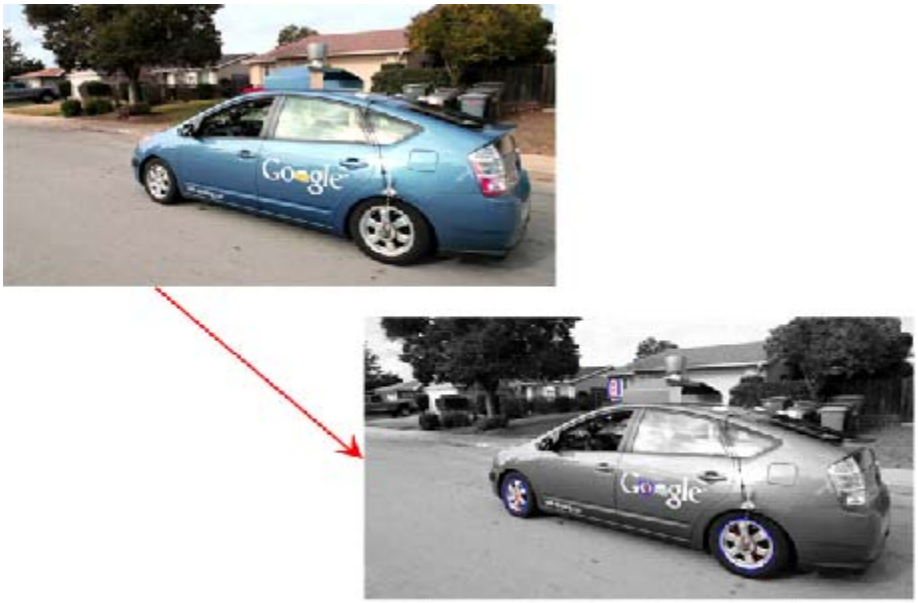
ELLIPSE DETECTION IN A TWO-DIMENSIONAL IMAGE
CASE STUDY: VEHICLE DETECTION

Mehrdad Ghyabi



Spring 2018

Contents

Motivation

As a relatively new approach, vision based structural health monitoring (SHM) aims to evaluate current conditions of aging structures by analyzing digital images. This technique could be especially useful in case of structures for which other conventional SHM methods are not feasible. The idea here is to print circular patterns (Figure 1) on various locations of the structure at hand. Knowing that under loading, a circle would deform into an ellipse, the strains and stresses of different structural members can be calculated under different loading conditions.

Figure 1- Sample Pattern

The first step of the vision based SHM process is to detect ellipses in a given digital image, which is the motivation of this project. This problem has been solved successfully during recent years by researchers [1][2][3][4][5] in other fields, it has been proved to be useful in various fields like traffic engineering (car wheel detection), urban engineering (feature detection), etc.

Abstract

The goal of this project is to implement ellipse detection algorithms which are presented in [1]. Addressing the problems in this project required identifying specific object (such as cars) and their location in a three-dimensional space using a two-dimensional image. This project consists of two steps. First ellipses, which are images of 3D circles on a 2D plane if not looked at frontally, are identified using a novel procedure that exploits intensity difference between objects in the image. 2D ellipses are then matched to 3D circles and their relative location are identified using a geometrical algorithm. These steps are used then to detect car wheels, which are circles in 3D space and ellipses in 2D images. This framework can be utilized instead of developing depth images, which needs substantial computational effort and works in specific scene conditions only. Results of this framework can be used for scene reconstruction. Instead of using an edge detection algorithm to detect ellipses, a statistical framework is developed which allows for significant noise on the edges.

Literature Review

In 1991, Safaee-Rad [2] compared the performance of four techniques for estimating parameters of elliptical patterns in an image, namely Weighted minimum-square error (MSE), application of Fourier descriptors, moment of area, and moment of perimeter. It was concluded that selecting a technique depends on computational cost, number of parameters to be estimated, the required degree of accuracy and the specific conditions under which the estimation must be performed.

