

COT 4930 COT 5930 EEL 4930 EEL 5661 Robotic Applications Fall 2022

Homework 4
Machine Vision

Shaun Pritchard

Choose any 4 (out of 6 questions) for the homework's 20 (G) or 25 (UG) points. There is a bonus of up to 2.5 points for each additional problem solved. The maximum grade for HW4 can reach 25 (G) or 30 (UG).

Most problems feature an analytical part, involving small synthetic images, taken from the Fall 2021 final exam. The image in each such problem has origin (0,0) is in the upper left corner. The u-axis goes from left to right, and the v-axis goes from top to bottom. Every one of these problems involves some analysis of a real image, as well.

Problem 1: A perspective camera has a focal length of $f = 14$ [mm]. Choose some reasonable values for the size of a pixel and for the number of pixels (W and H). A $10 \times 10 \times 10$ [cm] box is standing parallel to the camera at a distance of 2.5 [m] along the optical axis.

How long (in [mm]) is an edge of the box as it is showed in the image? Explain your calculations and demonstrate using MATLAB Machine Vision Toolbox (MVTB). Plot how the box shows up on the image of the camera.

Essentially $f=14\text{mm}$ would be $F/2.8$ aperture where the optical axis converges $P^*=CP^*=(CH-1)(CH-1)=C1P1$ the homogeneous transformation shows that the point of XYZ is pre multiplied with respect to the objects edge. Parallel lines lie on the plane that is parallel to the image plane remain parallel. By converting the points of the corners and then get the distance between them, that's the length of the edge

```
cam = CentralCamera('focal', 0.014, 'pixel', 10e-6, 'resolution', [1280 1024], 'centre', [640 512], 'name', problem1)
```

```
name: problem1 [central-perspective]
```

```
  focal length:  0.014
```

```
  pixel size:   (1e-05, 1e-05)
```

```
  principal pt: (640, 512)
```

```
  number pixels: 1280 x 1024
```

```
  pose:         t = (0, 0, 0), RPY/yxz = (0, 0, 0) deg
```

```
Tcam =
```

```
    1    0    0    0
```

```
    0    1    0    0
```

```
    0    0    1    1
```

```
    0    0    0    1
```

```
cam = CentralCamera('focal', 0.014, 'pixel', 10e-6, 'resolution', [1280 1024], 'centre', [640 512], 'name', 'mycamera')
```

```
Tcam=SE3(0,0,1.0)
```

```
%P = mkgrid(3, 0.2, 'pose', Tcam);
```

```
%P(:,1:4)
```

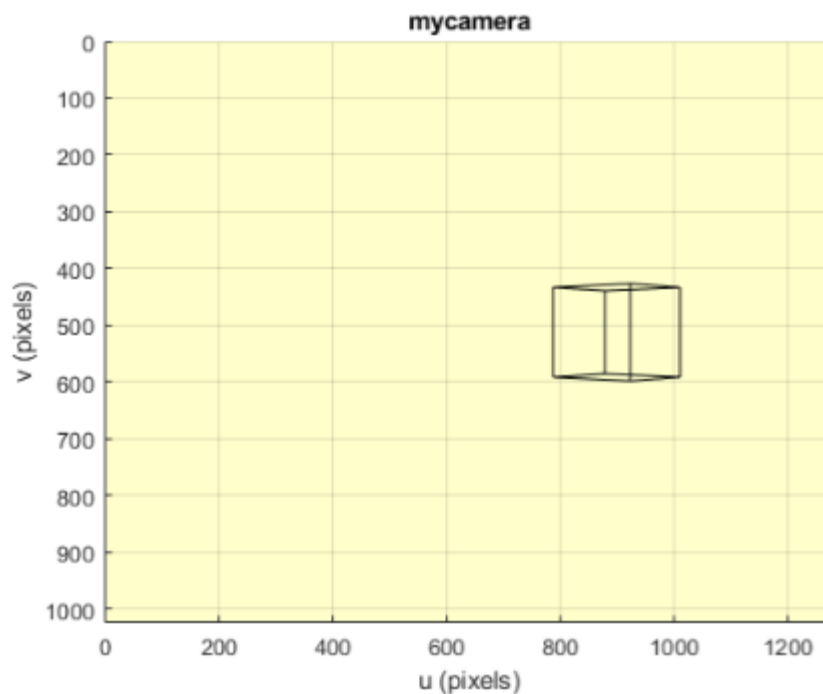
```

[X,Y,Z]=mkcube(0.25,'pose',SE3(0,0,1), 'edge');
%cam.fov() *180/pi
%cube=mkcube(0.2, 'pose',SE3(0,0,1), 'edge')
Tcam=SE3(-0.75,0,0.25)*SE3.Ry(0.8);
cam.mesh(X,Y,Z, 'pose',Tcam)

creating new figure for camera
h =
  Axes (mycamera) with properties:
      XLim: [0 1280]
      YLim: [0 1024]
      XScale: 'linear'
      YScale: 'linear'
      GridLineStyle: '-'
      Position: [0.1300 0.1100 0.7750 0.8150]
      Units: 'normalized'

Show all properties
make axes

