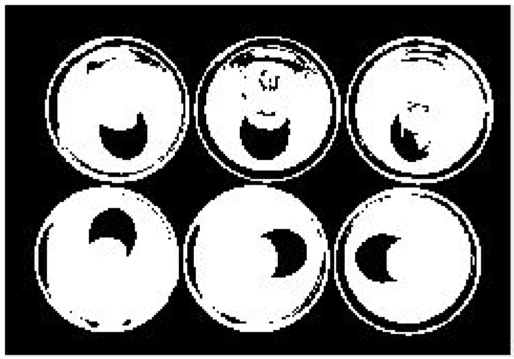
1)

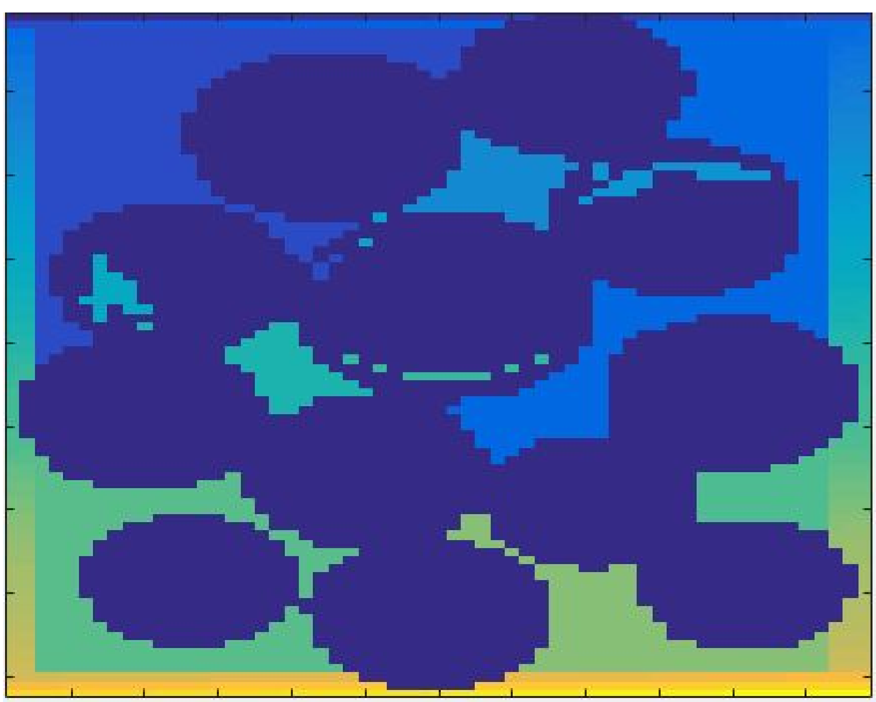
For this problem I converted the image to grayscale and obtained a histogram from the grayscale image. I inputted this histogram into a function which uses otsu's method to find the threshold value. This value is used to binarize the image, resulting in the image shown below.



2.1)

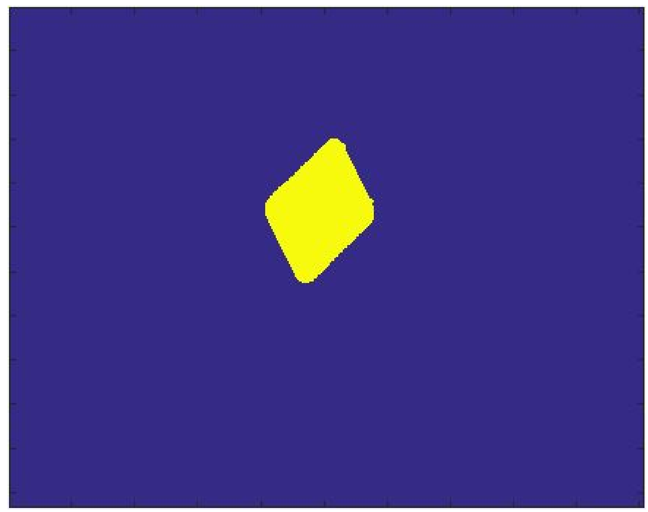
a) The 8 connectedness algorithm uses an empty matrix and the binary image obtained in problem 1 to separate the objects in the image. This is done by iterating through the binary image and calling a function that recursively marks the unvisited pixels of the empty matrix. The marked matrix has different values for the separate components in the image.

b) I originally thought silver coins would be filtered out as part of the background and there will be 3 objects because of the overlap between the coins. After testing it was clear that the coins were interpreted as the background objects while the negative space between the coins were shaded different colors since they were considered the foreground.

