MEng Biomedical Engineering: Computer Programming Coursework Written Report

Edward Okoekpen

# Introduction

The problem description for this design pertains to Coursework 3 Take-Home Assignment concerning a robotic device for interventional purposes. This report elucidates the structured design process, employing stepwise refinement based on structure charts and pseudocode.

The program took a procedural programming approach. By doing so, I could have a dedicated function to solve each sub-problem and call them in the main function. Using functions also allows the program to be able to be used to solve the same problem using different data, as well as reusability of code to solve similar problems in future.

# First Level Factoring

The optimal solution that I discovered was to incorporate a functional programming approach, breaking down each main subsystem of the question into self-contained functions. Figure 1a illustrates the structure chart of the solution after the first level of factoring has been performed.

A diagram of a device

Description automatically generated

*Figure 1a*

The best way to read in the data to the program, was to use the *load* function as it forms the desired array structure without need for manipulation unlike using, for example, *textscan*.

# Further Factoring

The modules identified in the structure charts are now broken down further using pseudocode as it allows for a quicker and easier transfer to the actual code.

## calc\_distances()

This module calculates the distances at each sample point between the actual trajectory taken by the robotic device and the target trajectory. Taking the actual trajectory data and target trajectory data as inputs, we first calculate the Euclidean distance between each measurement by taking the square root of the difference of the squares. Then, using the built-in *max*() and *mean*() MATLAB functions, we obtain the mean distance and the maximum distances.

A screen shot of a computer program

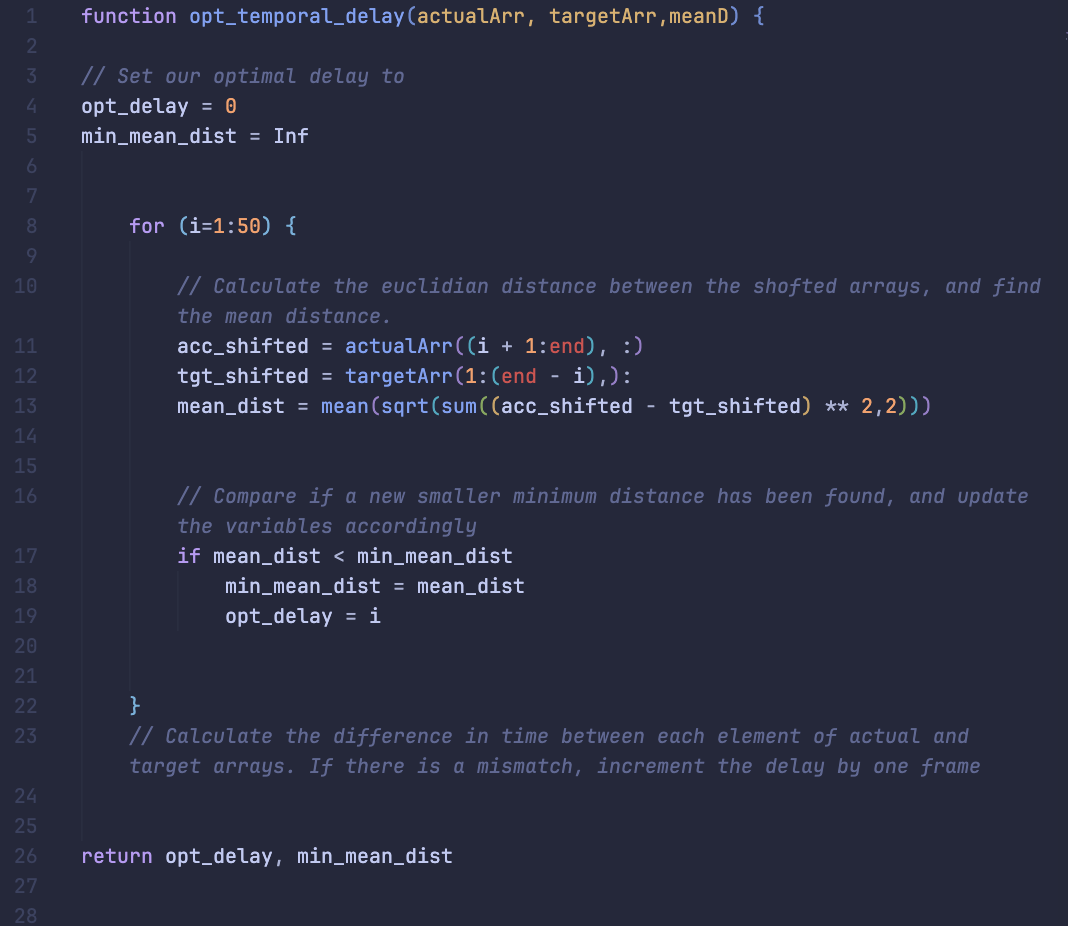
Description automatically generated

## opt\_temporal\_delay()

This module targets question 3 of the problem stem. The objective is to find the best temporal alignment between the actual and target trajectories and find the delay that results in the lowest mean distance. Our approach is to iterate over the specified delay range, which is from 1 to 50ms, and search for which gives the lowest meanD. From my understanding of the problem, for each delay in the delay range, we need to delay the actual trajectory signal and calculate the distance between this delay-corrected signal and the target trajectory. For each iteration (for each delay), find which delay will give us the smallest meanD value.

We delay our trajectories by indexing from the current delay being tested to the end for actual trajectory, and from the start to the end minus the current delay for the target trajectory.

The solution is modelled in the following pseudocode:



## time\_away\_from\_traj()

This module targets question 5 of the problem stem – to find the total time the delay-corrected actual trajectory was more than 2 mm away from the target trajectory, considering the possibility of measurement noise. For this part of the problem, the following pseudocode gives the structure used to model the problem:

A computer screen shot of a program

Description automatically generated

## display\_results()

Tis final simple module aims to produce the desired output in the command window, requested in question 6 of the problem stem. Taking in all the key values outputted by the other functions previously as input arguments, we structure our output using *fprintf* statements:

A computer screen shot of text

Description automatically generated

Any other modules are simple enough to implement directly from the structure charts.