```



name: noname [central-perspective]

focal length:

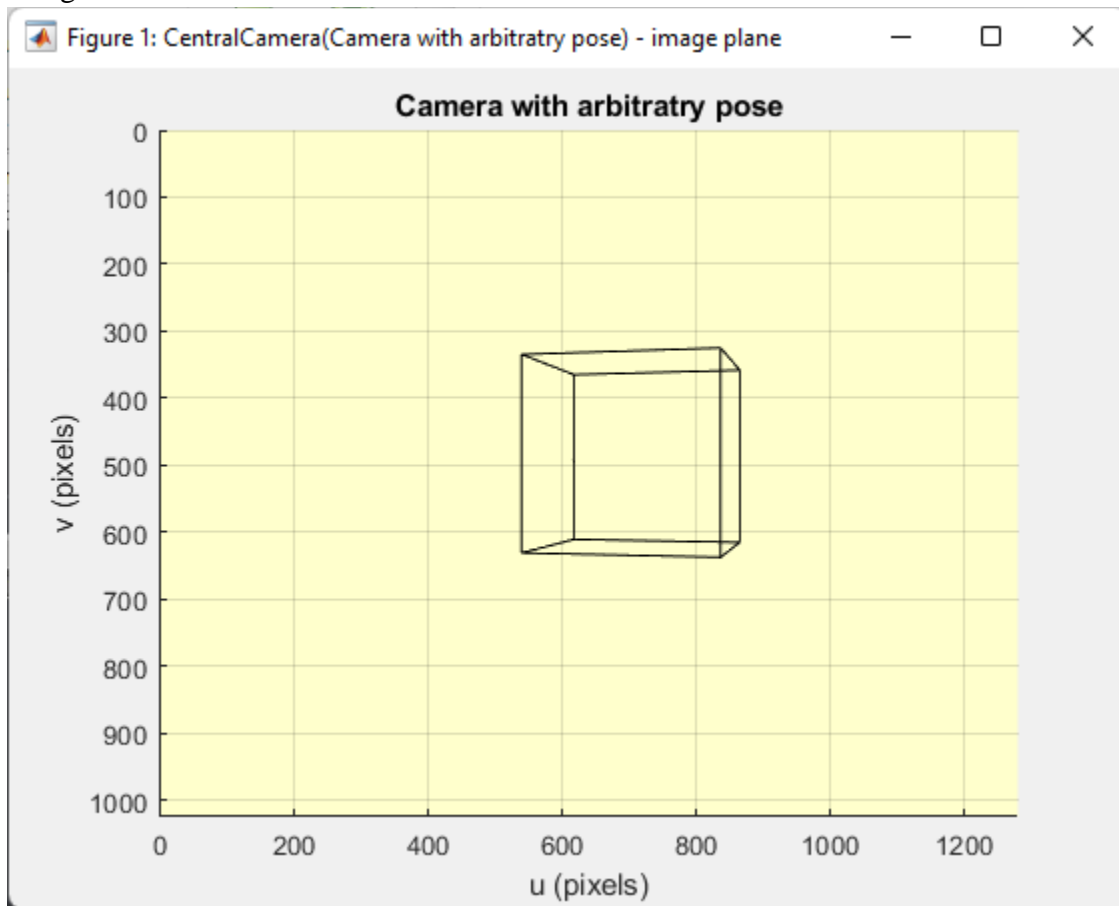
pixel size: (1, 1)

principal pt: (0, 0)

pose: $t = (0, 0, 0)$, RPY/yxz = (0, 0, 0) deg

If the box is displaced along the world's X axis, how long can it be displaced before leaving the camera's field of view? Demonstrate using MVTB.

The camera is 2.5m straight back along the optical axis (Z-axis). using the CentralCamera subclass, we dont have to set the position of the camera. when we make the cube with mkcube(), we pass a pose parameter with an SE3 transform of the X,Y,Z position of the cube with respect to the camera. using Tcam to further offset the camera.



Problem 2:

2.1 An image of 3 x 3 pixels has 4 grey levels {0, 1, 2, 3}. It is known that the binary image has 2 labels and a corner point. The greyscale image histogram is given by the table:

# Pixels	4	1	0	4
Grey level	0	1	2	3

Create the original image (there can be many correct solutions) and explain briefly.