Thirteen years later, in 2004, Kanatani [4] proposed a method to detect circular objects in an image. Using Hugh transform, an osculating circle to an elliptic arc is found, and then is transformed into an ellipse by iterating deformations. It is worth noting that voting space for Hugh transform, in this particular process, is restricted to one and two dimensions (Figure 2).

Figure 2

In 2007, Chia [5] exploited Hugh transform in one-dimensional parametric space to detect ellipses in images. Although Hugh transform is a robust method, its storage and computational requirements increase the difficulties in practical effectiveness. However, implementing the procedure with a one-dimensional accumulator reduces storage requirements. A sample set of results of this method is presented in Figure 3.

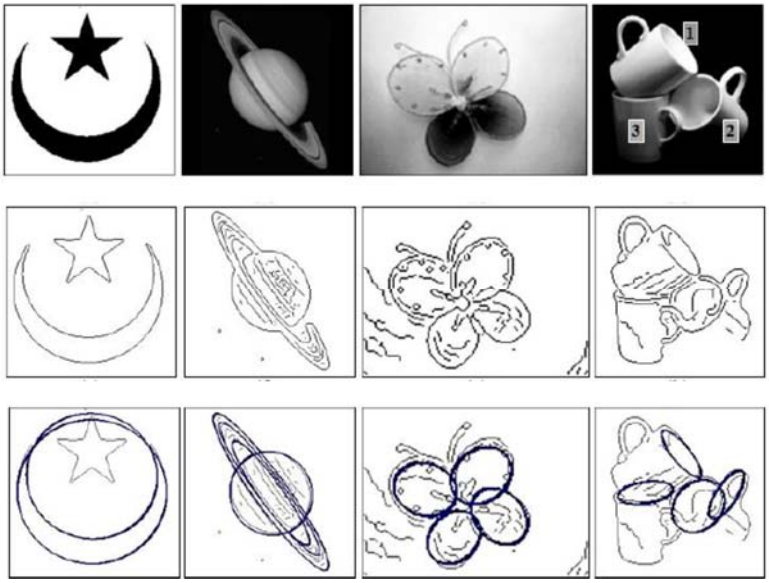


Figure 3- Hugh Transform Results

The following year, Mai [3] presented a hierarchical approach for ellipse extraction. After processing an image into a set of edge pixels, line segments which are potential candidates of elliptic arcs are linked to create arc segments. Then, segments from same ellipse are grouped together and an ellipse is ultimately fitted to each group using a robust statistical method called RANSAC (Figure 4).

Figure 4- Result from RANSAC

Implementation of procedure

Detection of ellipses will be performed in four distinct steps:

Bright Pixel Detection

These steps will be implemented on bright pixels, however, dark pixels or pixels with any other intensity property could be distinguished using this framework.

In order to identify pixels which are brighter than their surrounding area, a new image which reflects the local mean intensity for each pixel is built. Each pixel in this new image contains the mean intensity for a square region centered on that pixel in the original image. This square is set to b% of the shortest image side. For the purposes of this project, a window 10% of the width of the image was found to be effective, but this can depend on the nature of the ellipses that are desired. If this window would extend beyond the edges of the image for a particular pixel, the mean of the closest possible window which is entirely contained within the image is used instead. The intensity integral image is used to efficiently calculate this windowed mean value, which was the primary motivation for using a simple box kernel for smoothing (Figure 5).

