BEng Biomedical Engineering
Computer Programming

Coursework 3 - Take-Home Assignment

Objective

To gain practical experience of computer programming and design using MATLAB.

Introduction

A team of engineers has developed a novel robotic device for interventional procedures. Based on pre-acquired 3D images (known as "road maps"), a target trajectory of the device is first generated to define an optimal route to reach an anatomical target. The robotic device is then automatically advanced based on the pre-defined target trajectory.

Instructions

In a first experiment, the target trajectory and actual trajectory achieved by the device have been stored into two separates files, available on KEATS. Each file contains one trajectory and is of the following format:

```
-0.00314 -0.08847 0.01996
-0.00345 -0.08737 0.01995
-0.00377 -0.08628 0.01994
...
```

where each line represent one 3D position in space in metres (as three comma separated numbers). Note that the sampling frequency in this file is 1000 Hz (i.e. one position every ms).

You are required to write MATLAB code (avoiding code duplication) to perform the following operations:

1. Read in the data from the files *targetTrajectory.txt* and *actualTrajectory.txt*. If a file does not exist report an error and stop the program. If the files contain different numbers of measurements, the program should also report an error.
   (Hint: Look at the MATLAB documentation for the `exist` *function.)*

2. Determine the mean distance (*MeanD*) and maximum distance (*MaxD*) between the actual trajectory and the target trajectory, ignoring the first and last 50 samples of both trajectories. The Euclidean distance should be used to compute the distance between each pair of points of the two trajectories. The use of the built-in function *norm* is not allowed in this implementation.

3. Determine the temporal delay (i.e. temporal shift) of the actual trajectory with respect to the target trajectory. A delay can be observed due to imperfect response of the system. Find the delay leading to the best temporal alignment of the two trajectories (i.e. the lowest *MeanD*). To find this delay, implement an exhaustive search for a range of realistic delays (from 0 ms to 50 ms, in steps of 1 ms) and identify the delay leading to the smallest *MeanD*. The

MeanD metric should again be computed by ignoring the first and last 50 samples of both trajectories.

4. Compute the *MeanD* and *MaxD* after correction of the delay, as described in Step 2.
5. Determine the total time when the delay-corrected actual trajectory was more than 2 millimetres away from the target trajectory. As such cases could happen due to measurement noise in the actual trajectory, report the total time when the delay-corrected actual trajectory was more than 2 millimetres away from the target trajectory for at least 3 ms (i.e. for three successive measurements).
6. Display the original actual trajectory and the target trajectory in subplot 1 and the delay-corrected actual trajectory and the target trajectory in subplot 2 as shown in Figure 1. Only the first 200 samples should be shown on these plots to facilitate visualisation. Annotate your plots appropriately. Display to the command window the following information: the delay, the total time away from the target trajectory by more than 2 mm, the consistent time away from the target trajectory by more than 2 mm (i.e. for at least 3 ms), and the *MeanD* and *MaxD* before and after delay correction. The *MeanD* and *MaxD* should be displayed with 7 digits after the decimal point.

For example, for the provided files, the command window output should be similar to this:

```
delay: 10
totalTimeAwayFromTraj: 98
consistentTimeAway: 0
meanD (init: 0.0000078; corrected: 0.0000033)
maxD (init: 0.0000206; corrected: 0.0000087)
```

The plot should look similar to that shown in Figure 1.

Note that you should write your program in such a way that it will work for any input files with the same format as those provided. Your code will be tested using files from different experiments.

General advice

Before starting any coding, you should produce a structured design for your program, using structure charts and/or pseudocode. You should then apply an incremental development approach to develop your code.

Reporting Requirements

You should submit MATLAB files that meet as many of the requirements as possible. Try to use variable/function naming, comments and indentation to make your program as easy to understand as possible.

You should also submit a short written report detailing the design of your program. This report should contain structure charts and possibly pseudocode for your program design, together with a short explanation of why you chose to design the program in this way. To help you in producing this report, a sample model report will be made available to you through the KEATS system. The report should not need to be longer than 3 pages of A4, and can be shorter.

