EECS 16B, Spring 2023
May 5th, 2023
Arduino ID: 82

EECS 16b Final Lab Report

Overview

The final lab report tests your understanding of EECS 16B Labs 6-9, with an emphasis on conceptual and analytical understanding. It also allows you to look at these labs from a bigger picture and reflect on your design process and choices. You may use your homeworks, pre-labs, labs, lab notes, presentation slides, and any other resources we provided throughout the semester to help you. However, all of your answers and explanations must be in your own words; you are not allowed to directly copy from those resources. The lab report questions for each lab will be released at the bottom of the corresponding Jupyter notebook. All of the final lab report questions will also be compiled together in this document and updated as new questions are released.

Requirements

Format

The report is to be done with your lab group using $\LaTeX$ or Google Docs/Microsoft Word. At the top of the report, please include the names and emails of all your group members, as well as the arduino ID you use for checkoffs.

Deliverables

For each section 1-4, complete the following:

- First, give a summary in your own words of what you did in that lab. Possible details to include: overview of the lab's objective, new components used, issues you encountered, etc. Details NOT to include: how you left your car at home, fried your BJT, or forgot to enable high-Z mode on your function generator.
- Then, answer all of the questions listed under the section header. Remember to fully and clearly explain your answers, and upload your work if necessary.

Submission

The final lab report is due on Friday, May 5th, 2023. Only one group member should submit the lab report to Gradescope and the rest of the group members should be added to the same submission.

1 System ID

Summary

In Lab 6: System ID, we needed to relate the input to the car (voltage changes) to the output (wheel velocity changes). To do so, we decided on a mechanical model using a θ value, being velocity change per voltage unit, and β, a constant offset due to any natural bias within the system. Problematically, we originally didn't choose a wide enough PWM range for the voltage input to the wheels at first so, the data analysis failed. Yet, had we chosen a wider range, the data would've stopped being linear, so we had to recollect the data into coarse_data.txt. Finally, we conducted least-squares on the nx2 data set and determined the optimal θ and β. Another issue we originally had was that we never grounded our mic board circuit and Arduino using the half rail, and instead used ground, so our measurements were off and the pulse-width modulation signals outputted by the Arduino were railing into ground (V = 0) since it can only output positive values,

Questions

1. What do θ and β represent physically, not mathematically? What are your values of θ and β, and do they reflect the car's performance while collecting data? Why/why not?

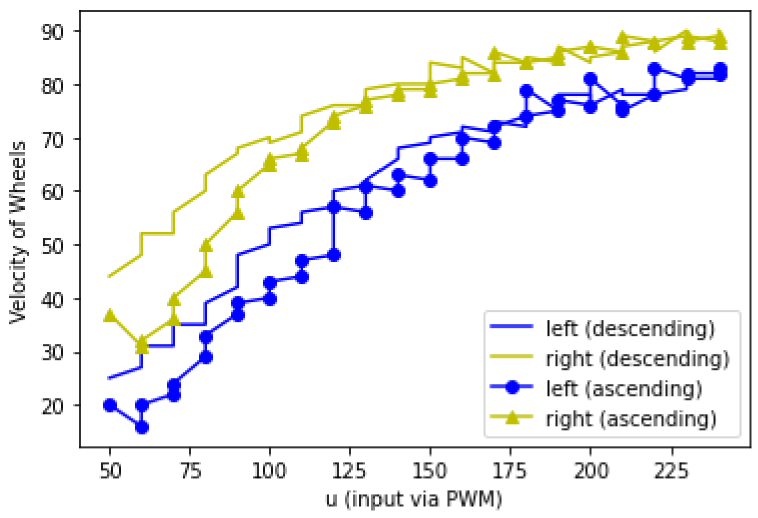
θ represents how big of a response the wheels and motor have in a change of a duty cycle. In simpler terms it acts like a sensitivity factor. β is a velocity stabilizer (offsetter) that will try to keep the velocity at a constant rate due to the impossibility of keeping a constant perfect velocity due to present imperfections (possibly air resistance or in general any considerable outside force) exerted on the car in the test field.

Values of θ and β:
Theta Left = 0.2436;
Theta Right = 0.1525;
Beta Left = -28.98;
Beta Right = -56.61;

For this case, Theta and Beta will not be a good representation of the performance of the car simply because the car is in an open loop system, meaning that the trajectory of the car is not predictable due to us not having a closed loop system that essentially acts like the feedback control for our car. As said before, as our car is an open loop system, the implications such as the input for the system not adjusting implies that the car won't be able to reflect the performance of the next time steps based on the input and current state from the open loop system.