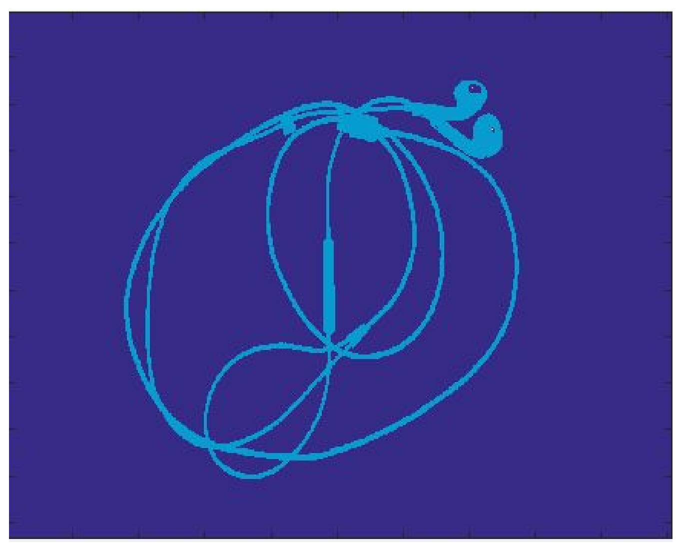


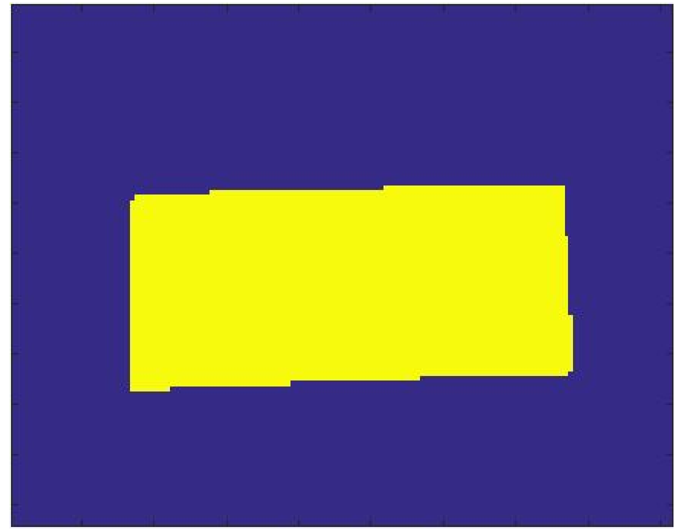
2.2) The algorithm did a pretty good job of separating the foreground and background because there was a very clear contrast between the two in the pictures taken. The centroid was found using the moment of the object with j and k corresponding to the y and x axes. The central moment, which is calculated using the centroid is used to find the eigenvalues and display the object's x and y in the correct direction/orientation.

Floss Container original and after CC algorithm



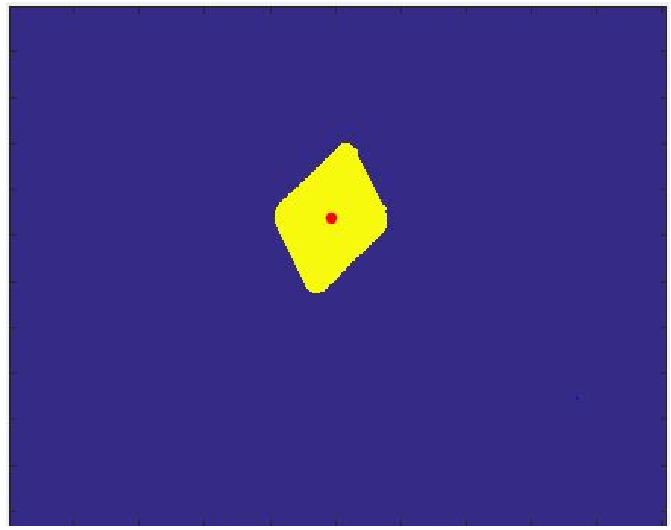
Headphones original and after CC algorithm



