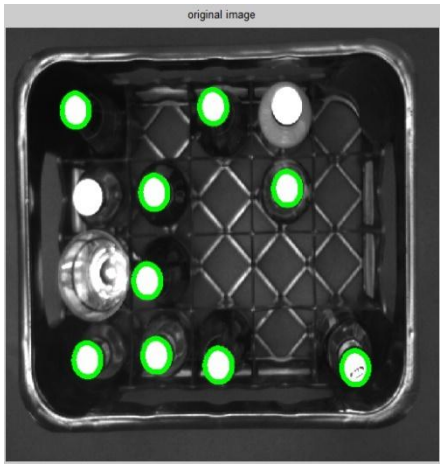

Figure 22: Bottle_crate_22

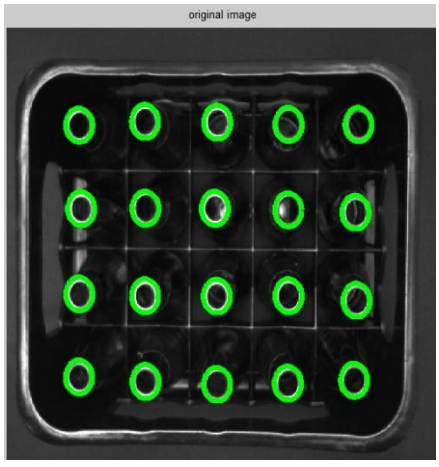

Figure 23: Bottle_crate_23

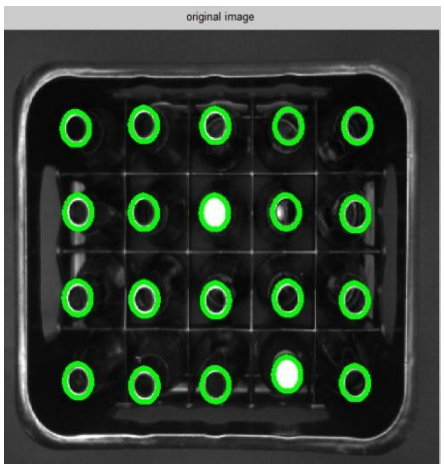

Figure 24: Bottle_crate_24

Discussion of results:

In most of the cases the program detected correct number of empty bottles as well as the placement of the bottle (whether upside down or not). Still my technique failed in very few cases.

i) In figure 8, it missed the 3rd bottle of the last row. As the reflection of the bottle in this case is very low contrast, so, at the time of thresholding, it goes into black level.

ii) In figure 19 and 20, my technique detects an empty space as a bottle. The structure of the empty space is such that it can reflect light a lot. We also have more empty spaces in other images. The reflection of the empty space texture is creating a large component in the processed binary image. In my solution, I have included a condition to remove such components. So, in other images, the empty spaces are not detected as bottle. But in these two cases, the size of the component given for removal condition is not matching, so, they are still there.

iii) In figure 22, my solution missed one bottle which is placed upside down (first in the 3rd row). The reflection is quite bad to detect.

So, where the bottle reflections are less erroneous, there my solution works fine.

Appendix:

I have created a function to detect the bottles and modified the example code. Included the detect function calling in example code and found iteratively the correctly and wrongly placed bottles. Made two different color vector and two different radius value for two different bottle placements. Used the circle as it was given.