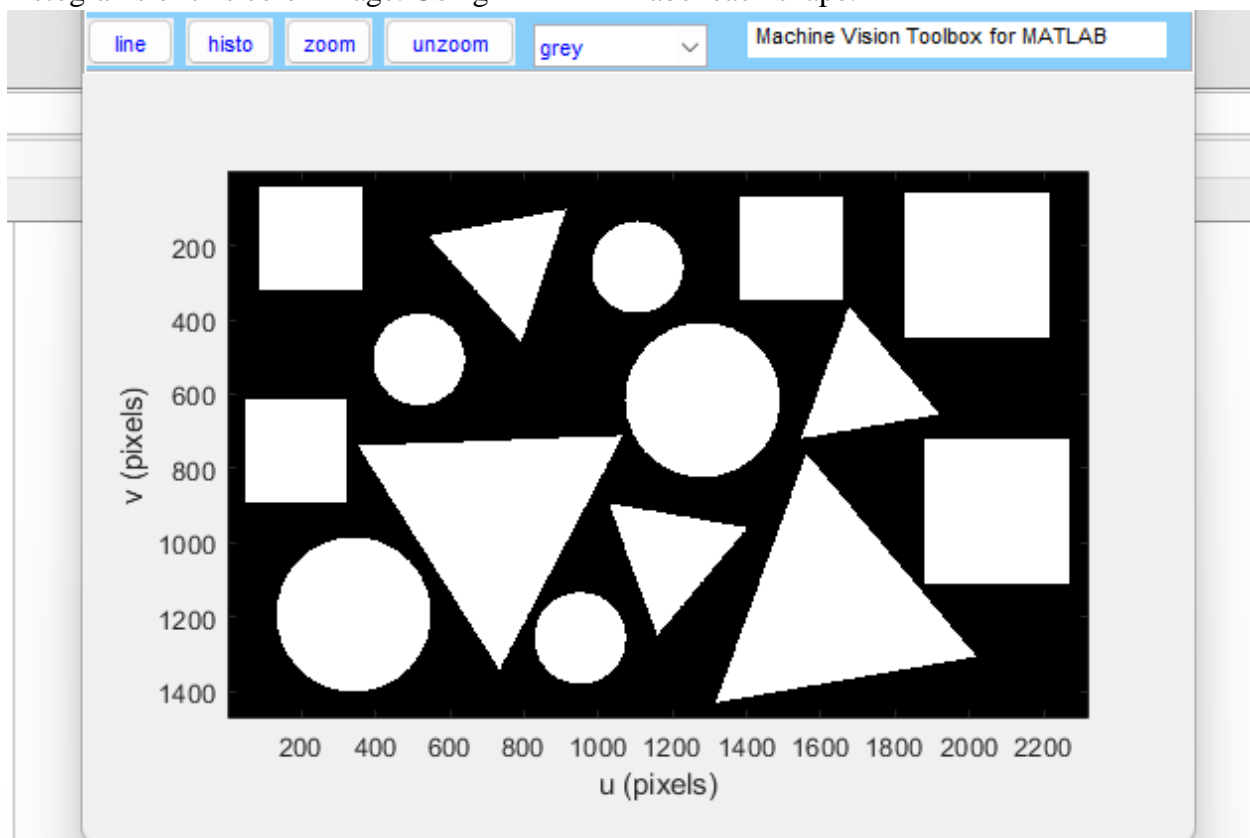
3	3	1
2	2	1
1	1	2

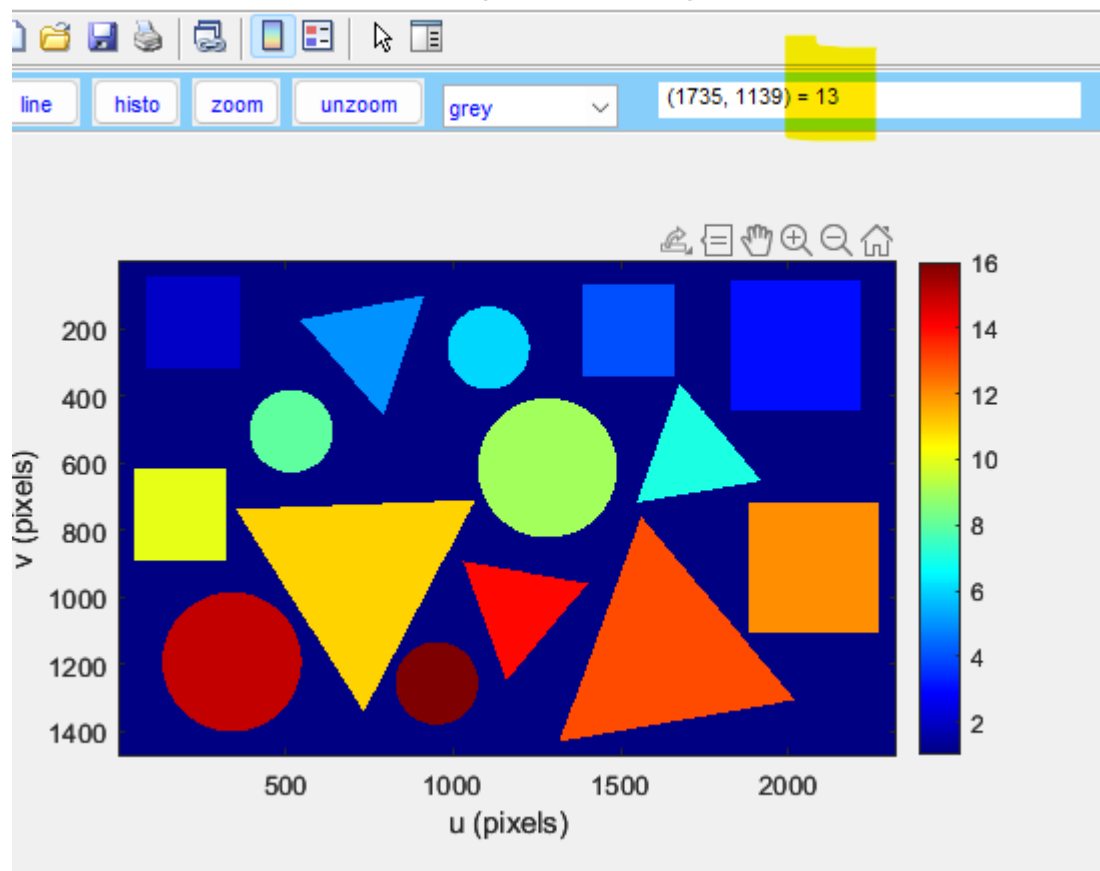
Create a binary image of the original image and explain briefly how you obtained it.

