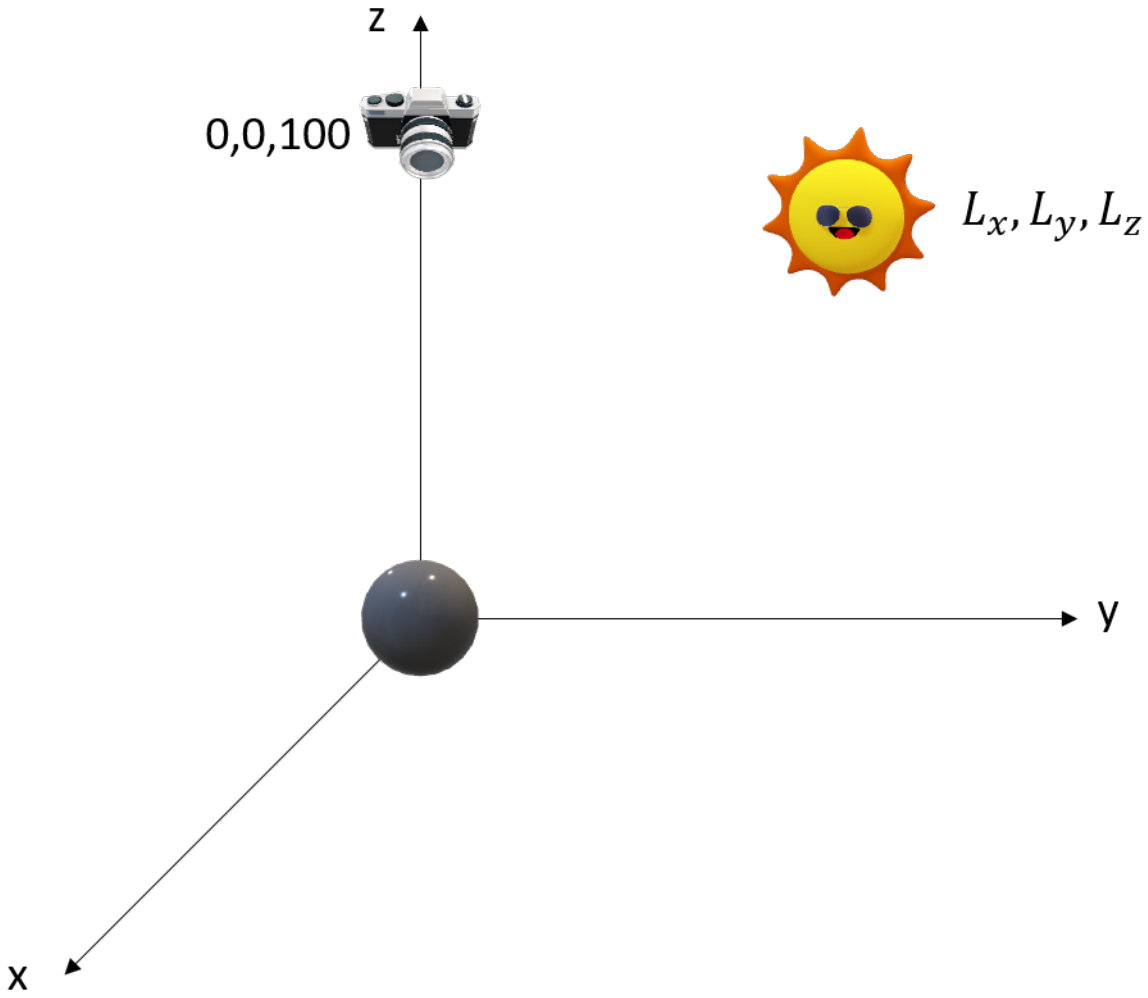
CV - Assignment 1

Eitan Kosman - 312146145
eitan.k@campus.technion.ac.il

December 1, 2019

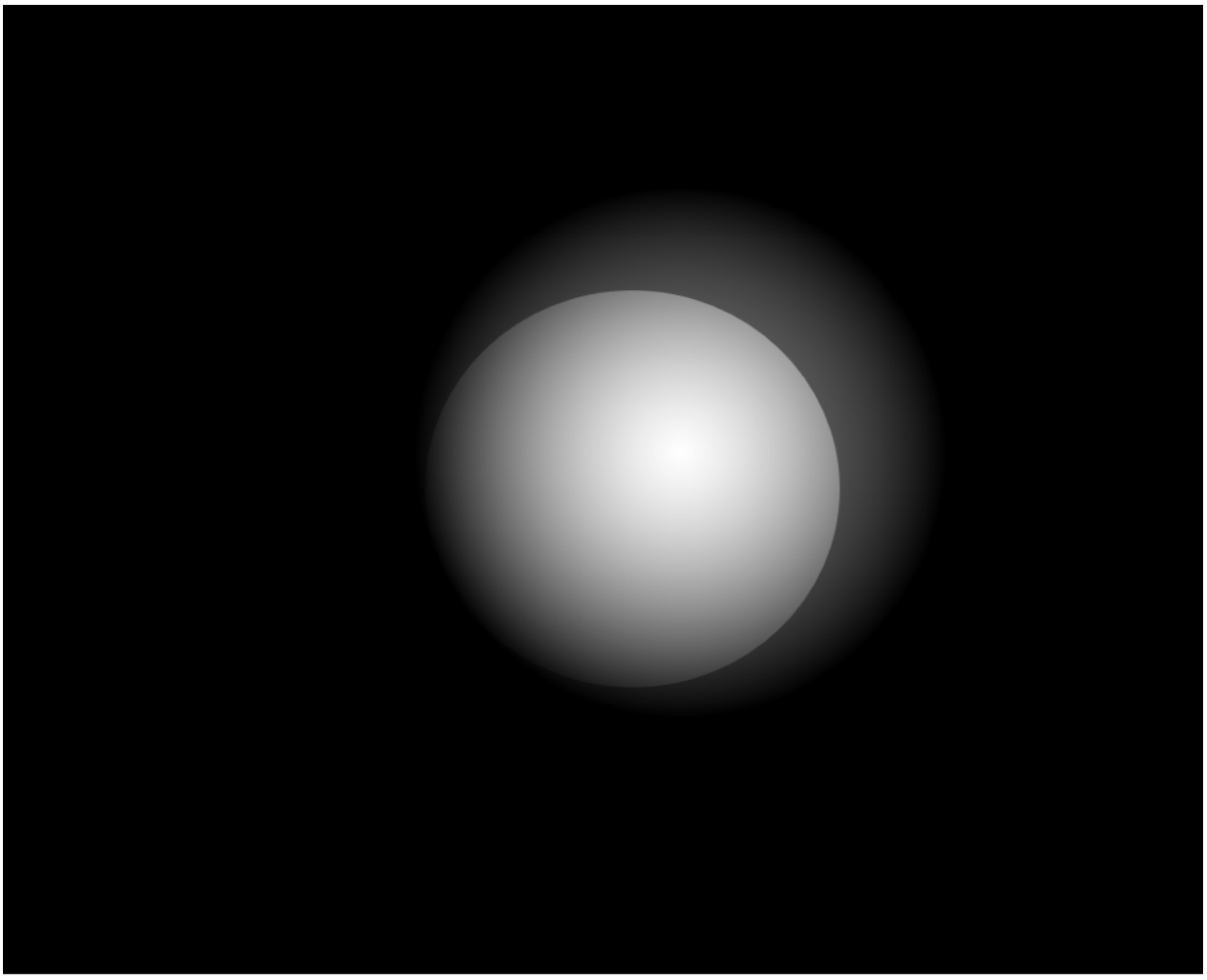
1 Question 1

1.1 A - Diagram of the system



The ball which has a radius of 10 is located in the origin. The camera is located at $(0, 0, 100)$ and points towards the ball. The point light source (sun) is located at $(L_x, L_y, L_z > 0)$