2. How did you choose the PWM input range for fine data collection? If there were other data ranges that could have also conceivably been chosen, why did you choose this range over other ranges? Please include a graph of your car's coarse data to support your answers.

For our PWM inputs for the fine data collection, our main objective was to choose the range of values that will give us the best chance to allow our wheels on the car to reach the same velocity. For our range specifically, we choose 100-155 due the data appearing as the most linear for values that are close to ideal in our given circumstance. We can presume that it could have been more linear in terms of data in other places but through trial and error and experimenting with different ranges within this vicinity, we found out these were the best values when running the car. A best-fitted line would also be a great way to represent the ranges of the PWM.



Graph of coarse data:

3. To implement a higher order polynomial model, what would you need to change in the current lab flow to calculate the coefficients for this new model? Evaluate the benefits and drawbacks between using a higher order model vs. a linear model.

D_{data} $\vec{P} \approx \vec{S}$

$$\begin{bmatrix} u[0] & - & 1 \\ u[1] & - & 1 \\ u[2] & - & 1 \\ \vdots & & \vdots \\ u[l-1] & - & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \beta \end{bmatrix} \approx \begin{bmatrix} v[0] \\ v[1] \\ v[2] \\ \\ v[l-1] \end{bmatrix}$$

The thing we would need to change in the current lab flow to calculate the coefficients for the new model is to add more state variables.

For the matrix that is displayed above, the matrix would need more columns to compensate for the increase in state variables. The (Theta and beta) column would end up having more terms in the column in order for the amount or rows and columns to be equal to each other.

The biggest con of the higher order model is its complicated and vast complexity in terms of the work that is required to be able to develop around an equilibrium point although the biggest pro to the higher order model would be that it provides a more ideal (case-wise) fit for the equilibrium point and being able to give a much better approximation for the dataset. In most real world situations almost all the time you would never have a perfect linear model for this data, hence why on a practical level, higher order polynomial models are a better representation of real world situations. A con is that there are possibilities of small deviations off an ideal data set, but regardless still a much more practical way of looking at the data in a more realistic way, additionally with our knowledge of linear algebra we can use some key concepts from linear algebra to analyze the data from a higher order polynomial model.

4. As the batteries run low, assume that the corresponding velocities for each PWM input go down linearly. In other words, the velocity vs. PWM input curves for both wheels shift down by some value v_o, where v_o is non-negative and increases as the battery power gets lower. If we still aim for the same operating point velocity and use the same linear model, how will the car's performance change as the battery level decreases?

In short the car will eventually stop in a way where it slows down slowly in a linear decreasing fashion. If somehow the velocity of our input range (represented by the variable u in the matrix), the linear model will not be correct causing the performance of the tires to longer be equal to each other which is one of the goals of selecting a PWM range in the first place (in hopes that the wheels will turn at the same velocity).

2 Controls

Summary

In Lab 7: Controls, the goal was to implement and test both an open-loop and closed-loop path-correction system and then implement turning functionality after. We at first were pretty confused on how to approach this concept, as we were just beginning to understand the modeling behind it (how θ related motion to voltage input), and therefore didn't realize that our data was absolutely horrendous. Since our car crashed a few times, there were spikes everywhere and we were pretty sure that all our parameter values were wrong. Fortunately, we were able to re-gather better data during the week of Lab 9, which simplified the task at hand greatly as we could see noticeable improvements with alterations of the feedback and jolt values that yielded simply chaotic results with the low-quality data. Two more mechanical issues we ran into was that we couldn't understand why our left wheel wasn't turning, so to debug we switched the pins and changed a few components. Finally, we figured out that a tiny wire grounding the BJT was loose and replaced it. We also measured the motor battery, which had a low voltage of ~5 V, and replaced it with a rechargeable battery, fixing the issue.

Questions

1. What are the open loop equations of SIXT33N's control scheme for our PWM input, $u[i]$? What are some advantages and disadvantages of open loop control?

$u_L[i] = (v_L[i] + \beta_L) / \theta_L$

$u_R[i] = (v_R[i] + \beta_R) / \theta_R$

An advantage is that it's easy to implement, predetermining each wheel's input for its constants theta and beta. This works if there are no disturbances to the system (bumps, etc.). A downside is that if the car is pushed off track, there's no mechanism to realign it back on track: it would just keep going straight in the off-track direction.