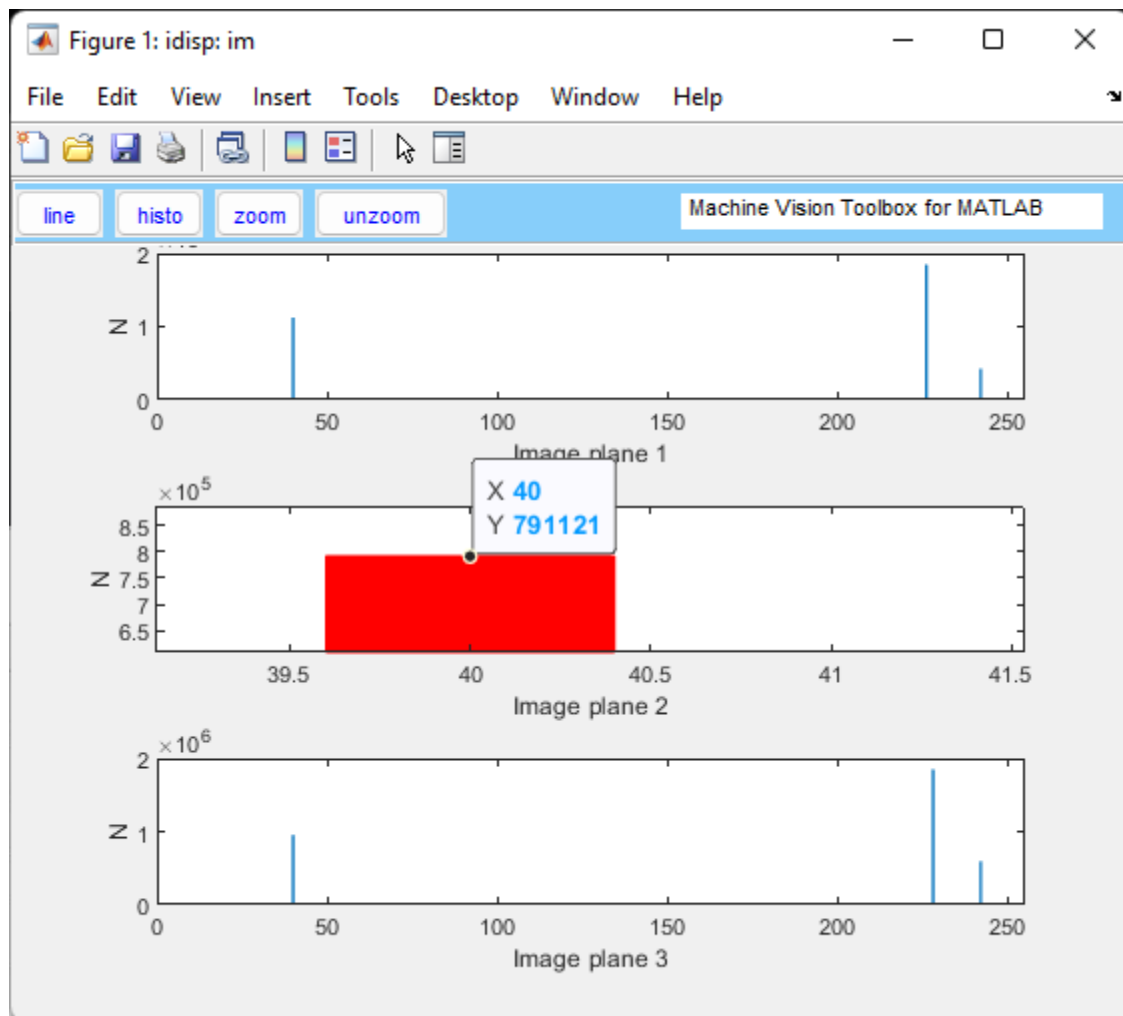
An `im2bw` command converts an image to a binary image, based on the threshold. Binary images are images in which each pixel has only two possible intensities. In terms of numerical values, the two values are usually 0 for black, and either 1 or 255 for white. The process of information abstraction is used to obtain a binary image from a grayscale image based on the process of information abstraction. Thresholding is the main techniques used at this stage. Each white pixel with a unique integer number. We number the pixels systematically moving from the top row to bottom row with increasing integer values along the columns. Using labeling techniques, we iterate through each pixel. At each iteration we call the present pixel 'i'. We look at the direct neighbors of our 'i' pixel. If the integer value assigned to the neighbors is smaller than the present value of 'i', then 'i' will take the integer value of the smallest neighbor. We implement this twice

