School of Computing
National University of Singapore
Biometrics Course
July 2016

Assignment #4

Part I

Q1. Suppose that in answering a question in a multiple choice test, an examinee either knows the answer, with probability p, or he/she guesses with probability $1 - p$. Assume that the probability of answering a question correctly is 1 for an examinee who knows the answer and $1/m$ for the examinee who guesses, where m is the number of multiple choice alternatives. What is the probability that an examinee actually knew the answer to a question, given that he has correctly answered it?

Q2. A prize is hidden behind one of three doors A, B, and C. The contestant picks a door, say A, but it is left closed. The host (who knows the actual location of the prize, but will never reveal the prize at this stage of the game) opens door C and shows that there is no prize behind it. Should the contestant change his mind and select door B? Formulate this problem as Bayes inference. What is the prior probability of the prize being behind each door (i.e., A, B, or C) before door C was opened? What is the posterior probability after door C has been opened? What should be the contestants decision?

Q3. Your spaceship has landed on an unknown planet. On this planet, it is possible with prior probability P_1, that the sun rises every day, but it is also possible, with prior probability $1 - P_1$, that the sun only rises 50% of the time. The sun has risen for the past three days. Your commander asks you to determine, with minimum probability of error, whether or not this is a planet on which the sun always rises. For what values of P_1 would you answer yes?

Part II

Geometric Transform

This task helps you understand the geometric transformation. Given a 2D point P at (P_x, P_y), you are now going to rotate it around the centre point C at (C_x, C_y). Follow the instruction:

1. Transform the original coordinates system to the one originated at C by $T_1 = \begin{bmatrix} 1 & 0 & -C_x \\ 0 & 1 & -C_y \\ 0 & 0 & 1 \end{bmatrix}$

2. Rotate the points by θ counter-clockwise by $R = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

3. Transform the altered coordinates system to original coordinates by $T_2 = \begin{bmatrix} 1 & 0 & C_x \\ 0 & 1 & C_y \\ 0 & 0 & 1 \end{bmatrix}$

4. Thus the transformation result for P would be computed by $\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = T_2 R T_1 \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$. The coordinates for P' is $(\frac{x'}{z'}, \frac{y'}{z'})$. Pay attention to the order of T_1, R and T_2.

Let coordinates for P, C be $(3, 3)$, $(2, 2)$, and $\theta = \frac{i}{4}\pi, i = 1, 2, \ldots, 7$. Use *matlibplot* to draw P and all its transformed results in a same figure. Save the figure and submit it.

Background Recovery

In this task, you will deal with videos with OpenCV. The attached trafic.mp4 is a video taken by stationary cameras. You will be instructed to recover background by two methods.

Method 1: Averaging

- Read the mp4 video.

```
cap= cv2.VideoCapture('traffic.mp4')
```

- Print the frame width, frame height, frames per second and frame count of the input video.

```
cap.get(cv2.CAP_PROP_FRAME_WIDTH)
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
cap.get(cv2.CAP_PROP_FRAME_FPS)
cap.get(cv2.CAP_PROP_FRAME_COUNT)
```

- Convert frame width, frame height, frames per second and frame count into integers using int().

- Get the background object by averaging away the foreground (i.e. moving) objects using the following suggested codes:

```
_, img = cap.get()
avgImg = np.float32(img)
for fr in range(1, frameCount):
    _,img = cap.read()
    # write your code here

cap.release()
```

Capture the background and save the background image.

Method 2: Color distribution

Read **Page5-13** in **removebg.pdf** and implement it. In this method, you will perform clustering for each pixel over all the frames of the video. For more detail, you may follow these two steps for each pixel(i,j) in the image:

1. Collect the BGR values of pixel (i, j) for all frames.
2. Use k-means method to perform clustering on these BGR values. Use *cv2.kmeans()* with 2 cluster centers. The center of larger cluster is then the desired BGR value for pixel (i,j) of background image.

Save the background image.

Questions

1. Compare the two methods. Which method gives a better result? Why?

2. How would you use SVD to recover the background?

Submission

Submit your answers for Part I. Submit your Python code, result images and answers for questions for Part II by the name of XXX_assignment4.zip.

Deadline: 22 July 2:00pm