1.2 B - Image taken by the camera



Assuming the light source is above the ball, which is a lambertian ball, the direction of the illumination is approximately uniform on the ball's surface which is visible to the camera. Since the camera and the light source are located above the ball, the ball looks like a circle. Apparently, The illumination of the ball would be stronger where the normal to the ball's surface points directly to the light source. In ?? that should be that the surface in the positive x, y would be more illuminated.

1.3 C - Algorithm for finding the illumination source direction

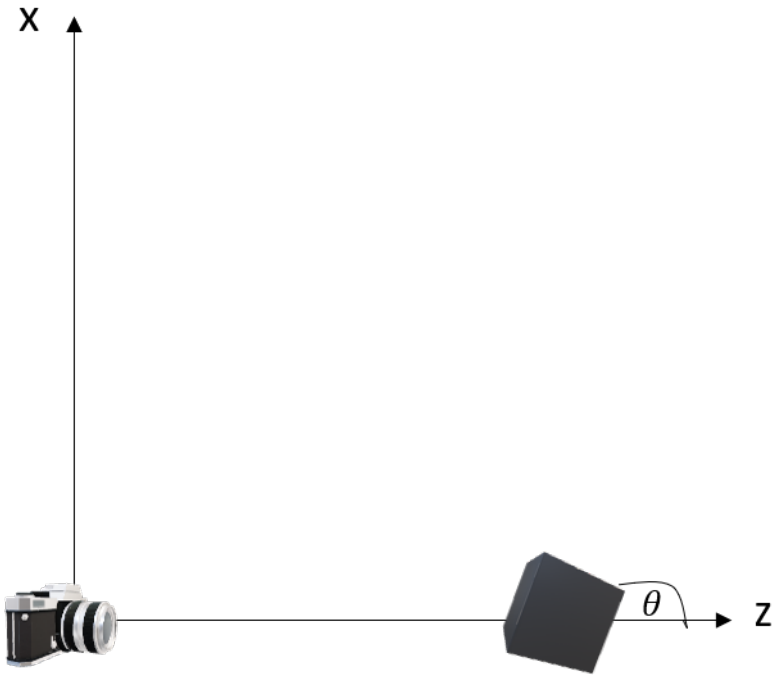
I propose to use the intensity of the illumination to find the direction of the light source. The proposed algorithm is:

Algorithm 1: Estimating light source direction

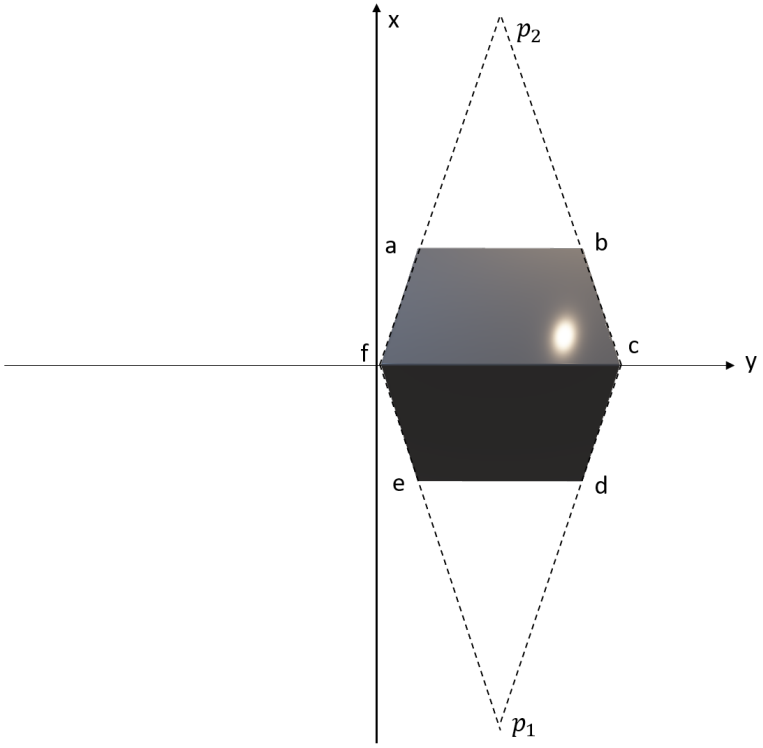
Input: Picture of a ball I and it's diameter D
Output: Direction of the light source
1. Estimate the diameter of the ball in pixel units - Let this be d
2. Calculate the ratio between the true dimensions and the picture dimensions by: $r = d/D$
3. Find the location in the image where the illumination is the highest by: $x, y = argmax_{x',y'} I(x', y')$
4. Calculate the true location of the brightest point by: $x_{true}, y_{true} = x, y/r$
5. Based on these coordinates, find the normal vector n to the ball's surface at x_{true}, y_{true}
6. Return n