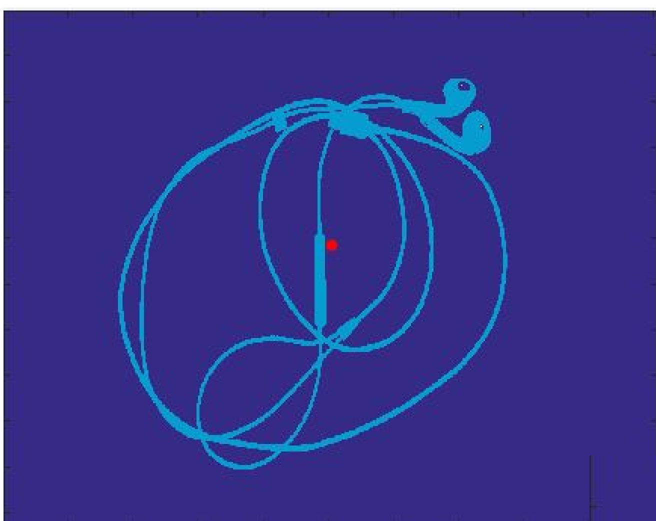


I Clicker original and after CC algorithm

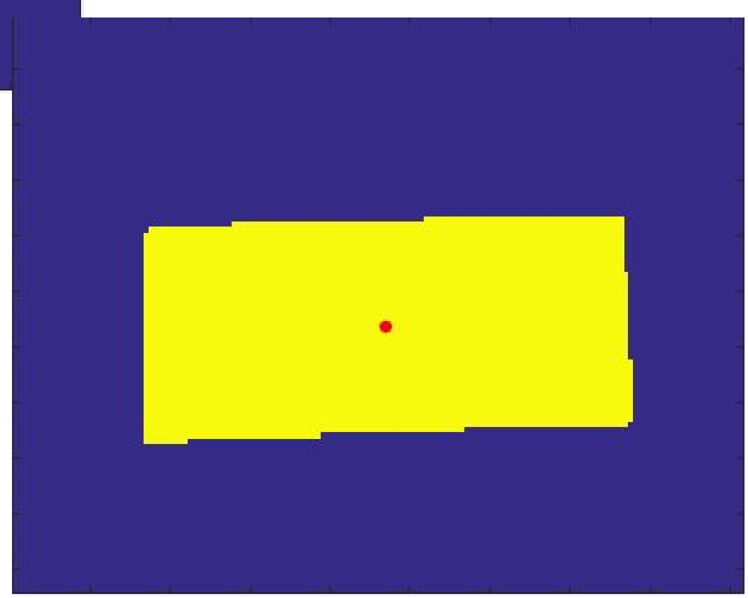
2.3) The centroid was calculated using the moment with j and k being 00 01 or 10.
This finds the center of the object because the moment only takes the current object's value into account.
The eigenvectors were calculated using the second central moment. The second central moment uses the centroid in conjunction with different j and k values in order to find the axes of the object.