0	0	1
0	0	1
1	1	0

2.2 Load and display the color MVTB image named "shapes.png". Plot and explain the three histograms of this color image. Using MATLAB label each shape.







Problem 3:

3.1 Create a 6 x 6 binary image that has 4 labeled regions. Region 1 is the parent of the other three 2-pixels white regions. Explain briefly.

There are 3 regions are black with many variations of other solutions, All pixels scanned along row 1 are labeled 1, so forth as we get to the next block pixel we implement sets for label to and so on assigning sets based on the regions in the rows.

1	1	1	1	1	1
1	0	0	1	0	1
1	1	1	1	0	1
1	0	0	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1

3.2 Pick up one of the white regions of (3.1) and calculate its m02 moment.

Choose the smallest region

> m00 = mpq(blob, 0, 0) m00 = 7728

White region (above): Area = m00 = 19

1st moment 10:(sum along rows)

m10=(0x1+1x1+2x1+3x1+4x1)+(0x1+1x0+2x0+3x1)+(0x1+4x1)+(0x1+2x1+3x1+4x1)+(0x1+1x1+2x1)= 34

A corner is detected when the minimum change produced by any of the shifts is larger than a certain threshold. we can represent the change E produced by a shift (u, v) as where w is the image window whose value is 1 within a specified rectangular region, and 0 elsewhere

3.3 Load and display the BW MVTB image titled “shark2.png”. Using moments analysis find the position and orientation of each of the two sharks.

Position: a = 70.43313 ; b = 39.2663 = [-0.857001074601707;0.515314620529555]

Orientation: 184.7285 x 503.4981

If we consider that the image region is thin plane where each unit of area(pixel has 1pt are and 1pt mass, where the total mass of a region is identified as m00 we can calculate the center of mass centroid coordinates with the moment normalized by total mass. Central moments are invariant to the position of the region.

Where $\mu_{pq} = \sum_{(u,v) \in I} (u - u_c)^p (v - v_c)^q I[u, v]$ we can then calculate the equivalent of the bounding ellipse to the object's orientation where $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ It is centered at the origin of the x,y plane as depicted below.

Code:


















```
sharks=imread('sharks.png'); %import image
%idisp(sharks)
[label,m]=ilabel(sharks); %Set labels
blob=(label==2);
idisp(blob)
hold on
m00=mpq(blob,0,0);
uc=mpq(blob,1,0)/m00; %Calulate centriod uc=m10/m00
vc=mpq(blob,0,1)/m00; %Calulate centriod vc=m01/m00
plot(uc,vc,'gx',uc,vc,'go'); %plot centriod
hold on
u20=upq(blob,2,0); % Diagnoal matrix u20
u02=upq(blob,0,2); % Diagnoal matrix u02
u11=upq(blob,1,1);
J=[u20 u11; u11 u02]; % calculate eingvalues of J
%plot_ellipse(4*J/m00, [uc,vc], 'y')
lambda=eig(J);
a=2*sqrt(lambda(2)/m00); % calualte square of J for position a
b=2*sqrt(lambda(1)/m00); % calualte square of J for position b
b/a;
[x,lambda]=eig(J);
```



```

x;
v=x(:,end); % calculate X coordinate and y coordinate
atand(v(2)/v(1))

