THE UNIVERSITY *of* EDINBURGH

Robotics: Science and Systems

Course Assignment 1

Henrique Ferrolho - s1683857
henrique.ferrolho@gmail.com

November 4, 2016

1 Forward and inverse kinematics

1.1 (10 marks)

$$q = \begin{pmatrix} 0.66 \\ -0.44 \\ 1.06 \\ 1.33 \end{pmatrix}$$

$$T_{base \to L_1} = \begin{bmatrix} cos(q_1) & -sin(q_1) & 0 & 0 \\ sin(q_1) & cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0.16 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{L_1 \to L_2} = \begin{bmatrix} cos(q_2) & 0 & sin(q_2) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(q_2) & 0 & cos(q_2) & 0.15 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{L_2 \to L_3} = \begin{bmatrix} cos(q_3) & 0 & sin(q_3) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(q_3) & 0 & cos(q_3) & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{L_3 \to L_4} = \begin{bmatrix} cos(q_4) & 0 & sin(q_4) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(q_4) & 0 & cos(q_4) & 0.15 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{L_4 \to gripper} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.35 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{base \to gripper} = T_{base \to L_1} \cdot T_{L_1 \to L_2} \cdot T_{L_2 \to L_3} \cdot T_{L_3 \to L_4} \cdot T_{L_4 \to gripper}$$

$$= \begin{bmatrix} -0.29243998 & -0.61311685 & 0.73387096 & 0.25840905 \\ -0.22696411 & 0.78999223 & 0.56956086 & 0.20055253 \\ -0.92895972 & 0 & -0.37018083 & 0.48346881 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And therefore:

$$\boldsymbol{y_t} = \begin{bmatrix} 0.25840905 \\ 0.20055253 \\ 0.48346881 \end{bmatrix}$$

1.2   **(10 marks)**

1.3   **(10 marks)**

2 Signal filtering & State Estimation

2.1 (10 marks)

2.2 (10 marks)

2.3 (20 marks)

3 Computer Vision

3.1 Camera Geometry (10 marks)

$f = 10\,\text{mm}$

$D = 10\,\text{m} = 10\,000\,\text{mm}$

$sensor : 10\,\text{mm} \times 10\,\text{mm} = 1000 \times 1000\ \text{pixels}$

$h = 100\ \text{pixels} = 1\,\text{mm}$

$$\frac{1}{D} + \frac{1}{d} = \frac{1}{f} \qquad\qquad \frac{d}{D} = \frac{h}{H}$$

$$\frac{1}{d} = \frac{1}{f} - \frac{1}{D} \qquad\qquad H = \frac{hD}{d}$$

$$\frac{1}{d} = \frac{1}{10} - \frac{1}{10000} \qquad\qquad H = \frac{1 \times 10000}{\dfrac{10000}{999}}$$

$$\frac{1}{d} = \frac{1000}{10000} - \frac{1}{10000} \qquad\qquad H = 999\,\text{mm} = 0.999\,\text{m}$$

$$\frac{1}{d} = \frac{999}{10000}$$

$$d = \frac{10000}{999}$$

The observed statue is $0.999\,\text{m}$ tall.
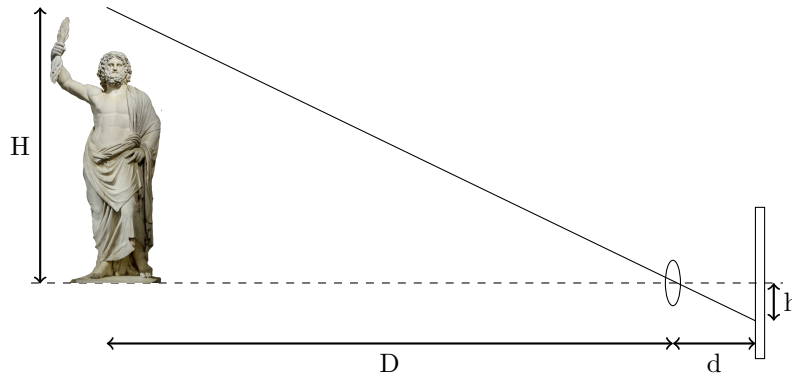
The distance D to the statue needs to be known in order to calculate the distance d between the lens and the film/sensor, using the focal length f. Furthermore, given D, d, and h, it is easy to calculate the height of the statue H.

3.2 Image Processing (10 marks)

The following row of images/samples is a visual representation of the *intermediate* keypoints of the car counter program.

Firstly the program fetches the frames from the live camera feed - the first image from the row of images above is an example of a frame fetched from the camera. Afterwards it does a *image segmentation* by applying a *background removal* technique - second image, counting from the left, in the images row above.
At this point it is useful to turn the image into a binary image - middle image sample.
Then two *morphological transformations* are applied to remove the noise left in the frame - also known as *morphological closing* and *opening* - penultimate image sample.
Finally the last image from the samples row above shows the *region of interest* - region in blue, activated by the passing car (red segment of the detection region).

The general idea is to count every time a white spot is detected in the blue line - actually, to introduce some tolerance, it should not be a line with a single pixel thickness, rather a 5 pixels thick line, for example.

Furthermore, to save computation time, and since we only care for what is going on inside the blue region, we can *crop* everything else besides that region right after the frame capture. Nonetheless, the entire frame of each of the samples above is used (no cropping) for better understanding of the main idea.

Algorithm 1 Car counter program

1: **procedure** CARCOUNTER
2:    $carPassing \leftarrow$ False
3:    $carCounter \leftarrow 0$
4:    **loop**
5:       $frame \leftarrow cam.fetchFrame()$
6:       $CropRegionOfInterest(frame)$ ▷ Crop to save computation time
7:       $SubtractBg(frame)$
8:       $BinaryThreshold(frame)$
9:       $MorphOpen(frame)$
10:      $MorphClose(frame)$
11:      **if** $carFound(frame)$ **then**
12:         **if** not $carPassing$ **then**
13:            $carPassing \leftarrow$ True
14:            $counter \leftarrow counter + 1$
15:         **end if**
16:      **else if** carPassing **then**
17:         $carPassing \leftarrow$ False
18:      **end if**
19:    **end loop**
20: **end procedure**

21: **function** SUBTRACTBG($frame$)
22:    $hourOfDay \leftarrow getTime().hour$
23:    $bg \leftarrow DB.image("backgrounds/" + hourOfDay + ".png")$
24:    $frame.subtractImg(bg)$
25: **end function**

26: **function** MORPHOPEN($frame$)
27:    erode(frame)
28:    dillate(frame)
29: **end function**

30: **function** MORPHCLOSE($frame$)
31:    dillate(frame)
32:    erode(frame)
33: **end function**

34: **function** CARFOUND($frame$)
35:    **for** $i \leftarrow 0, frame.size$ **do**
36:       **for** $j \leftarrow 0, frame[i].size$ **do**
37:         **if** $frame[i][j] = RGB(255, 255, 255)$ **then**
38:            **return** True
39:         **end if**
40:       **end for**
41:    **end for**
42:    **return** False
43: **end function**

3.3 Shape Recognition (10 marks)

The first thing to do in order to narrow the search results by 50% is to check if the card has *red* or *black* symbols - a simple *RGB normalization* can be used. After this step the search results are narrowed either to *spades and clubs*, or *hearts and diamonds*.

A Bayes classifier needs to be trained using *4 classes*, one for each suit.