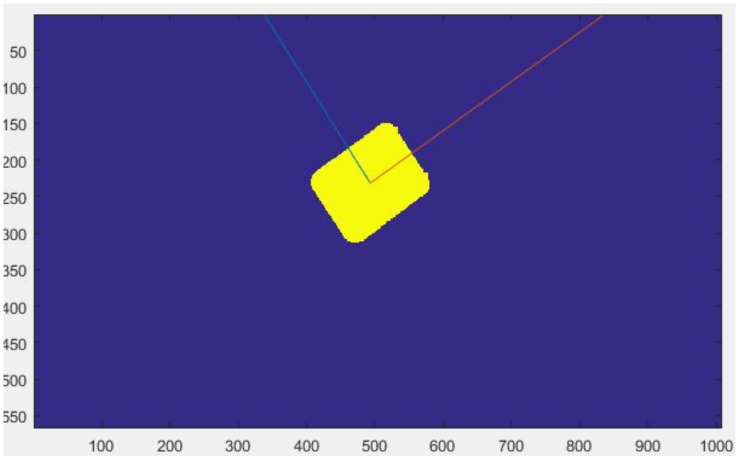
Clicker w/ Centroid



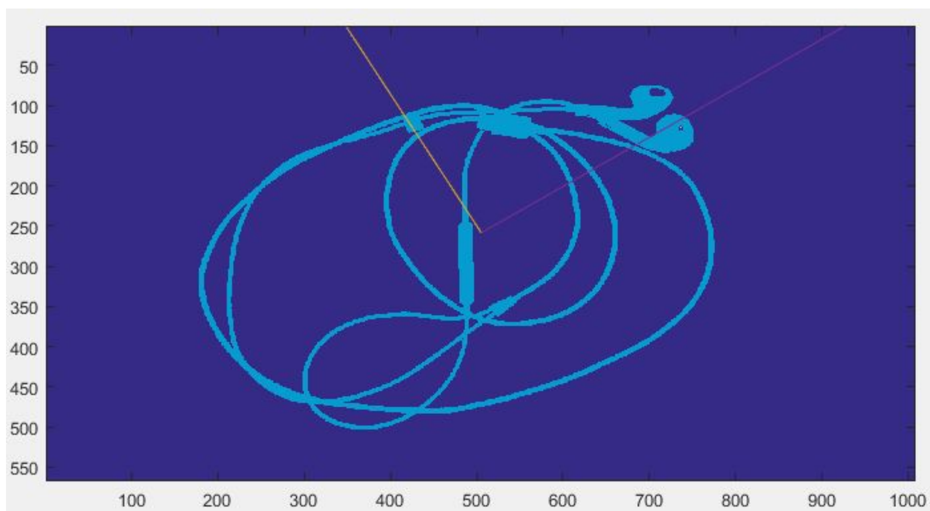
Headphones w/ Centroid



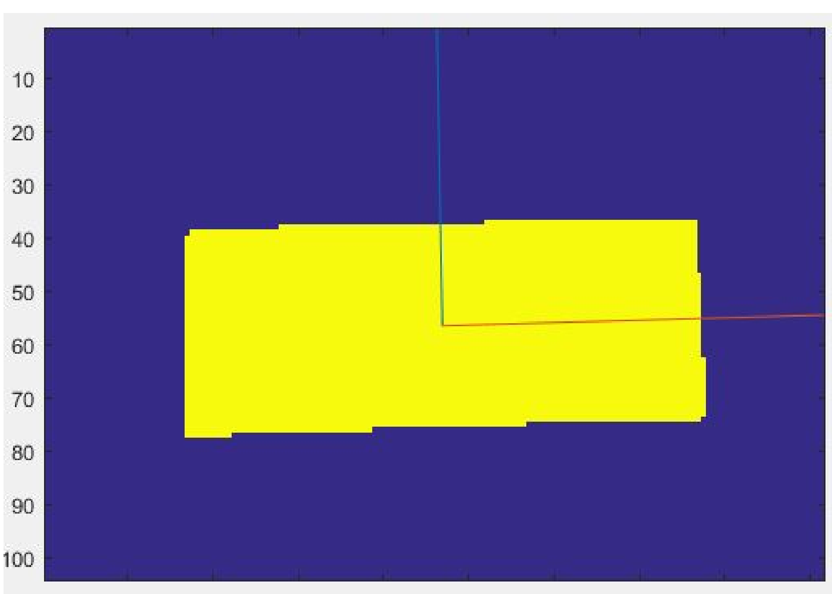
I Clicker w/ Centroid



Floss w/ Eigenvectors

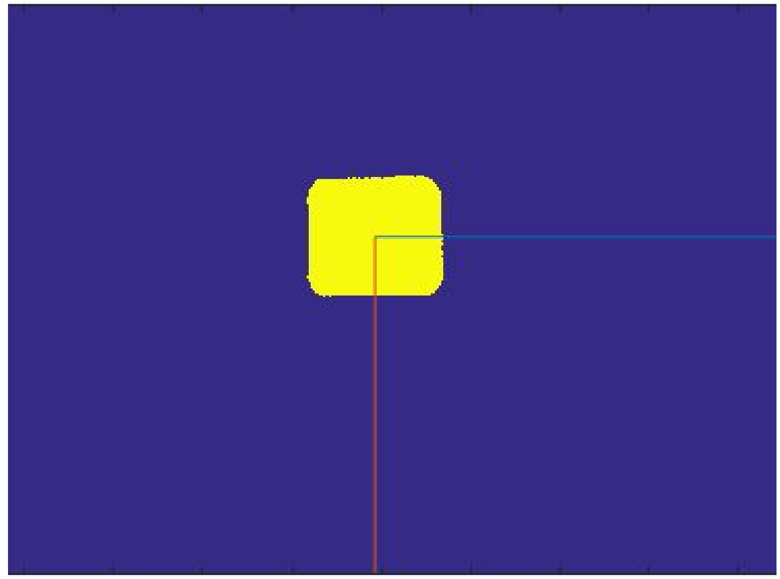


Headphones w/ Eigenvectors

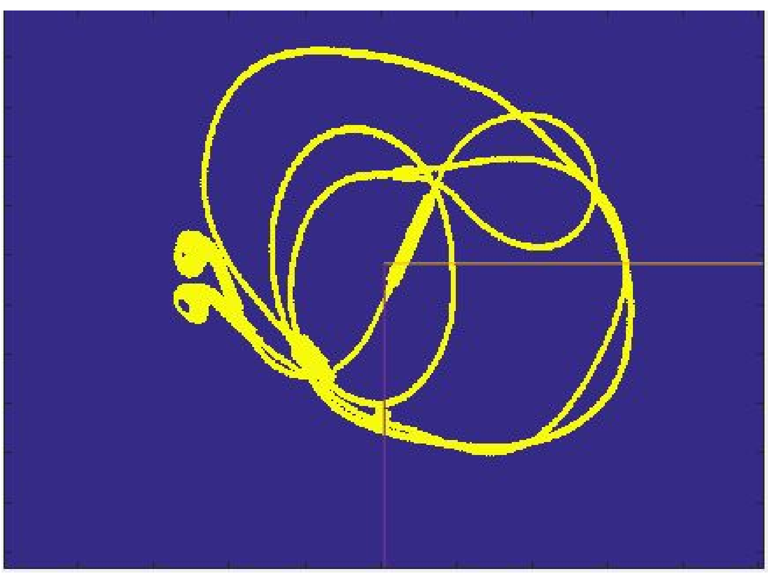


Clicker w/ Eigenvectors

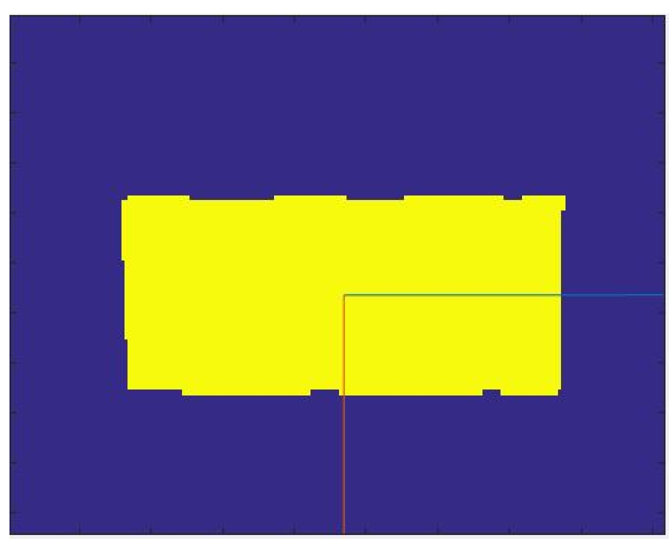
2.4) In order to rotate the object, the arctan of the eigenvectors was used to find the angle between the largest eigenvector and the x axis. The object was moved to the origin, rotated by this amount, and then moved back



Floss aligned w x