```

Name ▲	Value
 a	70.4313
 ans	-31.0185
 b	39.2663
 blob	511x603 logical
 J	[7.8299e+06,-2.9169e...
 label	511x603 double
 lambda	[2.9788e+06,0;0,9.583...
 m	5
 m00	7728
 sharks	511x603 logical
 u02	4.7328e+06
 u11	-2.9169e+06
 u20	7.8299e+06
 uc	503.4981
 v	[-0.8570;0.5153]
 vc	184.7285
 x	[-0.5153,-0.8570;-0.85...

```

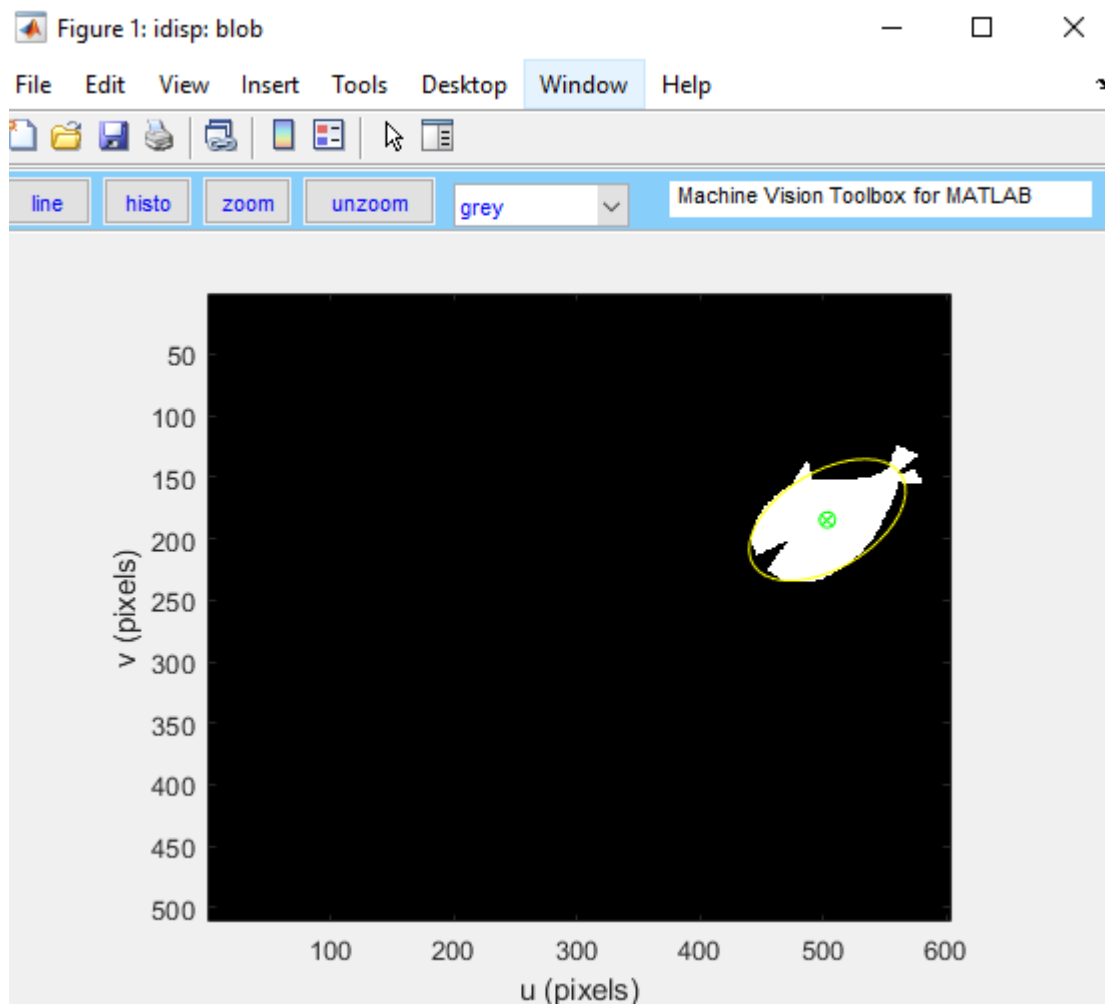
sharks=imread('sharks.png');
%idisp(sharks)
[label,m]=ilabel(sharks);
