CruiseAuto Project – Milestone 2

ANSWER SHEET: Algorithm Development

Table of Contents

Part 1. Assignment Header

Section and Team ID: 011_03

Team Member Name	Purdue Career Account Login	Programmer Number	Detailed Description of the Work	Percent of Work
Justin Clark	clar1062	2	Primary programmer on managing data noise and errors. Your code will ensure that the data are usable and will be accurate for parameter identification. This programmer worked on cleaning and smoothing the data.	25%
Peter Teal	pteal	4	Programmer responsible for finding both the initial and final speeds. This programmer worked on finding the initial and final speeds and ensured they were withing the allowed bounds.	25%
Ethan Zhang	Zhan5173	1	Create the main function and create the data visualization models to display the data, analyze it and produce results. This programmer is also responsible for coordinating the	25%

			subfunctions so they are organized and properly used in the main function. This programmer created a loop to graph the smoothed data and print the acceleration start time, Time Constant, Initial and final speeds.	
John Soares	soaresj	3	Programmer responsible for finding the acceleration start time and time constant. This programmer created the calculations for the acceleration start time and ensures it was within the allowed bounds.	25%

Part 2. Milestone 1 Feedback and Reflection

Strength: Problem statement was well formulated and described with good use of in-text citations. The main function was able to call all the subfunctions. All our approaches had evidence-based justifications.

Limitation: There was a lack of clarity in the evidence-based justification, overall having a lower quality. Additionally, a lack of commenting and well-explained function/subfunctions purposes in our skeleton model was another limitation.

How could the feedback from M1 lead to improvements? The feedback will lead us to having a more in-depth and richer evidence-based justification for all our approaches. As well as commenting on the whole code during the programming of our individual subfunctions

What concrete steps will you take to incorporate the M1 feedback to improve your approaches to managing noise and errors in data? Making clear and consistent comments throughout the data. As well as a clear explanation on the evidence-based justification with the methodology. Additionally, a well-developed and written program description will be implemented for clear understanding of the purpose of the piece of code when managing noise and errors in data.

What concrete steps will you take to incorporate the M1 feedback to improve your approaches to identifying the parameters?

Our identification of parameters in M1 suggests that we followed a clear, correct and well-reasoned path when determining the parameters of each function. Thus, following the same logic throughout the process when constructing the actual code is a must to present the best output and correct function purpose.

Part 3. Planning your Algorithm

What method(s) will you use to manage data noise and errors? Why?

To manage data noise and errors we will utilize the averaging method. This will be primarily useful for the data drops and bursts that are much larger than the actual data points at that region. Furthermore, for N/A datapoints we plan on assigning the value of the average of the

previous and following datapoints to try and reduce the number of data points lost in the process and to create a smoother curve within the averaging model.

What method(s) will you use to determine the acceleration start time, the time constant, the initial speed and the final speed? Why?

Once the data is cleaned, we will find the starting velocities and ending velocities by averaging the velocities towards the start and end of the data set. Once these variables are solved for, we will then calculate the time constant using these values plugging them into the formula from M0, and we will calculate the acceleration start time by checking the slope (change in speed / change in time) for each index in the velocity vector. Once these values are calculated we will use a loop to go through all the indexes until an acceleration higher than a given threshold is reached, we will then check this index in the velocity vector against the same index in the time vector and this will give us the acceleration start time.

How will the subfunctions operate together? Who needs what values from which functions? *Decide how team members can start working on their programs while they wait for outputs from in-development programs*.

Firstly, all the subfunctions will require clean data. The clean data is completed by the first subfunction, programmed by Programmer 1 (Justin Clark). After the data is cleaned it will be passed to the main function in order to be used by the rest of the subfunctions.

The second subfunction, Programmer 3 (John Soares), will utilize the clean data to obtain the starting acceleration time and the initial velocity as well as the final velocity to find the time constant for all the car types, and tire types respectively. These calculated values will be then passed back to the main function.

In the third subfunction, Programmer 4 (Peter Teal), will utilize the clean data to calculate the initial and final velocity for each category of car and tire.

How does the data displays need to change or be updated to help the team understand how well the algorithm is working? If it doesn't, please explain why?

The data displays will change according to the cleaning of the data after the first subfunction has completed its purpose to demonstrate to the rest of the team how the pure data looks like for a better analysis of the key points that the calculations need to look out for. This, for example, would be a good basis to understand when the acceleration starts, and thus can be used as an adequate comparison to whether the calculated acceleration starting time is near the same time value as demonstrated by the cleaned data graph. The same principle would be applied to the initial and final velocity calculations, as well as the time constant calculation.

Part 4. Programming your Algorithm

All deliverables should be submitted to Gradescope. ©

Part 5. Algorithm Reflection

Describe your process for choosing how you would develop your algorithm? How did you use your data in this process?

We started by identifying what the algorithm needed to accomplish based on the project goals. We needed to smooth noisy speed data, identify key performance parameters, and present results clearly. From there, we broke the problem down into modular subfunctions for smoothing, acceleration detection, and speed estimation. We used the structure from Milestone 1 and built off of our initial ideas to create a more refined and functional algorithm. We tested ideas directly using the provided dataset, especially one test case (Compact Winter 1), to visualize what the raw vs. smoothed data looked like and validate whether our outputs matched expected behaviors. Then once that was working, we incorporated our design into all 45 plots and used our peer design feedback to determine how we wanted to plot the graphs. (Nixon 2002)

Describe your process for making sure your algorithm is meeting the needs of the client and running smoothly. What did you do to debug your algorithm? How did you use your data in this process?

We debugged the algorithm by running each subfunction independently and checking its output with known or expected values from the data. For example, we checked for NaNs in the smoothed data and printed the first few values to catch issues early. We also added print statements to verify time constant and acceleration start calculations and used MATLAB plots to visually confirm whether the acceleration start time matched the point where the speed began increasing. If anything didn't look right, we just traced it back to its function and adjust the logic or indexing until it made sense. This helped ensure the algorithm was accurate and worked for all 45 tests.

Identify at least one strength and one limitation of your team's algorithm you created in M2.

One strength of our algorithm is its modular design. Each subfunction is focused on a single task, thus making it easier to debug, reuse, and improve without having to run the whole program. It also made collaboration more efficient, as we decided to use source control using git, being able to work and test each function individually then concatenating drastically improved our productivity and workflow. A limitation is that the smoothing function can be slow when run over all 45 tests due to the high number of passes, which could affect performance in larger datasets. Since the smoothing function had a time complexity of O(n^2)(Lindeburg 2024), we had to limit the amount of times our smoothing function ran as with one test, running 500 times was feasible, but with 45 tests it takes too much compute time, In the future, we could explore ways to optimize the smoothing step or reduce redundancy across tests.

Part 6. References

Lindeberg, T., "Discrete approximations of Gaussian smoothing and Gaussian derivatives," Journal of Mathematical Imaging and Vision, 66(5): 759–800, 2024.

Computer Vision Demonstration Website. Gaussian Filtering - Computer Vision Website Header. (n.d.). https://www.southampton.ac.uk/~msn/book/new_demo/gaussian/#:~:text=The%20Gaussian%20Smoothing%20Operator%20performs,how%20large%20the%20template%20is.

Mark Nixon & Alberto Aguado, 2002, Feature Extraction & Image Processing, Newnes