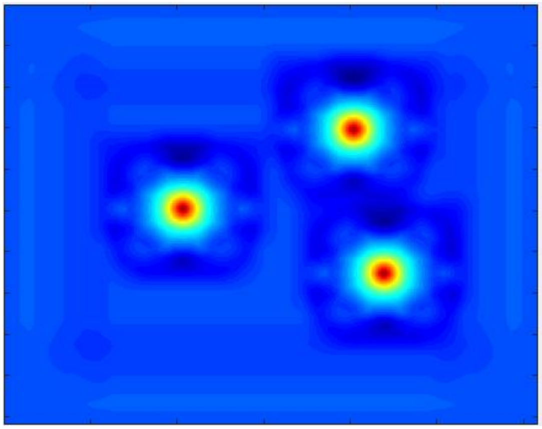


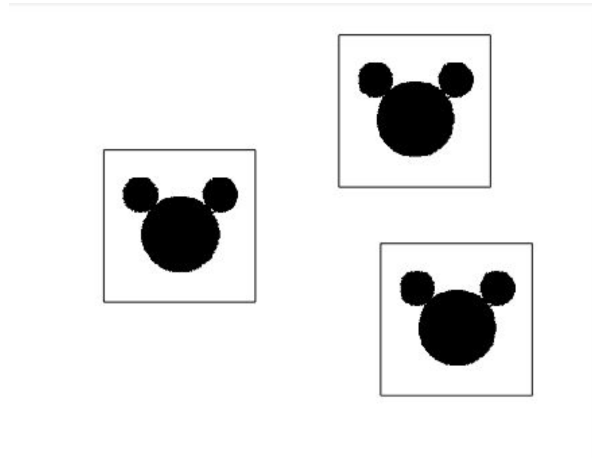
Headphones aligned w x



Clicker aligned w x

3.1)





The heatmap of the image found by using both the original image and a filter that was rotated by 180. The mean value of each image was subtracted in each pixel in order to get a new image and filter. These were then convoluted in order to find the heatmap, which has intensity values corresponding to the strongest similarities. The box was drawn by taking the pixels with the most intense values and using that as the center. The size of the filter was used for the box's proportions.

