EE371Q Digital Image Processing

Homework #4

Please note that instead of submitting Homework 4, you have an alternative option to receive 100% on this assignment by **participating in an interesting video quality assessment human study!**

Description of the Human Study

This study is trying to understand the perceived quality of 4K videos and is open to anyone with normal or corrected-to-normal vision. Subjects will watch short videos of 8s duration on a 4K TV and provide a rating between 0-100 based on the perceived video quality.

The study will start next week between Nov 13-15, 2023. It is divided into two separate sessions of approximately 45 minutes each. Sign up for the study at the following Microsoft Booking page: https://outlook.office365.com/owa/calendar/LIVE@utexas.onmicrosoft.com/bookings/

You need to sign up for **two 1 hour** sessions. These two sessions need to be at least 24 hours apart. Please follow these steps:

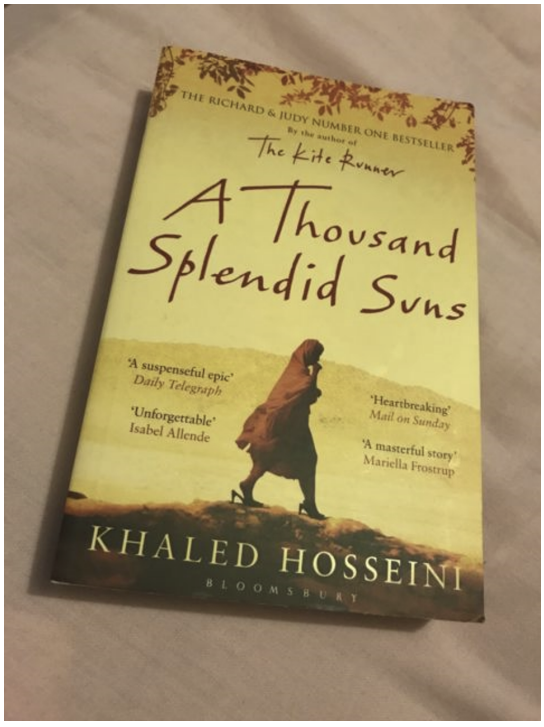
1. Pick a date and time on the booking page for your **first session**.
2. Add name, email address and confirm.
3. Select “new booking” on the confirmation page.
4. Pick a date and time for your **second session**.
5. Add name, email address and confirm.

You will receive a confirmation email after signing up as well as reminder emails before the sessions.

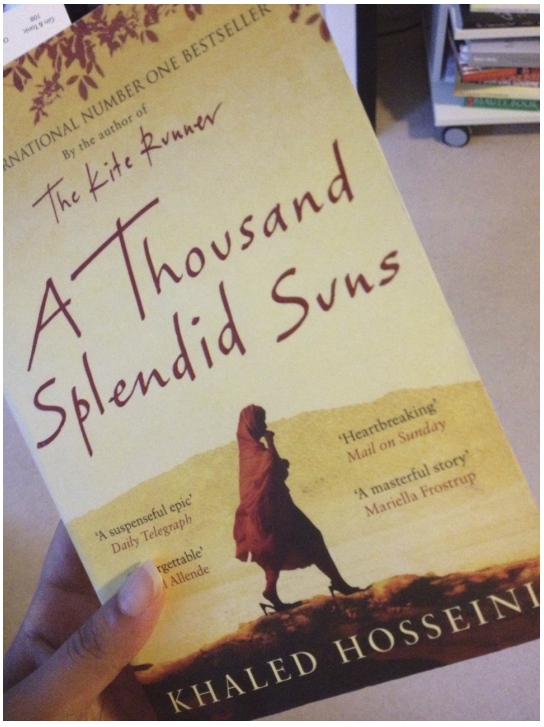
Contact Asvin (asvin@utexas.edu) or Seobin (seobinpark@utexas.edu) if you have any doubts or questions.

- This homework is due two weeks from now, on **November 29th at 11:59pm**.
- You are encouraged to work with other students to understand the course material. However, sharing source code, images, or other answers from any homework is strictly forbidden. You are to submit your own work.
- Please include your written answer and/or output images for each problem.
- The instructions are written with Matlab and Python in mind, but feel free to use any programming language for this assignment.
- Please show all steps for full credit and attach the code to each problem.
- **All the images needed for this homework can be downloaded as .jpg files from Canvas under the HW4 folder in the Files tab.** The homework should be formatted as a PDF file with the output images.