Submission will be via the KEATS system. The submission point will only allow you to upload a single file so if you have multiple files you should combine all files into a single *zip* file.

The hand-in date is **14 Dec 2023, 5 pm**. Work submitted within 24 hours of the deadline will be marked but 10 raw marks will be deducted from the coursework mark. If the deduction takes

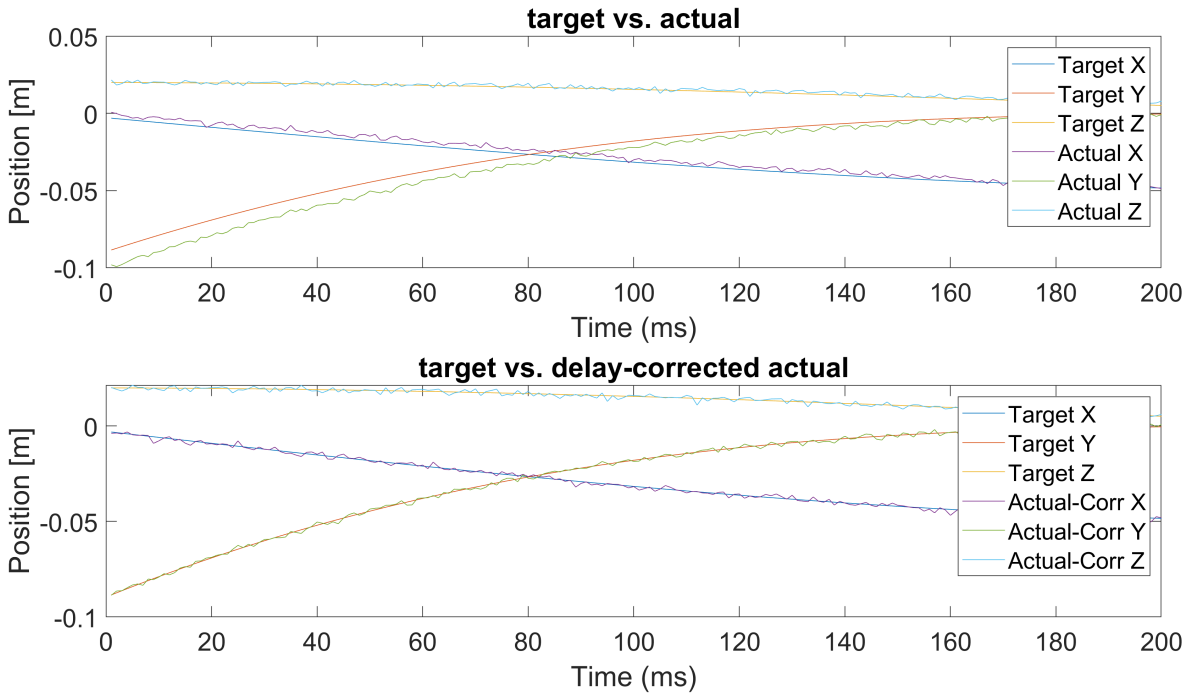

Figure 1: Plots of actual vs target trajectories over time. Top: original actual trajectory. Bottom: delay-corrected actual trajectory.

a student below the pass mark, the coursework mark will be capped at the pass mark. Work submitted after the 24 hour deadline will receive a mark of zero.

If your program does not meet all requirements then please submit what you have written by the deadline.

Assessment

Your coursework will be marked on a number of factors:

- Does the program work? Does it meet all requirements? (60%)
- Program design, i.e. structure charts and/or pseudocode (20%)
- Appropriate use of MATLAB language features, e.g. control structures, functions, etc. (10%)
- Use of comments, indentation and variable/function names to make code easy to understand (10%)

The overall mark for this coursework will make up 40% of your total mark for this module.

This is an individual assignment. You are not permitted to work together with any other student. Note that general discussions about design decisions and/or coding strategies are permitted, and such discussions can be a useful learning experience for you. But you should not, under any circumstances, share details of designs or code.

If you have any questions about this coursework please contact Sebastien Roujol (sebastien.roujol@kcl.ac.uk).