Modified **example.m**:

```
% Machine Vision
%
% example code for Home Assignment: Counting Empty Bottles


number_of_points=10000;                 % used for drawing circles
t=linspace(0,2*pi,number_of_points);    % used for drawing circles
daspect([1 1 1])                        % the axis aspect ratios are set to equal ratios

number_of_bottles=[20 18 20 20 20 20 20 20 13 17 20 17 19 20 18 20 17 20 16 19 11 11 20 20];
estimated_number_of_bottles=zeros(1,length(number_of_bottles));

for i=1:length(number_of_bottles)
   fname=sprintf('crate images/bottle_crate_%02i.png',i);
   f=imread(fname);
   imshow(f),title('original image')
   hold on
   [centres,area]=detect(f);            % calling the detect function which will detect the bottles.
                                        % Returns the centroid array and area array of the detected bottles
   sz=size(centres);                    % finds the size of the centroid array
   j=0;
```

```
   for j=1:sz(1)                          % iteratively finds which bottles are correctly placed and which are not
      centroid=centres(j,:);
      if (area(j) < 3500)
         radius=20;
         col=[0 1 0];                     %for blue color circle
      else
         radius=(sqrt(area(j)/pi));
         col=[0.9 0.65 0];                %for orange color cirle
      end
      h=circle(centroid,radius,number_of_points,'-');
      set(h,'LineWidth',5)
      set(h,'Color',col)
   end
   hold off
   estimated_number_of_bottles(i)=j;
   [estimated_number_of_bottles(i) number_of_bottles(i)]
pause
end
```

detect.m:

```
function [centres,area]=detect(image)          %function for detecting the bottles;
                                               %input is original image and output is centroid and area array

im_back=edge(image,'prewitt');      %edge detection is applied to extract out the background components
im_bottle=(image>105);              %thresholding performed to extract out the bottle components
se1=strel('line',5,0);              %structuring element created for opening
opened=imopen(im_back,se1);         %background components are opened
diff=im_bottle-opened;               %removed the background from bottle image

label=bwlabel(diff,4);              %labeling 4-connectivity components
s=regionprops(label,'Area');        %finding the area of the different components
idx = find([s.Area] > 6000);        %finding the area values which are greater than 6000 pixels
bw2 = ismember(label,idx);          %finding the components which are greater than 6000 pixel area i.e clutter
w_c=diff-bw2;                       %removing the clutter
se_clut=strel('line',1,0);          %structuring element creation for opening the clutter free image
w_c=imopen(w_c,se_clut);            %opening
cand=bwlabel(w_c,4);                %labeling 4-connectivity components
s2=regionprops(cand,'Area');        %finding the area of the different components
comp=bwareaopen(cand,175);          %removing the small noises which contains maximum 175 pixel area
se2=strel('line',2,0);              %structuring element creation for dilation
opd=imdilate(comp,se2);             %dilation is done to make full circle of the bottle components
c=bwlabel(opd,4);                   %labeling 4-connectivity components
s3=regionprops(c,'Eccentricity');   %finding the eccentricity of the components
idx2 = find([s3.Eccentricity]<=0.65); %finding the circular components
bw3 = ismember(c,idx2);             %circular components are extracted
bw3=imfill(bw3, 'holes');           %filling the holes
bw3=bwlabel(bw3,4);                 %labeling 4-connectivity components
s4=regionprops(bw3,'Area');         %finding the area of the different components
a=cat(1,s4.Area);                   %concatenating area values into an array
sz=size(a);                         %finding the size of the array
for i=1:sz                          %checking iteratively the areas of the circular components which are not bottles
```

```
    if a(i)>1800 && a(i)<3500          %bottles are placed either correctly or upside down.
        idx3 = find([s4.Area]<a(i));   %If it is placed correctly then the mouth of the bottles are upside
        bw3 = ismember(bw3,idx3);      %and if placed wrongly then back is upside which is bigger circle than the mouth
    end                                %so the components should have areas of mouth or areas of back.
    bw3=bwlabel(bw3,4);                %The components with middle area is suspicious
    s4=regionprops(bw3,'Area');        %here, I have found the mouth can be of
end                                    %maximum 1800 pixels and back is not less than 3500 pixels.
                                       %so, I removed the middle components
bw4=bwlabel(bw3,4);                    %labeling 4-connectivity components
v=regionprops(bw4,'Centroid','Area'); %finding the centroid and area of the different components
centres=cat(1,v.Centroid);             %concatenating the centroid in centres array and returning the centres
area=cat(1,v.Area);                    %concatenating the area in area array and returning the area
end
```