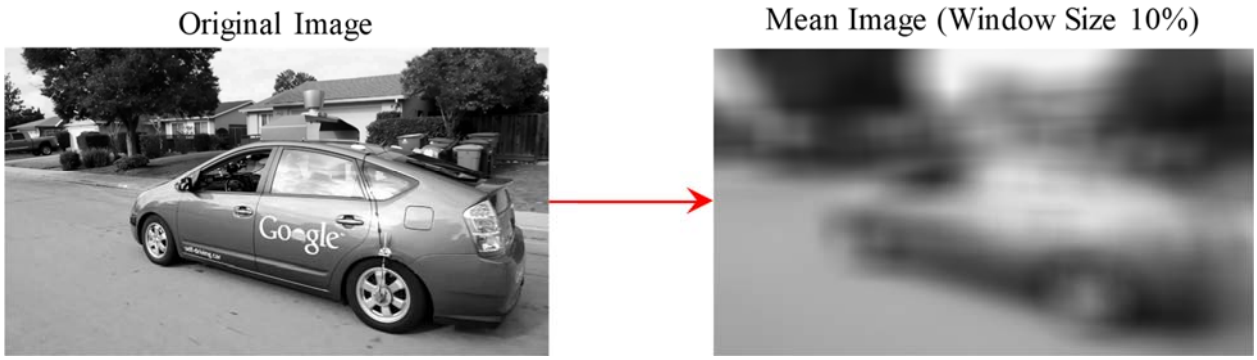


Figure 5- Smoothing

Identifying bright pixels is done using an averaging filter, and a mean image is built showing the average intensity of a square region centered on every pixel. Depending on the nature of the ellipse of interest, the size of the averaging filter is set to 10% of the width of images. The difference with a conventional averaging operation is that in this case, if the filter goes beyond the edges of image, it only averages the pixels that are inside the image, hence no padding is performed [6].

After that, the mean image is subtracted from the original image (Figure 6) and then a threshold T is set to define the amount of difference in intensity between a pixel and its local mean in order for it to be called bright. In this project, T will be set one standard deviation above the mean of the average image. The result of this step is a binary image in which “ones” are showing bright pixels and “zeros” are representing dark pixels.

Figure 6- Subtracted Image

This threshold method tends to produce a thick `edge' around the ellipse perimeter as seen in Figure 7, and frequently contains small gaps corresponding to darker areas of the ellipse surface, corresponding to occlusions, decoration or simply image noise.

Figure 7- Result of Different Thresholds

Connected components

To handle noise in image resulting from the previous step, first the small bright regions are omitted, after that 8-adjacencies are detected by convolving the image by kernel shown in Equation 1 (Figure 8). Then each 8-connected region will be named by a unique identifier (Figure 9).

Equation 1

$$Kernel: \frac{1}{8} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 8- Image after Omission of Small Blobs

Figure 9- 8-Adjacencies Detected

Filling the Gaps

In this project, ellipse parameters are computed using a statistical method, hence, identified objects must be filled. As it is important to preserve the overall shape of the blobs, filling procedure should keep the perimeter intact and fills gaps in a radial manner. If a pixel in a blob is labeled bright, then all the pixels on a line between it and center of the blob will be marked bright as well. This method works well because blobs are not completely closed due to presence of noises. Each of these sections should be checked to see if they are of elliptical nature (Figure 10).

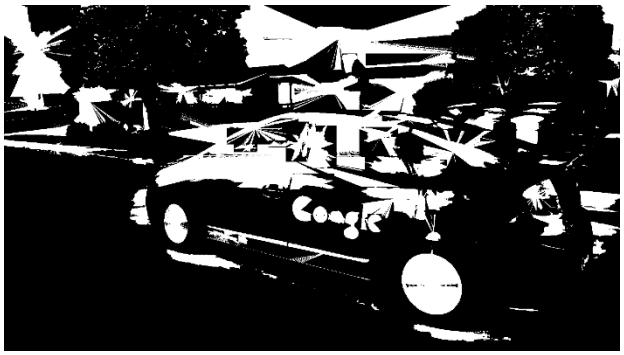


Figure 10- Image with Radially Filled Blobs

Ellipse Identification

Extracting blob properties is done by finding mean and covariance matrix of each blob, and with that done, blobs can be compared it to their equivalent ellipses. Area, mean and covariance matrix of blobs can be found using Equation 2 to Equation 4:

Equation 2

$$|W| = \sum_{p \in W} 1$$

Equation 3

$$\mu = \frac{1}{|W|} \sum_{p \in W} p$$

Equation 4

$$C = \frac{1}{|W|} \sum_{p \in W} (p - \mu)(p - \mu)^T$$

In which |W| is the surface area, μ is centroid and C is covariance matrix of the blob.

