Robotics: Science and Systems

Course Assignment 1

Henrique Ferrolho - s1683857
henrique.ferrolho@gmail.com

November 3, 2016

1 Forward and inverse kinematics

1.1 (10 marks)

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)

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

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

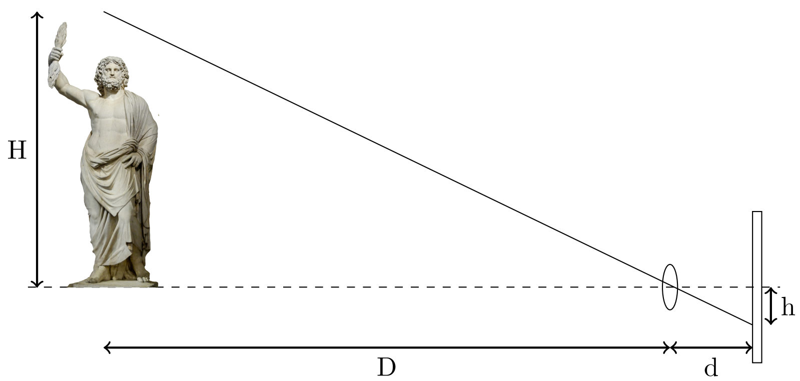
$sensor : 10\,\text{mm} \times 10\,\text{mm} = 1000 \times 1000$ pixels

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

$$\begin{aligned}
\frac{1}{D} + \frac{1}{d} &= \frac{1}{f} \\
\frac{1}{d} &= \frac{1}{f} - \frac{1}{D} \\
\frac{1}{d} &= \frac{1}{10} - \frac{1}{10000} \\
\frac{1}{d} &= \frac{1000}{10000} - \frac{1}{10000} \\
\frac{1}{d} &= \frac{999}{10000} \\
d &= \frac{10000}{999}
\end{aligned}$$

$$\begin{aligned}
\frac{d}{D} &= \frac{h}{H} \\
H &= \frac{hD}{d} \\
H &= \frac{1 \times 10000}{\frac{10000}{999}} \\
H &= 999\,\text{mm} = 0.999\,\text{m}
\end{aligned}$$

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 should fetch 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 should do 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, I used the entire frame in the above row of samples (no cropping) for better understanding of the main idea.

Algorithm 1 Car counter program

procedure CarCounter
    $carPassing \leftarrow$ `False`
    $carCounter \leftarrow 0$
    loop
        $frame \leftarrow cam.fetchFrame()$
        $CropRegionOfInterest(frame)$ ▷ Crop to save computation time
        $SubtractBg(frame)$
        $BinaryThreshold(frame)$
        $MorphOpen(frame)$
        $MorphClose(frame)$
        if $carFound(frame)$ **then**
            if `not` $carPassing$ **then**
                $carPassing \leftarrow$ `True`
                $counter \leftarrow counter + 1$
            end if

Algorithm 2 Continuation

```
        else if carPassing then
            carPassing ← False
        end if
    end loop
end procedure

function SubtractBg(frame)
    hourOfDay ← getTime().hour
    bg ← DB.image("backgrounds/" + hourOfDay + ".png")
    frame.subtractImg(bg)
end function

function MorphOpen(frame)
    erode(frame)
    dillate(frame)
end function

function MorphClose(frame)
    dillate(frame)
    erode(frame)
end function

function CarFound(frame)
    for i ← 0, frame.size do
        for j ← 0, frame[i].size do
            if frame[i][j] = RGB(255, 255, 255) then
                return True
            end if
        end for
    end for
    return False
end function
```

3.3 Shape Recognition (10 marks)