Koç University İş Bank Artificial Intelligence Research Center
Koç University Graduate School of Sciences and Engineering
Fall 2021 Admissions

KUIS AI Fellowship Exam

Assigned: June 28, 2021 10:00 AM & **Due:** June. 30, 2021 10:00 AM

- The exam contains two parts (A and B) to assess your analytical and computing skills, respectively.
- **Please respond to the problems based on your understanding.**
- In part A, you have two problems to work on.
  Please submit your work to each in any format, either scanned or typed.
- In part B, you have one computing problem to work on and to submit.
  You can write your program in any language you want
  e.g., C, C++, Python, Java or MATLAB/Octave, etc.
  We should be able to run your results without any issue.
  If there's a build or install process please include that as a readme file.
  When possible, please avoid using ML/NLP libraries and try to write the code from scratch.
  Your code will be evaluated based on the following criteria:
  correctness, readability, efficiency, scalability, and elegance.
- Submit solutions separately in the corresponding submission page.
- This is an open-book exam but copying your answers from another source is not allowed.
- Any type of academic dishonesty will not be tolerated.
  Your submission has to be your individual work.
  Any sharing of ideas, codes, material (even in an abstract level) is not allowed.
  Please upload your singed honor code statement (see below) at the end of the exam.
- The exam material is a property of KUIS AI Center, and for individual use.
  Sharing of the exam material is strictly prohibited.
  Partial / complete use of this material without permission
  will be considered as a violation of copyright infringement.

Name: _____

Honor Code Certification : [Please submit a handwritten and signed copy.]
I hereby certify that I have completed this exam on my own without any help from anyone else. I understand that the only sources of authorized information in this exam are textbooks and reference material on public domains available to all other attendees. As such, I confirm that I have not used, accessed or received any information from any other unauthorized source in taking this exam. The effort in the exam thus belongs completely to me.

A Analytical

A.1 Linear Algebra

The following nonlinear equations are defined between $\mathbf{X} = [x_1,x_2]$ and $\mathbf{Y} = [y_1,y_2]$

$$y_1 = a_1 \cos(x_1) + a_2 cos(x_1 + x_2)$$
$$y_2 = a_1 \sin(x_1) + a_2 sin(x_1 + x_2)$$

where a_1 and a_2 are constant coefficients.

(a) Calculate the small changes in $\mathbf{Y}$ for the small changes in $\mathbf{X}$

(b) How do you calculate the small changes in $\mathbf{X}$ for the small changes in $\mathbf{Y}$? (no need to carry out the calculations)

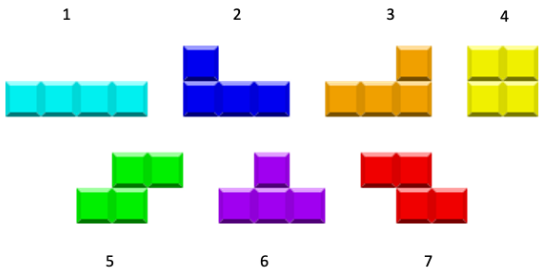
(c) At what values of $\mathbf{X} = [x_1,x_2]$ the calculation asked in item (b) is not possible?

(d) Calculate the closed form solution for x_2 (Hint: take the square of the above equations and then sum them up)

(e) Calculate the closed form solution for x_1 (Hint: write down the above equations in terms of x_1 assuming x_2 is calculated in item (d) and then apply change of variables)

A.2 Probability

In computer game Tetris, there are 7 distinct objects each consisting of 4 squares as shown in Figure below:



- Here, each object is assigned an index between 1 and 7.
- Let $\{x_n, n \in \mathbb{Z}_+\}$ represent the sequence of indexes of the objects used in the game. For example, x_1 is the index of the first object generated, x_2 is the index of the second object generated, and so on...

Now, we consider two randomization methods for the generation of this index sequence:

Method I. Each x_n is independently generated from a uniform distribution over $\mathcal{I}_7 = \{1, 2, 3, 4, 5, 6, 7\}$, i.e., each index have the same probability.

Method II. The indexes are generated in groups of size 7. For example, at time n=1, we obtain a uniformly random shuffle of $\mathcal{I}_7 = \{1, 2, 3, 4, 5, 6, 7\}$, and use this resulting sequence from $n = 1$ to $n = 7$. Then, we generate another independently shuffled version of $\mathcal{I}_7$ at time $n = 8$, and use this sequence from $n = 8$ to $n = 14$. We continue with this procedure for the rest.

Please answer the following questions separately for these two randomization methods:

a. Write down the probability mass function $p_{x_n}(k) = P(x_n = k)$ for each randomization method.

b. Find the mean value of x_n, i.e., $E(x_n)$ for each randomization method.

c. Find the correlation function, $r(n, m) = E(x_n x_m)$ for each randomization method.

d. Let $\{y_n\}$ represent a random sequence which is the minimum time separation between x_n and the next time the value in x_n is repeated:

$$y_n = \min\{d \mid x_{n+d} = x_n,\ d \in \{1, 2, \ldots, \}\}.$$

Find the probability mass function for y_n for each randomization method.

B Programming

In this programming assignment, you will work on the registration of two sets of 3D data points (e.g., given in Q_data.txt and P_data.txt). Registration of 3D data sets is a frequently encountered problem in computer graphics, computer vision, and robotics. Mathematically speaking, registration refers to finding the best 3D transformation (rotation and translation) between two corresponding data sets based on minimum distance error.