The center of the ellipse would be μ and the major and minor axis of the ellipse can be calculated from eigenvalues of covariance matrix. At this point, some obvious non-ellipse blobs can be detected using following inequity:

$$\left| |W| - 4\pi\sqrt{detC} \right| > T$$

Then the ratio of mismatching pixel amount to blob area, can be used to approve or disapprove the remaining blobs. In this project a mismatch of up to 5% is considered acceptable. We can then directly compare the remaining blobs to the equivalent ellipse on a per pixel basis by counting the number of pixels in which the blob and its ellipse do not match. As we are interested in the elliptical nature of the blob regardless of size, we divide the number of mismatches by the blob area. Blobs with a sufficiently low mismatch can be regarded as ellipses. We found experimentally that a mismatch ratio of 5% identifies human-visible ellipses while discarding non-ellipses (Figure 11).

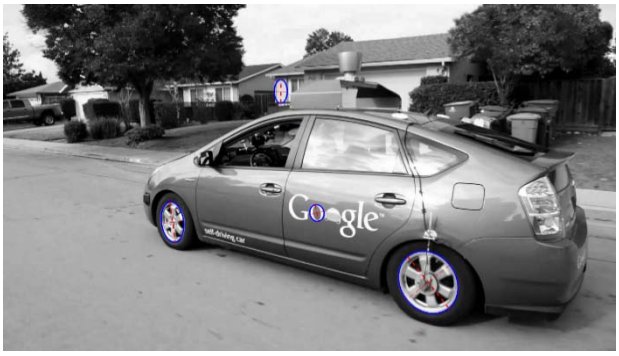


Figure 11- Final Result

To map 2D Ellipse to a 3D circle, an ellipse can be described using its covariance matrix:

$$E(\mu, C) := \{p \in \mathbb{R}^2 : (p-\mu)C^{-1}(p-\mu)^T \leq 4\}$$

And the values of its greater and smaller radii can be calculated from the equations below:

$$a_1 = 2\sqrt{\lambda_1} \text{ and } a_2 = 2\sqrt{\lambda_2}$$

In which λ_1 and λ_2 are eigenvalues of covariance matrix and $\lambda_1 > \lambda_2$ Transforming a 3D circle to a 2D ellipse: Unit normal vector of a circle in 3D space will be defined as follows:

$$\varphi \equiv \begin{pmatrix} \varphi_x \\ \varphi_y \\ \varphi_z \end{pmatrix} = \frac{1}{a_1} \begin{pmatrix} \pm\sqrt{{a_1}^2 - 4C_{xx}} \\ \pm\sqrt{{a_1}^2 - 4C_{yy}} \\ \pm a_2 \end{pmatrix}$$

Input Data for 2D detection

To use this method, 2D images are needed in which, elliptical patterns are detectable. These elliptical patterns can usually be found easily; they could be car wheels or circular patterns in buildings and urban areas. They could even be printed objects so that different experiments can be ran in a controlled environment.

In case of vehicles, the arrangement of ellipses in the image can help to identify location and orientations of the vehicle. With a few assumptions like the relative size of preferred ellipses to the size of image, this method improves.

Method Advantages

Framework of this project can be used in many scientific fields. Specifically, it will be used to detect circular (or elliptical) patterns printed on structures which is very useful in vision-based structural health monitoring. Comparing to conventional methods in this field, this method is very promising. A widely implemented vision-based method in this field is “Digital Image Correlation (DIC)”, which uses fixed cameras to capture images from elements in structures. The main limitation of DIC is that, the location of camera cannot be altered during tests, making it impractical in case of structures, in which both specimen size and time length is very large.

More Results

In Figure 12 the front wheel image is blocked, hence the algorithm was unable to detect it.

Figure 12

In Figure 13 the rear wheel was not detected because of its strange shape (multiple ellipses intersecting with each other).

Figure 13

In Figure 14 ellipses are not properly fitted to the wheels because of poor lighting conditions.