2 Question 2

This is the diagram of the system:



When looking from the camera's point of view, we'll see this:



This makes vanishing points at p_1 and p_2. The edges which are parallel to the y-axis remain parallel because the rotation occurs on the y-axis, thus we'll not end up with more vanishing points. Firstly, we define the rotation matrix by: $\begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}$

we calculate the position for each of the 6 points by:

$a'_x = -a_z \cdot sin(\theta) + a_x \cdot cos(\theta) = -9.5 \cdot sin(\theta) + 0.5 \cdot cos(\theta)$

$b'_x = -b_z \cdot sin(\theta) + b_x \cdot cos(\theta) = -9.5 \cdot sin(\theta) + 0.5 \cdot cos(\theta)$
$c'_x = -c_z \cdot sin(\theta) + c_x \cdot cos(\theta) = -9.5 \cdot sin(\theta) - 0.5 \cdot cos(\theta)$
$d'_x = -d_z \cdot sin(\theta) + d_x \cdot cos(\theta) = -10.5 \cdot sin(\theta) - 0.5 \cdot cos(\theta)$
$e'_x = -e_z \cdot sin(\theta) + e_x \cdot cos(\theta) = -10.5 \cdot sin(\theta) - 0.5 \cdot cos(\theta)$
$f'_x = -f_z \cdot sin(\theta) + f_x \cdot cos(\theta) = -9.5 \cdot sin(\theta) - 0.5 \cdot cos(\theta)$

Do the same for z values: $a'_z = a_z \cdot cos(\theta) + a_x \cdot sin(\theta) = 9.5 \cdot cos(\theta) + 0.5 \cdot sin(\theta)$
$b'_z = d_z \cdot cos(\theta) + d_x \cdot sin(\theta) = 9.5 \cdot cos(\theta) + 0.5 \cdot sin(\theta)$
$c'_z = c_z \cdot cos(\theta) + c_x \cdot sin(\theta) = 9.5 \cdot cos(\theta) - 0.5 \cdot sin(\theta)$
$d'_z = d_z \cdot cos(\theta) + d_x \cdot sin(\theta) = 10.5 \cdot cos(\theta) - 0.5 \cdot sin(\theta)$
$e'_z = e_z \cdot cos(\theta) + e_x \cdot sin(\theta) = 10.5 \cdot cos(\theta) - 0.5 \cdot sin(\theta)$
$f'_z = f_z \cdot cos(\theta) + f_x \cdot sin(\theta) = 9.5 \cdot cos(\theta) - 0.5 \cdot sin(\theta)$