blob=(label==2);
idisp(blob)
hold on
m00=mpq(blob,0,0);
uc=mpq(blob,1,0)/m00;
vc=mpq(blob,0,1)/m00;
plot(uc,vc,'gx',uc,vc,'go');
hold on
u20=upq(blob,2,0);
u02=upq(blob,0,2);
u11=upq(blob,1,1);
J=[u20 u11; u11 u02];

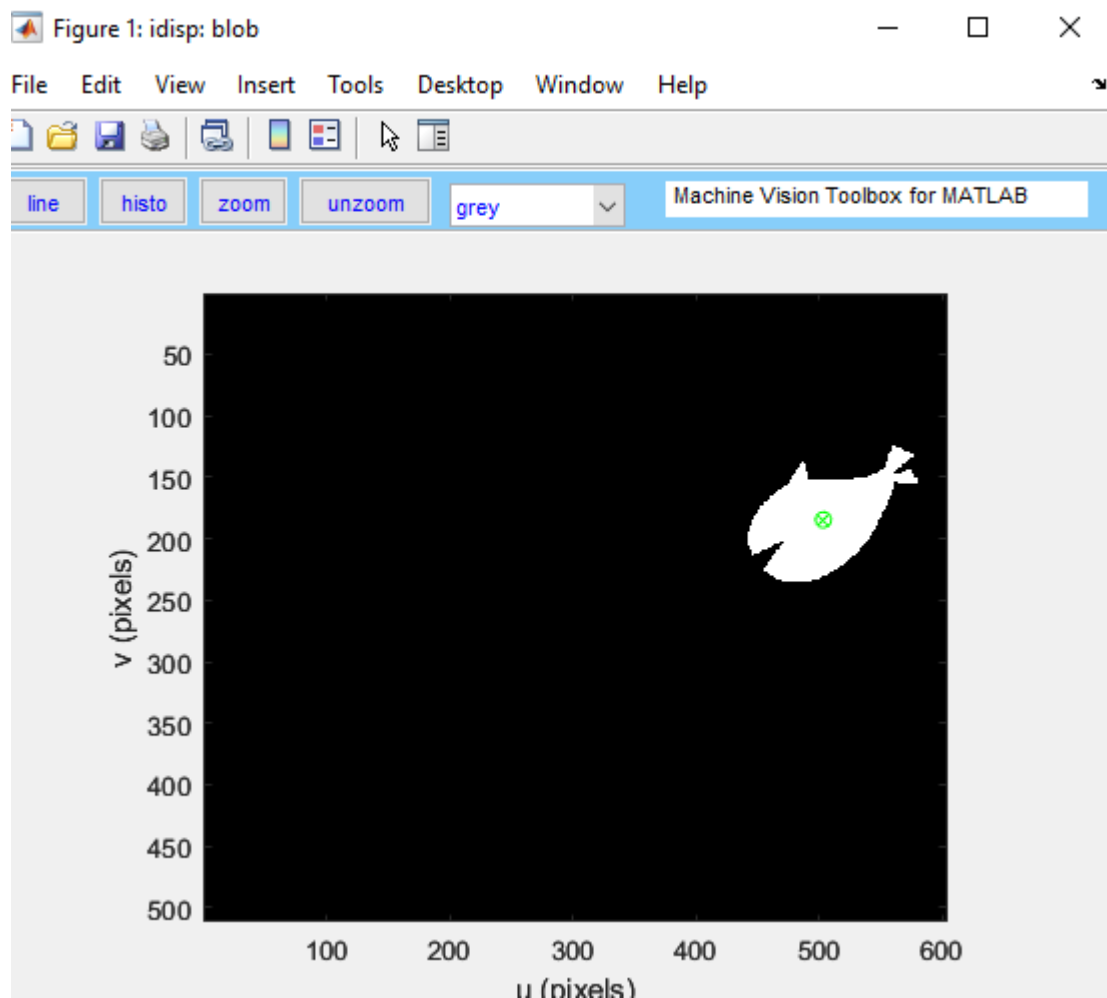
```

```

%plot_ellipse(4*J/m00, [uc,vc], 'y')
lambda=eig(J);
a=2*sqrt(lambda(2)/m00);
b=2*sqrt(lambda(1)/m00);
b/a;
[x,lambda]=eig(J);
x;
v=x(:,end);
atand(v(2)/v(1))