Figure 14

In Figure 15 front wheel was not detected because of presence of a white blob on the front tire.

Figure 15

MATLAB Codes

Smoothing and Bright Spotting

```
   AveWinSizeRatio=10; %ratio of size of averaging window size
   m1=round(m*AveWinSizeRatio/200);
   n1=round(n*AveWinSizeRatio/200);

   for i=1:m
       for j=1:n
           x1=i-m1;
           x2=i+m1;
           y1=j-n1;
           y2=j+n1;
           x1(x1<1)=1;
           y1(y1<1)=1;
           x2(x2>m)=m;
           y2(y2>n)=n;
           b=a(x1:x2,y1:y2);
           a1(i,j)=mean(b(:));
       end
   end
a2=a-a1;
a2(a2<0)=0;
Treshold=mean(a2(:))+0.5*std(a2(:)); %threshold for bright spots
a2(a2<Treshold)=0;
a2(a2~=0)=1;
```

8-Adjacencies

```
   kernel=[1 1 1; 1 0 1; 1 1 1]/8;
   a3=conv2(a2,kernel,'same')==1;

   data1=regionprops(a3,'centroid','BoundingBox','ConvexImage','MajorAxisLeng
   th','MinorAxisLength','Orientation');
   n=1;
   for i=1:numel(data1)
       x=data1(i).BoundingBox;
       if x(3)<50 && x(4)<50
           y(n)=i;
           n=n+1;
       end
   end
   data1(y)=[];

   hold on
   for i=1:numel(data1)
       rectangle('Position',round(data1(i).BoundingBox),'EdgeColor','r')
   end
```

Radial Filling

```
   for k=1:numel(data1)
       box=round(data1(k).BoundingBox);
       abox=a3(box(2):box(2)+box(4),box(1):box(1)+box(3));
```

```
    [xcrd,ycrd]=find(abox==1);
    crd=[xcrd ycrd];
    center=round(mean(crd));

    m=(ycrd-center(2))./(xcrd-center(1));
    for i=1:numel(xcrd)
        if xcrd(i)~=center(1)
            x=min([xcrd(i) center(1)]):max([xcrd(i) center(1)]);
            y=round(m(i)*(x-center(1))+center(2));
            for j=1:numel(x)
                abox(x(j),y(j))=1;
            end
        else
            y=min([ycrd(i) center(2)]):max([ycrd(i) center(2)]);
            for j=1:numel(y)
                abox(xcrd(i),y(j))=1;
            end
        end
    end
    a4(box(2):box(2)+box(4),box(1):box(1)+box(3))=abox;
end

data2=regionprops(a4,'centroid','BoundingBox','ConvexImage','MajorAxisLeng
th','MinorAxisLength','Orientation');
n=1;
for i=1:numel(data2)
    x=data2(i).BoundingBox;
    if x(3)<50 && x(4)<50
        z(n)=i;
        n=n+1;
    end
end
data2(z)=[];

hold on
for i=1:numel(data2)
    rectangle('Position',round(data2(i).BoundingBox),'EdgeColor','r')
end
```

Ellipse Check

```
n=1;
data3=data2;
for k=1:numel(data2)
    box=round(data2(k).BoundingBox);
    abox=a4(box(2):box(2)+box(4),box(1):box(1)+box(3));
    area1=sum(sum(abox));
    area2=0.25*pi*data2(k).MajorAxisLength*data2(k).MinorAxisLength;
    if abs(area2-area1)/area1>0.20
        z1(n)=k;
        n=n+1;
    end
end
data3(z1)=[];
```

```
hold on
for i=1:numel(data3)
    rectangle('Position',round(data3(i).BoundingBox),'EdgeColor','r')
end
```