The next step is to calculate the 4 edges:
$(a \leftrightarrow f)_x : a'_x - f'_x = -9.5 \cdot sin(\theta) + 0.5 \cdot cos(\theta) - (-9.5 \cdot sin(\theta) - 0.5 \cdot cos(\theta)) = cos(\theta)$ $(a \leftrightarrow f)_z : a'_z - f'_z = 9.5 \cdot cos(\theta) + 0.5 \cdot sin(\theta) - (9.5 \cdot cos(\theta) - 0.5 \cdot sin(\theta)) = sin(\theta)$
$(b \leftrightarrow c)_x : b'_x - c'_x = -9.5 \cdot sin(\theta) + 0.5 \cdot cos(\theta) - (-9.5 \cdot sin(\theta) - 0.5 \cdot cos(\theta)) = cos(\theta)$ $(b \leftrightarrow c)_z : b'_z - c'_z = 9.5 \cdot cos(\theta) + 0.5 \cdot sin(\theta) - (9.5 \cdot cos(\theta) - 0.5 \cdot sin(\theta)) = sin(\theta)$
$(f \leftrightarrow e)_x : f'_x - e'_x = -9.5 \cdot sin(\theta) - 0.5 \cdot cos(\theta) - (-10.5 \cdot sin(\theta) - 0.5 \cdot cos(\theta)) = sin(\theta)$ $(f \leftrightarrow e)_z : f'_z - e'_z = 9.5 \cdot cos(\theta) - 0.5 \cdot sin(\theta) - (10.5 \cdot cos(\theta) - 0.5 \cdot sin(\theta)) = -cos(\theta)$
$(c \leftrightarrow d)_x : c'_x - d'_x = -9.5 \cdot sin(\theta) - 0.5 \cdot cos(\theta) - (-10.5 \cdot sin(\theta) - 0.5 \cdot cos(\theta)) = sin(\theta)$ $(c \leftrightarrow d)_z : c'_z - d'_z = 9.5 \cdot cos(\theta) - 0.5 \cdot sin(\theta) - (10.5 \cdot cos(\theta) - 0.5 \cdot sin(\theta)) = -cos(\theta)$
The 2 vanishing points would then be for:

For the "parallel" edges a-f and b-c:
$(x_1, y_1) = (-f\frac{v_x}{v_z}, -f\frac{v_y}{v_z}) = (-1\frac{cos(\theta)}{sin(\theta)}, -1\frac{0}{sin(\theta)}) = (\frac{-cos(\theta)}{-sin(\theta)}, 0)$

For the "parallel" edges f-e and d-c:
$(x_2, y_2) = (-f\frac{v_x}{v_z}, -f\frac{v_y}{v_z}) = (-1\frac{sin(\theta)}{-cos(\theta)}, -1\frac{0}{-cos(\theta)}) = (\frac{-sin(\theta)}{cos(\theta)}, 0)$

3 Question 3

3.1

True
A line in 3D could be written as:

$$\begin{aligned} x &= m_x \cdot z + b_x \\ y &= m_y \cdot z + b_y \end{aligned} \tag{1}$$

Assuming the point of view is directed to the z-axis (as we usually use this notations in the lectures), every point (x,y,z) will be projected as:

$$\begin{aligned} x' &= -\frac{f}{z} \cdot x = -\frac{f}{z} \cdot (m_x \cdot z + b_x) \\ y' &= -\frac{f}{z} \cdot y = -\frac{f}{z} \cdot (m_y \cdot z + b_y) \\ &\Downarrow \\ x' \cdot zf - m_x \cdot z &= b_x \\ y' \cdot zf - m_y \cdot z &= b_y \end{aligned} \tag{2}$$

By diving the last 2 equations we get:

$$\begin{aligned} \frac{x' \cdot zf - m_x \cdot z}{y' \cdot zf - m_y \cdot z} &= \frac{b_x}{b_y} \\ &\Downarrow \\ \frac{x'f - m_x}{y'f - m_y} &= \frac{b_x}{b_y} \end{aligned} \tag{3}$$