Problem 1 - Scale-Invariant Feature Transform (SIFT) and Object Matching (30 points)



book1.jpg



book2.jpg

a) (5 points) Read the image "book2.jpg" and convert it to grayscale. Extract SIFT keypoints and descriptors using the SIFT_create() function. Visualize a random selection

of 50 SIFT features using the drawKeypoints function. Display the grayscale image and the detected SIFT keypoints (features) side by side.

b) (5 points) Extract SIFT descriptors from the detected keypoints for both "book2.jpg" and "book1.jpg." Store these descriptors in separate objects, one for each image.

c) (5 points) Extract keypoints and descriptors for "book1.jpg" using the same SIFT feature extraction process. Store these keypoints in an object named "keypoints2" and descriptors in an object named "descriptors2."

d) (10 points) Use SIFT matching methods with the BFMatcher (Brute-Force Matcher) to match descriptors between "book1.jpg" and "book2.jpg." Display the matching of SIFT descriptors in a visually interpretable way.

e) (5 points) Observations and Advantages of SIFT: Comment on what you have observed during the SIFT matching process. Explain how the extracted SIFT features are useful for object matching and why they are advantageous for local feature extraction in computer vision applications.

Problem 2 - Huffman Binary Tree (30 points)

Huffman coding is a lossless image compression technique. In this problem, we will understand the Huffman coding algorithm better, but will not be required to implement it. PS; you will not be required to generate the compressed image from the original image.

mario.jpg

a) (5 points) Read the image mario.jpg, convert the image to grayscale with each pixel ranging between 0 and 15 (only 4 bits to represent each pixel), and display the grayscale image and its histogram side by side.

b) (10 points) Write a python/matlab function to generate the Huffman tree. The function should take the input parameter as an array of size 16 which contains the corresponding number of occurrences for each pixel value. The function should then calculate the probabilities for each pixel value, and then generate the Huffman tree as described in the module slides. Display the Huffman tree for mario.jpg with proper formatting by generating the input array of size 16 for the image and passing through the function.

c) (10 points) Write a python/matlab function which extends/uses the previous function, and additionally calculates the codeword for each pixel value from 0 to 15. The function should take the input parameter as an array of size 16 with the corresponding number of occurrences for each pixel value. Display the codeword generated for each pixel value in mario.jpg by generating the input array of size 16 for the image and passing through the function.

d) (5 points) Write a python/matlab function which extends/uses the previous function, and additionally calculates the new BPP and the compression ratio after applying Huffman coding. The function should take the input parameter as an array of size 16 with the corresponding number of occurrences for each pixel value, and the output of the function should be the new BPP and the compression ratio. Report the BPP and compression ratio achieved for mario.jpg by generating the input array of size 16 for the image and passing through the function.

Problem 3 - Hough Transform and Line Detection (30 points)

In this problem, you will work with the Canny edge detector and implement a Hough Transform-based line detector to identify lines in a given image

lanes.jpg

a) (10 points) Read in the image lanes.jpg and convert it to grayscale. Denoise the grayscale image using a Gaussian filter with a sigma value of 1 and a window size of 7. Compute the binary edge map using a Canny edge detector (there are many libraries to do this, e.g. cv2.Canny()). Show the original image, denoised image, and binary edge map.
b) (15 points) Implement a function that detects lines using the Hough Transform. The inputs should be the image edge map and a threshold which specifies the number of votes required by the accumulator to be considered a line. The output should be an image that displays the computed lines.
c) (5 points) Apply your function to lanes.jpg. Play around with the threshold and parameters of the edge detector to get something that resembles the lines in the image. Display the edge map and the detected lines image.

Problem 4 - Object Detection using Machine Learning (10 points)

You will experiment with object detection using a machine learning model of your choice. There are many tutorials available online on using pre-trained models for object detection. Feel free to follow any of these.

a) (5 points) Do some research on pretrained models that can be used for object detection. Find one that looks interesting to you and load it into your development environment.
b) (5 points) Find and load an image that contains multiple objects and display it. Then, pass the image through your model and display it with bounding boxes and labels corresponding to the objects identified by the model.