Ellipse Drawing

```
theta = 0:(pi/64):(2*pi);
figure,
imshow(a4)
hold on
 for k=1:numel(data3)
     ax1=[0 data3(k).MinorAxisLength/2];
     ax2=[data3(k).MajorAxisLength/2 0];
     alpha=-pi*data3(k).Orientation/180;



     x1=ax1(1)*cos(theta)+ax2(1)*sin(theta);
     y1=ax1(2)*cos(theta)+ax2(2)*sin(theta);
     cent=data3(k).Centroid;
     x=x1*cos(alpha)-y1*sin(alpha)+cent(1);
     y=y1*cos(alpha)+x1*sin(alpha)+cent(2);
     plot(x,y,'b','LineWidth',2)

     yl1=[-data3(k).MinorAxisLength/2 data3(k).MinorAxisLength/2];
     xl1=[0 0];
     xL1=xl1*cos(alpha)-yl1*sin(alpha)+cent(1);
     yL1=yl1*cos(alpha)+xl1*sin(alpha)+cent(2);

     xl2=[-data3(k).MajorAxisLength/2 data3(k).MajorAxisLength/2];
     yl2=[0 0];
     xL2=xl2*cos(alpha)-yl2*sin(alpha)+cent(1);
     yL2=yl2*cos(alpha)+xl2*sin(alpha)+cent(2);

     plot(xL1,yL1,'--r')
     plot(xL2,yL2,'--r','LineWidth',2)
 end
figure,
imshow(a)
hold on
 for k=1:numel(data3)
     ax1=[0 data3(k).MinorAxisLength/2];
     ax2=[data3(k).MajorAxisLength/2 0];
     alpha=-pi*data3(k).Orientation/180;



     x1=ax1(1)*cos(theta)+ax2(1)*sin(theta);
     y1=ax1(2)*cos(theta)+ax2(2)*sin(theta);
     cent=data3(k).Centroid;
     x=x1*cos(alpha)-y1*sin(alpha)+cent(1);
     y=y1*cos(alpha)+x1*sin(alpha)+cent(2);
```

```
        plot(x,y,'b','LineWidth',2)

        yl1=[-data3(k).MinorAxisLength/2 data3(k).MinorAxisLength/2];
        xl1=[0 0];
        xL1=xl1*cos(alpha)-yl1*sin(alpha)+cent(1);
        yL1=yl1*cos(alpha)+xl1*sin(alpha)+cent(2);

        xl2=[-data3(k).MajorAxisLength/2 data3(k).MajorAxisLength/2];
        yl2=[0 0];
        xL2=xl2*cos(alpha)-yl2*sin(alpha)+cent(1);
        yL2=yl2*cos(alpha)+xl2*sin(alpha)+cent(2);

        plot(xL1,yL1,'--r')
        plot(xL2,yL2,'--r','LineWidth',2)
    end
```

References

[1] M. Hutter and N. Brewer, "Matching 2-D ellipses to 3-D circles with application to vehicle pose identification," *2009 24th Int. Conf. Image Vis. Comput. New Zealand, IVCNZ 2009 - Conf. Proc.*, pp. 153–158, 2009.

[2] R. Safaee-Rad, K. C. Smith, B. Benhabib, and I. Tchoukanov, "Application of moment and Fourier descriptors to the accurate estimation of elliptical shape parameters," *[Proceedings] ICASSP 91 1991 Int. Conf. Acoust. Speech, Signal Process.*, pp. 2465–2468 vol.4, 1991.

[3] F. Mai, Y. S. Hung, H. Zhong, and W. F. Sze, "A hierarchical approach for fast and robust ellipse extraction," *Icip*, no. 852, p. V-345-V-348, 2007.

[4] K. Kanatani and N. Ohta, "By Ellipse Growing," vol. 4, no. 1, pp. 35–50, 2004.

[5] "ELLIPSE DETECTION WITH HOUGH TRANSFORM IN ONE DIMENSIONAL PARAMETRIC SPACE Alex Yong Sang Chia , Maylor K . H . Leung School of Computer Engineering Nanyang Technological University Nanyang Avenue , Singapore 639798 How-Lung Eng , Susanto Rahardja Institu," *Comput. Eng.*, pp. 333–336, 2007.

[6] Gonzalez, R.C. and Woods, R.E., 2002. Digital image processing