3.2)

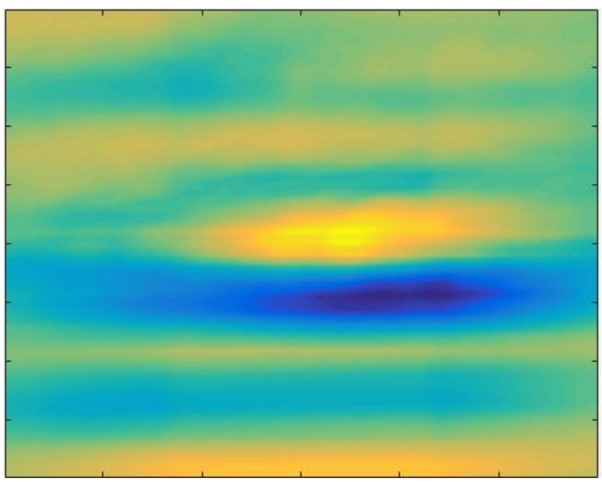
Center of box in same spot width/height scaled by 10
Err = 0.8593

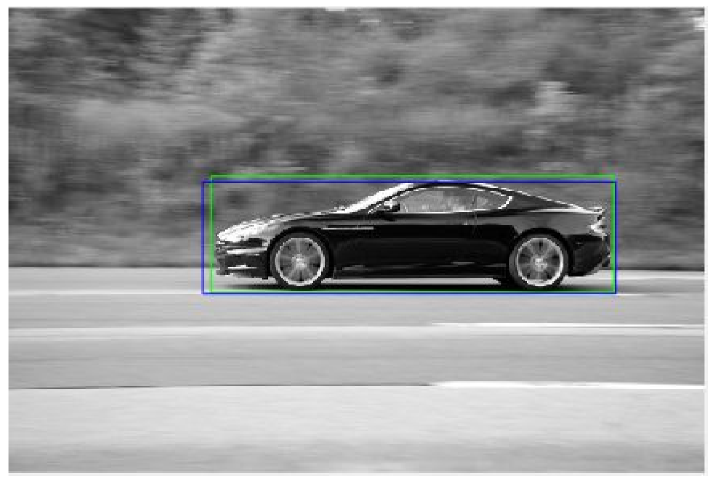
Center of box moved 5 up and down width/height same
Err = 0.8567

Center of box moved by 5 and scaled by 10
Err = 0.7437

This is in line with what is expected of the error. The further away/larger the difference in size is the less overlap there will be.

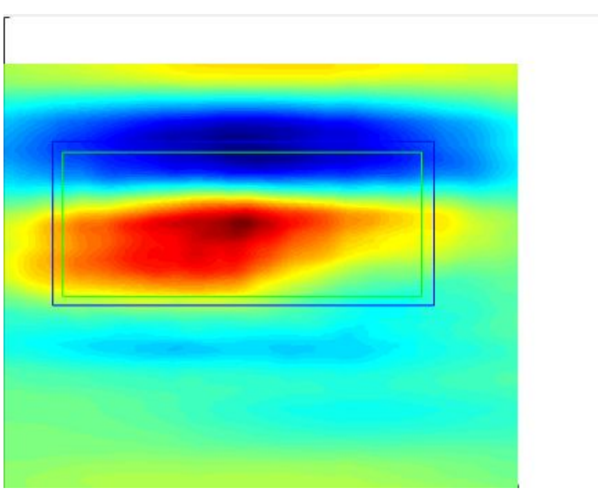
3.3)
car 1, scaled the filter image .1 vertical and .2 horizontal 0.9180 overlap





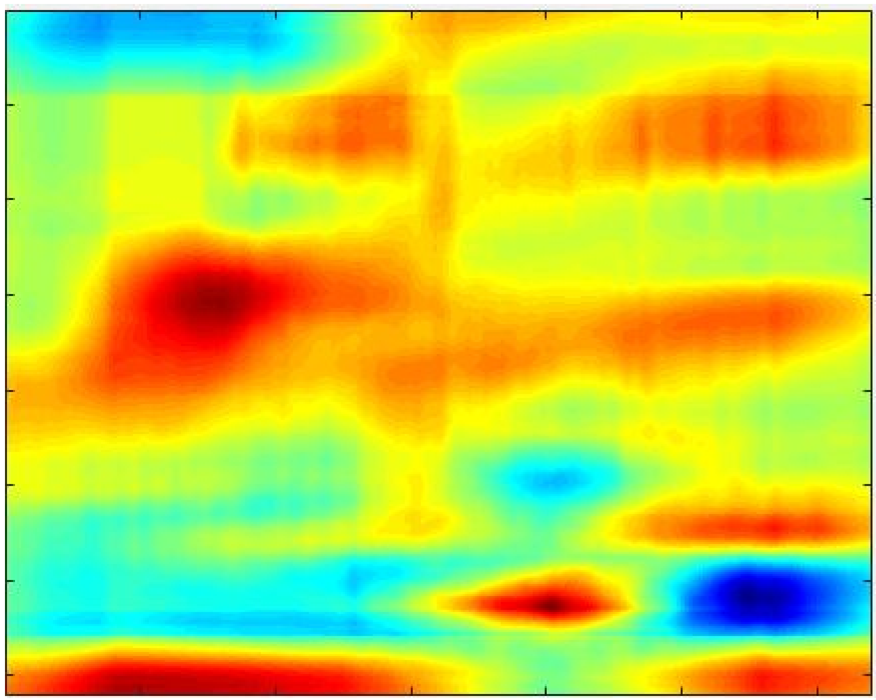
When the filter image was scaled to a similar size and proportion to the original image, the convolution did a good job in detecting the heatmap for the image and the bounding box was very accurate. Another reason that the overlap percentage is so high is that the filter image is very similar to the car shown above.