Hence, given two sets of corresponding 3D points $Q = \{q_1, q_2, ..., q_N\}$ and $P = \{p_1, p_2, ..., p_N\}$, we seek for a rotation matrix $\mathbf{R}_{3\times3}$ and a translation vector $\mathbf{t}_{3\times1}$, that minimize the distance error E between the sets Q and P:

$$E = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{q}_i - \mathbf{R}\mathbf{p}_i - \mathbf{t}\|^2$$

where $\mathbf{p}_i$ and $\mathbf{q}_i$ are column vectors of size 3×1, respectively representing the data points p_i and q_i in terms of x, y and z coordinates. In the above equation, $||\cdot||$ denotes the $l2$ norm of a column vector, and for each q_i, p_i is the point that corresponds to q_i based on minimum distance; i.e., p_i is the closest point to q_i.

In order to calculate the optimal rotation matrix $\mathbf{R}_{3\times3}$ and translation vector $\mathbf{t}_{3\times1}$, you first need to subtract the center of mass (mean) of each data set from each data point in that set:

$$\mathbf{q}'_i = \mathbf{q}_i - \boldsymbol{\mu}_Q$$
$$\mathbf{p}'_i = \mathbf{p}_i - \boldsymbol{\mu}_P$$

where

$$\boldsymbol{\mu}_Q = \frac{1}{N}\sum_{i=1}^{N} \mathbf{q}_i$$

$$\boldsymbol{\mu}_P = \frac{1}{N}\sum_{i=1}^{N} \mathbf{p}_i$$

Let $\mathbf{Q}_{3\times N}$ and $\mathbf{P}_{3\times N}$ be the two data matrices, each having the mean-subtracted data points as its columns: $\mathbf{Q} = [\mathbf{q}'_1\mathbf{q}'_2...\mathbf{q}'_N]$ and $\mathbf{P} = [\mathbf{p}'_1\mathbf{p}'_2...\mathbf{p}'_N]$. Then, you calculate the cross correlation matrix $\mathbf{W}_{3\times3}$ as

$$\mathbf{W} = \mathbf{Q}\mathbf{P}^T$$

If SVD (Singular Value Decomposition) is applied to the cross correlation matrix so that $\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, then the optimal rotation matrix and translation vector can be calculated as

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T$$

$$\mathbf{t} = \boldsymbol{\mu}_Q - \mathbf{R}\boldsymbol{\mu}_P$$

However, it is not usually possible to determine the optimal relative rotation and translation in one single step since the correct correspondences are typically not known in advance. Hence, an *iterative* solution is required. For this purpose, point-to-point correspondences (i.e. the pairs of closest points) between two data sets are identified at each iteration and one of the data sets, which is P in our case, is incrementally rotated and translated so as to bring it close to the other data set Q until the percent relative change in distance error is less than some threshold value.

The pseudocode of this algorithm can be given as:

repeat

identify point-to-point correspondences between Q and P;
calculate the optimal rotation matrix $\mathbf{R}_k$ and translation vector $\mathbf{t}_k$;
update (transform) P by applying the calculated $\mathbf{R}_k$ and $\mathbf{t}_k$;
compute the distance error E;
compute the percent relative change in the distance error,

$$RE(\%) = 100 * (E_{k-1} - E_k)/E_{k-1} \ ;$$

until $RE < RE_{threshold}$

Note above that $\mathbf{R}_k$ and $\mathbf{t}_k$ represent the optimal rotation matrix and translation vector computed at iteration k, whereas E_k denotes the distance error at that iteration.

(a) Implement an algorithm for the pseudocode given above. Then run this algorithm for the data sets given in this assignment (Q_data.txt and P_data.txt). Note that although we ask you to run your algorithm on these data sets and to report its results, we expect your algorithm to correctly run on other given data sets.

You can use any programming language for your implementation; but you are not allowed to use any built-in library functions except for mathematical matrix operations and SVD calculation.

It is important to note that we expect you to provide clean codes, which will also affect your grade.

(b) Calculate the accumulated rotation matrix $(\prod_{j=1}^{k} \mathbf{R}_j)$ and translation vector $(\sum_{j=1}^{k} \mathbf{t}_j)$ after the end of each iteration k.

(c) Finding point-to-point correspondences is computationally the most expensive step of the above pseudocode. At each iteration, the brute force

solution calculates the distance from each $q_i \in Q$ to every updated point $p_j \in P$ and selects the point p_i for which this distance is minimum.

(c.1) Implement first the brute force solution to make sure that your algorithm works fine.

(c.2) Implement an algorithm that works faster than the brute force solution that you have implemented previously. Make sure that the result is consistent with the result of the brute force algorithm.

Output

Your code when executed must output (print) at least the following information (on the output screen or as a text file):

- the rotation matrix $\mathbf{R}_k$ and the translation vector $\mathbf{t}_k$ computed at each iteration k,
- the accumulated rotation matrix $(\prod_{j=1}^{k} \mathbf{R}_j)$ and the translation vector $(\sum_{j=1}^{k} \mathbf{t}_j)$ after the end of each iteration k,
- the distance error E_k at iteration k,
- the difference between the accumulated rotation matrix of the brute force algorithm and that of your "more efficient" algorithm at convergence (and the same difference for the translation vectors).

What to submit

1. The source code of your implementation and a readme file that explains how to compile/execute the source code.
2. A report, containing at least the following:
   - Explain briefly your implementation.
   - Compare the time complexity of the brute force solution and your implementation using big-O notation. Discuss how your efficient solution improves the running time of the brute force solution.
   - Report the rotation matrix $\mathbf{R}_k$ and the translation vector $\mathbf{t}_k$, for each iteration.
   - Report the accumulated rotation matrix $(\prod_{j=1}^{k} \mathbf{R}_j)$ and translation vector $(\sum_{j=1}^{k} \mathbf{t}_j)$ after the end of each iteration k.
   - Report the distance error E_k for each iteration k.

- Display the 3D points Q and P before and after registration by using any convenient display tool you can find (here it is not important which tool you use for displaying; just include the displayed point sets).