With a few more algebric manipulations we get the following relation between x',y' which indicates a linear relation, thus it's a straight line:

$$x' = \frac{b_x}{b_y} \cdot y' + f(m_x - \frac{b_x \cdot m_y}{b_y}) \tag{4}$$

3.2

True
As we've seen in question **a** straight lines in 3D are projected to straight lines in 2D. We can define a family a parallel lines in 3D by:

$$\begin{aligned} x &= m_x z + b_x \\ y &= m_y z + b_y \end{aligned} \tag{5}$$

Here, m_x and m_y are the same for the whole family but b_x and b_y are different for each line since they define the intercept parameter. Wlog, we assume that the center of the projection is at the origin and the projecting plane is given by $z = f$. The vanishing point for the given family of parallel lines is the intersection of the projecting place with a line l that is parallel to the given family and passes thru the center of projection. Thus, l has the same slope as other lines in the family but its intercept is at the origin, i.e. given by:

$$\begin{aligned} x &= m_x z \\ y &= m_y z \end{aligned} \tag{6}$$

This line intersects the place $z = f$ at

$$\begin{aligned} x &= m_x f \\ y &= m_y f \\ z &= f \end{aligned} \tag{7}$$

The vanishing point of the family of parallel lines is located at $(m_x f, m_y f)$ in the image.

3.3

True
Assuming that the lines are parallel, thus they all have the same slope (m_x, m_y) up to a multiplicative factor. Given from subsection b that the vanishing point is located at $(m_x f, m_y f)$, we can derive that all the points are of the form $\alpha(m_x f, m_y f)$ which implying that all the vanishing points that are parallel to the plane form a straight line.

4 Question 4

4.1

This is the camera matrix I've got: $\begin{bmatrix} 5.713e-3 & 8.2348e-2 & -1.0426e-3 & 8.644e-1 \\ 1e-2 & -1.3219e-4 & 8.2e-2 & -4.891e-1 \\ -8.0398e-6 & 1.5561e-6 & 1.2369e-7 & -5.6149e-4 \end{bmatrix}$

4.2

The re-projection is:

The points clearly appear where they are supposed to be, thus we can be sure that the camera matrix P is approximately good. The error measure I defined is relative to the original points. I assume that as the original vector norm is going bigger, the residual norm has less effect on the error. The formula is given by:

$$E = \sum_i \frac{|x_i estimate - x_i original|}{|x_i original|} = 0.0199 \tag{8}$$

Following that the units are used for plotting the images and points on the screen, the units are in pixels.

4.3

4.3.1

The rq() function decomposes a matrix to 2 matrices - K, R - where K is an upper triangle matrix and R is a unitary matrix. The QR decomposition does the opposite - it decomposes a matrix to 2 matrices - Q, R - where Q is a unitary matrix and R is an upper triangle matrix. The second doesn't fit our needs.

4.3.2

line 2 - retrieves the QR composition of a matrix
line 3 - flips the rows of the transposed R matrix - let this matrix be R
line 4 - flips the columns of R
line 5 - blank
line 6 - transposed Q
line 7 - flips the rows of Q
line 8 - black
line 9 - constructs a diagonal matrix T where each element of the diagonal corresponds to the sign of the diagonal of R
line 10 - computes RT (matrix multiplication)
line 11 - computes TQ (matrix multiplication)

4.4

The solution of the optimization is correct by a multiplicative factor, thus K has to be normalized first.
What I get is: $\begin{bmatrix} 1e-4 & 7.0314e1 & 1.2237e3 \\ 0 & 1.0032e4 & -1.0531e3 \\ 0 & 0 & 1 \end{bmatrix}$

I got approximately the same value for f_x and f_y which is good. However, p_x and p_y have positive and negative values respectively, but I expected it to be the opposite.