Car 2 scaled .18 vertically and .25 horizontally for an overlap rate of 0.8273
The error was slightly higher for this car because it was oriented in another direction and was viewed from a slight angle. Flipping the filter image did not change the output very much.



Car 3 scaled .09 vertically and horizontally and 0.6636 overlap

Since there were many objects in this image, a slightly different scaling would result in the max intensity value being a different point in the image that may have closer proportions to the filter image. The filter image must be very accurate in order to detect the car. The viewing angle of the car was extremely different than the car in the image so the convolution was not enough to accurately detect the car.



```
set(0,'RecursionLimit', 1000);
I = imread('floss.jpg');
IG = rgb2gray(I);
level = threshold(IG);
BinI = imbinarize(IG,level);
BinI = imresize(BinI, 0.25, 'bilinear');
%imshow(BinI);
global s;
s = size(BinI);
global mark;
mark = zeros(s(1),s(2));
global marker;
marker = 0;
for r = 1:s(1)
   for c = 1:s(2)
      if BinI(r,c) == 1 && mark(r,c) == 0
         marker = marker + 1;
         label(BinI,r,c)
      end
   end
end
%imagesc(mark);
M00 = moment(mark,0,0,1);
M10 = moment(mark,1,0,1);
M01 = moment(mark,0,1,1);
x = M10/M00;
y = M01/M00;
CM20 = centralMoment(mark,x,y,2,0,1);
CM11 = centralMoment(mark,x,y,1,1,1);
CM02 = centralMoment(mark,x,y,0,2,1);
Cmoment2 = [[CM20,CM11];[CM11,CM02]];
[v,d] = eig(Cmoment2);
Angx = atan2(v(2,1),v(1,1));
Rot1 = rotate(Angx);
d0 = Rot1*d;
% imagesc(mark);
% hold on;
% plot([x,d0(1,1)],[y,d0(2,1)])
% plot([x,d0(1,2)],[y,d0(2,2)])
% plot(x,y,'r.','MarkerSize',20)
[row,col] = size(mark);
```

```
alignedI = zeros(row,col);
ang = atan2(v(1,1),v(2,1));
rot = rotate(ang);
for r = 1:row
    for c = 1:col
      if mark(r,c) == 1
         orig = [c-x ;r-y];
         newcoord = rot*orig + [x;y];
         alignedI(floor(newcoord(2)),floor(newcoord(1))) = 1;
         alignedI(floor(newcoord(2)),ceil(newcoord(1))) = 1;
         alignedI(ceil(newcoord(2)),floor(newcoord(1))) = 1;
         alignedI(ceil(newcoord(2)),ceil(newcoord(1))) = 1;
      end
    end
end
% imagesc(alignedI);
% hold on;
% plot([x,d(1,1)],[y,d(2,1)])
% plot([x,d(1,2)],[y,d(2,2)])
function Rot = rotate(ang)
   Rot = [[cos(ang),-sin(ang)];[sin(ang),cos(ang)]];

end

function mom = moment(image,j,k,d)
[row,col] = size(image);
mom = 0;
   for r = 1:row
      for c = 1:col
         if image(r,c) == d
            mom = mom + ((r^k)*(c^j));
         end
      end
   end
end
function Cmoment = centralMoment(image,x,y,j,k,d)
   [row,col] = size(image);
   Cmoment = 0;
   for r = 1:row
      for c = 1:col
         if image(r,c) == d
            Cmoment = Cmoment + (((r-y)^k)*((c-x)^j));
         end
```

```
        end
    end
end
function Nmoment = normalMoment(image,x,y,m00,cm20,cm02,j,k,d)
    [row,col] = size(image);
    Nmoment = 0;
    sigx = sqrt(cm20/m00);
    sigy = sqrt(cm02/m00);
    for r = 1:row
        for c = 1:col
            if image(r,c) == d
                Nmoment = Nmoment + ((((r-y)/sigy)^k)*(((c-x)/sigx)^j));
            end
        end
    end
end
function thresh = threshold(image)