```





Problem 4:

4.1 A corner detector, based on Moravec's algorithm, scans the binary image (below). How many corners does it find? Explain briefly.

It is then necessary to apply a threshold to this score in order to determine which corners should be selected from the list of options. The region is generally considered to be flat when the value of R is small. There is a difference between an edge and a corner when R is less than 0, and a corner when R is greater than 0.

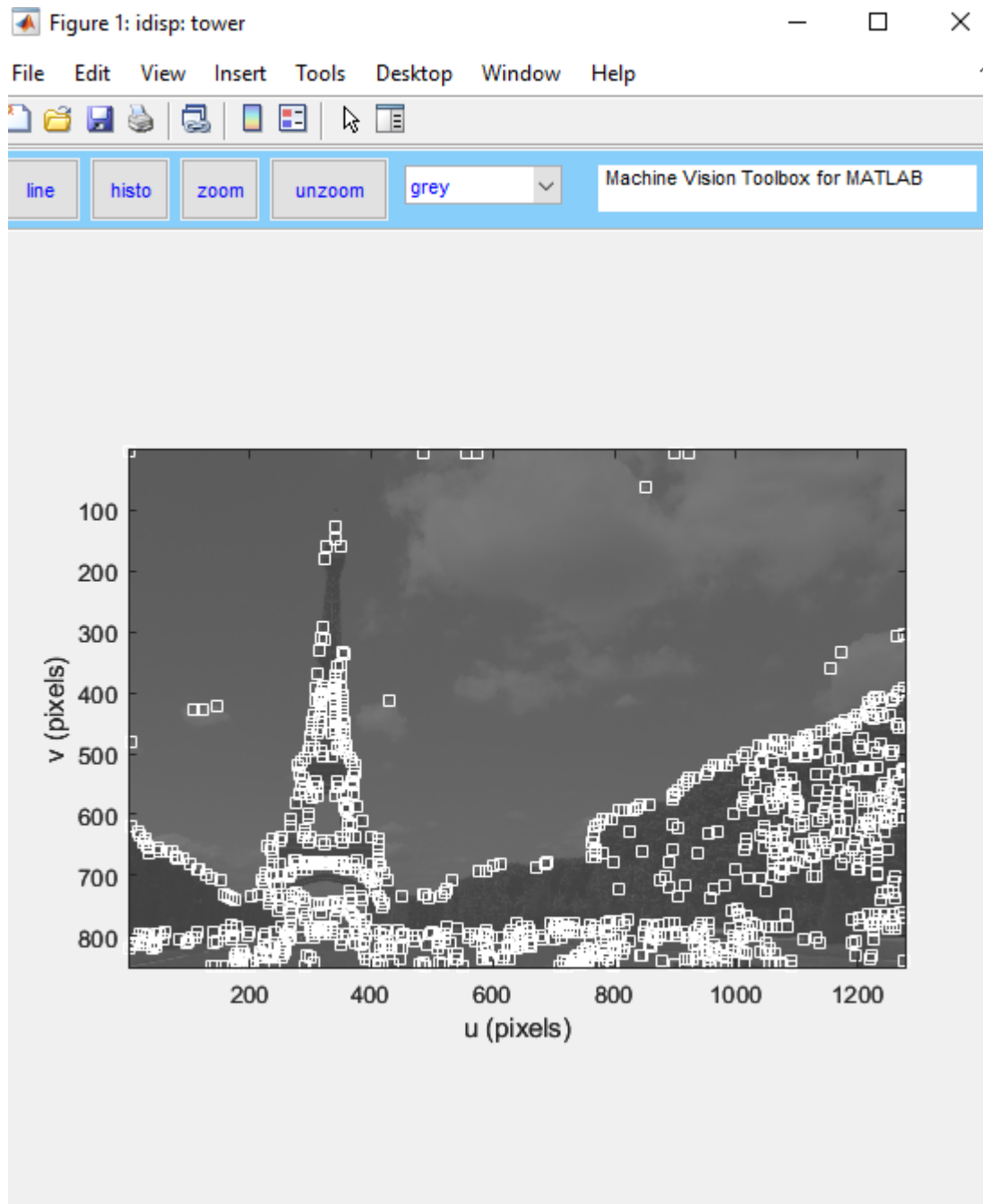
0	1	0	0
1	1	1	0
0	1	0	0
0	0	0	0

4.2 Load and display a grey version of the image titled "eiffel-1.png". Use Harris' corner detector to detect all the corners.

```

tower=imread('eiffel-1.png','grey','double'); %import image
C=icorner(tower,'nfeat',1000);
idisp(tower, 'dark');
C.plot();

```



Problem 5:

5.1 A Sobel edge detector is applied to the binary image below. How many edges are produced?

A Hough Transform is then used to choose the best line. What are the ρ and θ parameters of the best line? Explain briefly.

0	1	0	0
0	1	1	0
0	0	1	0
0	0	1	0

5.2 Load and display a grey version of the image titled “notre-dame.jpg”. Using Hough Transform find straight lines in the image.

Problem 6: Consider the 5 x 5 binary image and the 1x3 structuring image (below).

0	1	1	1	1
1	1	0	0	0
1	1	0	0	0
0	0	0	1	0
0	0	0	1	0

1	1	1
---	---	---

As long as the structuring image is fully contained within the intersection of the tested region, erosion equals pixel equals 1 in this case. A group of pixels containing more than one pixel becomes a group of pixels containing more than one pixel. A dilation of 1 is assigned to any set that overlaps with another set that overlaps with it.

6.1 Perform manually a dilation morphological operation, and briefly explain your calculations.

1	1	0	1	1
1	1	1	0	0
1	1	1	0	0
0	0	1	1	1
0	0	1	1	1

If X represents the Euclidean coordinates of the input binary image, and K represents the coordinates of the structuring element, then let Kx denote the translation of K . It follows that the dilation of X by K is simply the set of all points x such that the intersection of Kx with X is not empty. The mathematical definition of grayscale dilation is identical except for the way in which the set of coordinates associated with the input image is determined.

6.2 Follow up (6.1) by an erosion morphological operation on the result of 6.1, and briefly explain your calculations. How is the sequence of actions called – opening or closing?

1	1	0	1	1
1	1	1	0	0
1	1	1	0	0
0	0	1	1	1
0	0	1	1	1

Closing, the erosion would not do anything because it has no gaps; furthermore, There are two pieces of data that are input into the erosion operator. The first thing that needs to be eroded is the image that needs to be eroded. In the second case, there is a set of coordinate points known as a structuring element (also known as a kernel) which is usually a small set of coordinate points. In fact, it is this structuring element that determines the precise impact of erosion on the input image as a result of the erosion process.

6.3 Load and display the grey version of the image titled “adelson.png”. Let a structuring image be a square of size 150 pixels. Using MATLAB demonstrate the opening of the original image.

Then demonstrate the closing of the original image.

```
%Load Image adelson.png
im = imread("adelson.png", "grey")
%Strucutre Element
se = strel('square', 150);
% Open Sequence
afterOpen = iopen(im,se);
% Display
idisp(afterOpen);
% Closing sequence
beforeClose = iclose(im,se);
idisp(beforeClose)
```