4.5

The matrix K makes the orientation of the camera bad. Note that $|K| = -1$ and it makes the camera face in the opposite direction. The problem is that the rq() function forces the diagonal to be positive, thus we can negate the R matrix and this way both x and y axes of the camera align with those of the plane and the camera would face the positive z direction.

4.6

The translation of the camera I've got is: $\begin{bmatrix} -66.4309 \\ 15.2481 \\ 14.0875 \end{bmatrix}$

Here, the positive x direction points left and positive y direction points "inside". This makes the values of the translation reasonable because the camera faces to the right (negative value for the x) and towards to ball. Moreover, the camera elevated which gives a positive value for z.

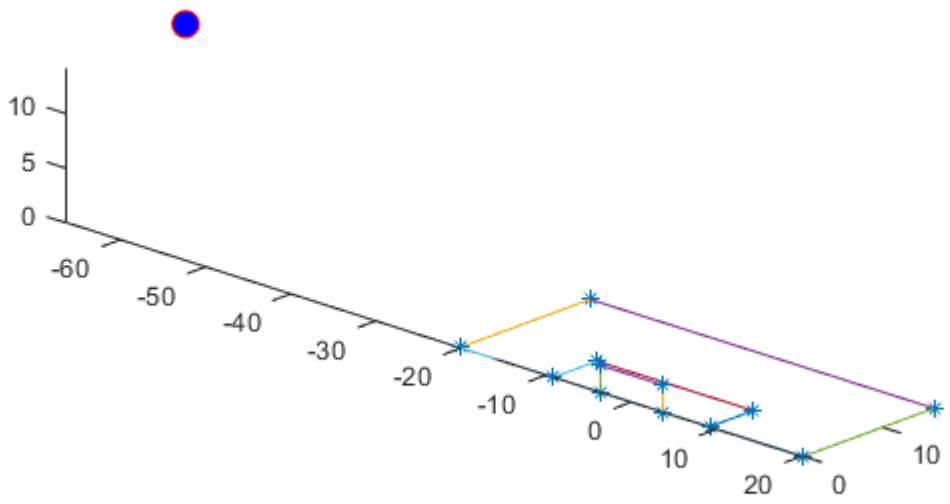


Figure 1: Caption

4.7

The camera position is the blue circle:

4.8

I couldn't determine the true location of the ball because it's actual size is missing. since the ball could be position anywhere on the direction from the camera's point of view. we must know the true dimensions of it to accurately locate it. For example, the smaller the true dimensions of the ball, this means the closer it is to the camera.

5 Question 5

The idea behind my algorithm is to use the images themselves as the kernels after preprocessing. I'll use the closing operation for making the images look smoother (although not really necessary - it works without it too). Later, I'll zeros-pad the images so that the test and train images have the same dimensions. The scoring mechanism will be based on the dot product between the preprocessed test image and train images. In order to have a score in $[0, 1]$ I normalize the images with frobenius-norm.

The algorithm's steps are as follows:

1. Load the test image I_{test}
   (a) Convert to gray-scale
   (b) Binarize and invert (background is zeros, content is ones)
   (c) Use closing without a disc of radius 30
   (d) Normalize using frobenius-norm
2. Load the training set and for every image I_{train}
   (a) Convert to gray-scale

(b) Binarize and invert (background is zeros, content is ones)

(c) Normalize using frobenius-norm

(d) Compare dimensions with the test image and zero-pad the smaller image if necessary

(e) Calculate $I_{test} \cdot I_{train}$ and print the value

The scores I get are:

Figure 2: 0.805089313960975

Figure 3: 0.785228023189117

Figure 4: 0.932678984485426

Figure 5: 0.775641291212660

Figure 6: 0.502276541639510

This algorithm clearly emphasize the similarity between the test image and leaf3 which looks very similar to the test image and it has the highest score. **Note:** I could omit the closing operation and still get the best score for leaf3, but this way I get a higher score (by 0.02) so I preferred keeping it for probably making the algorithm more robust.