    histC = histcount(image);

    histS = sum(histC);
    sumA = dot((0:255),histC);
    sumB = 0;
    wB = 0;
    max = 0.0;
    for i= 1:256
        wB = wB + histC(i);
        if(wB == 0)
            continue;
        end
        wF = histS - wB;
        if(wF == 0)
            break;
        end
        sumB = sumB + (i-1) * histC(i);
        mB = sumB/wB;
        mF = (sumA - sumB)/wF;
        between = wB * wF * (mB - mF) * (mB - mF);
        if(between >= max)
            thresh = i;
            max = between;
        end
    end
    thresh = thresh/256;
```

```
end
function hist=histcount(w)
[H,W] = size(w);
    w = uint8(w);
    hist=zeros(1,256);
    for i = 1:H
        for j = 1:W
            hist(w(i,j)+1)=hist(w(i,j)+1)+1;
        end
    end
    hist=hist/(H*W);
end
function lab = label(conI,row,col)
    global marker;
    global mark;
    global s;
    mark(row,col) = marker;
    for r = row-1:row+1
        for c = col-1:col+1
            if r ~= row || c ~= col
                if r > 2 && r < s(1)- 2 && c > 2 && c < s(2) - 2
                    if conI(r,c) == 1 && mark(r,c) == 0
                        label(conI,r,c);
                    end
                end
            end
        end
    end
end
```

```
I = imread('toy.png');
F = imread('filter.jpg');
FR = imrotate(F,180);
FR2 = im2double(FR);
I2 = im2double(I);
It = I2 - mean2(I2);
FRt = FR2 - mean2(FR2);
If = conv2(It,FRt,'same');
colormap jet
maxValue = max(If(:));
[rMax, cMax] = find(If == maxValue);
filtS = size(FR);
%imshow(I);
hold on;
rectangle('Position',[cMax(1)-(filtS(2)/2),rMax(1)-(filtS(1)/2),filtS(2),filtS(1)]);
rectangle('Position',[cMax(2)-(filtS(2)/2),rMax(2)-(filtS(1)/2),filtS(2),filtS(1)]);
rectangle('Position',[cMax(3)-(filtS(2)/2),rMax(3)-(filtS(1)/2),filtS(2),filtS(1)]);
Pos = [cMax(1)-(filtS(2)/2),rMax(1)-(filtS(1)/2),filtS(2),filtS(1)];
ground1 = [cMax(1)-(filtS(2)/2)-5,rMax(1)-(filtS(1)/2)-5,filtS(2)+10,filtS(1)+10];
err1 = boxerr(Pos,ground1)

ground2 = [cMax(1)-(filtS(2)/2)-5,rMax(1)-5-(filtS(1)/2),filtS(2),filtS(1)];
err2 = boxerr(Pos,ground2)

ground3 = [cMax(1)-(filtS(2)/2)+5,rMax(1)+5-(filtS(1)/2),filtS(2)+10,filtS(1)+10];
err3 = boxerr(Pos,ground3)


function err = boxerr(Pos,ground1)
   overlap1 = rectint(Pos,ground1);
   nooverlap1 = Pos(3)*Pos(4) + ground1(3)*ground1(4) - overlap1;
   err = overlap1/nooverlap1;
end
```

```
I = imread('car3.jpg');
F = imread('cartemplate.jpg');
Fsize = size(F);
F = imresize(F, [.09*Fsize(1),.09*Fsize(2)],'bilinear');
FR = imrotate(F,180);
FR2 = im2double(FR);
I2 = im2double(I);
It = I2 - mean2(I2);
FRt = FR2 - mean2(FR2);
If = conv2(It,FRt,'same');
colormap jet
maxValue = max(If(:));
[rMax, cMax] = find(If == ma
xValue);
filtS = size(FR);
%hold on;
imagesc(If);
%imshow(I);
Pos = [cMax(1)-(filtS(2)/2),rMax(1)-(filtS(1)/2),filtS(2),filtS(1)];
ground = [329, 251, 480-329, 345-251];
%rectangle('Position',ground,'EdgeColor','green');
%rectangle('Position',Pos,'EdgeColor','blue');
err = boxerr(Pos,ground)
function err = boxerr(Pos,ground1)
   overlap1 = rectint(Pos,ground1);
   nooverlap1 = Pos(3)*Pos(4) + ground1(3)*ground1(4) - overlap1;
   err = overlap1/nooverlap1;
end
```