2. What are our equations for closed loop control? Why might we want to incorporate closed loop control, rather than open loop, in SIXT33N?

$u_L[i] = (v* - f_L\delta[i] + \beta_L) / \theta_L$

$u_R[i] = (v* + f_R\delta[i] + \beta_R) / \theta_R$

Closed loop control allows us to factor in the current degree to which the car is off track in the input. Hence, there's a "loop" in, or connection between, the position of the car, and the change in position with time (velocity). Closed loop is hard to implement but if the car is pushed off track and $\delta[i]$ drifts from 0, then the inputs will adjust to bring the car back onto the straight path.

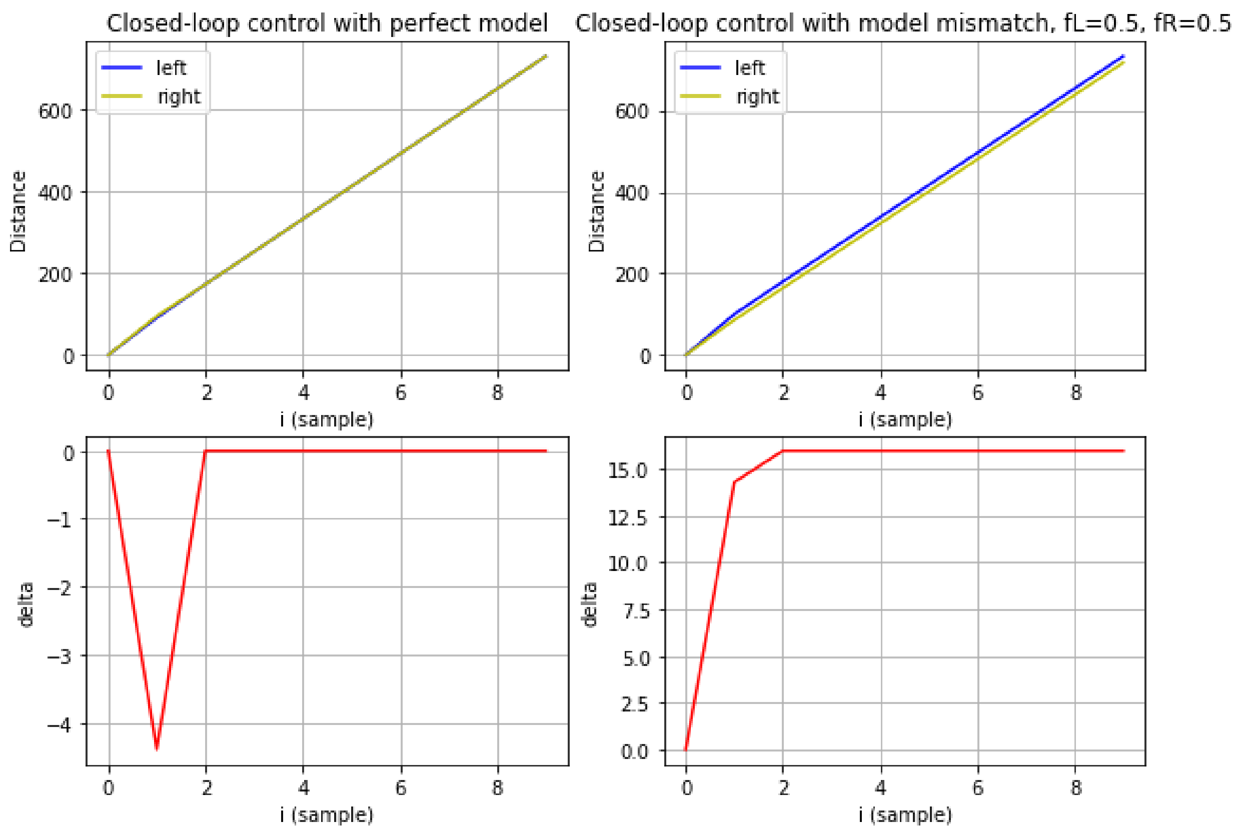
3. What is our system eigenvalue? What are the conditions on f_l and f_r such that our system eigenvalue is internally stable?

$\lambda = 1 - f_L - f_R$

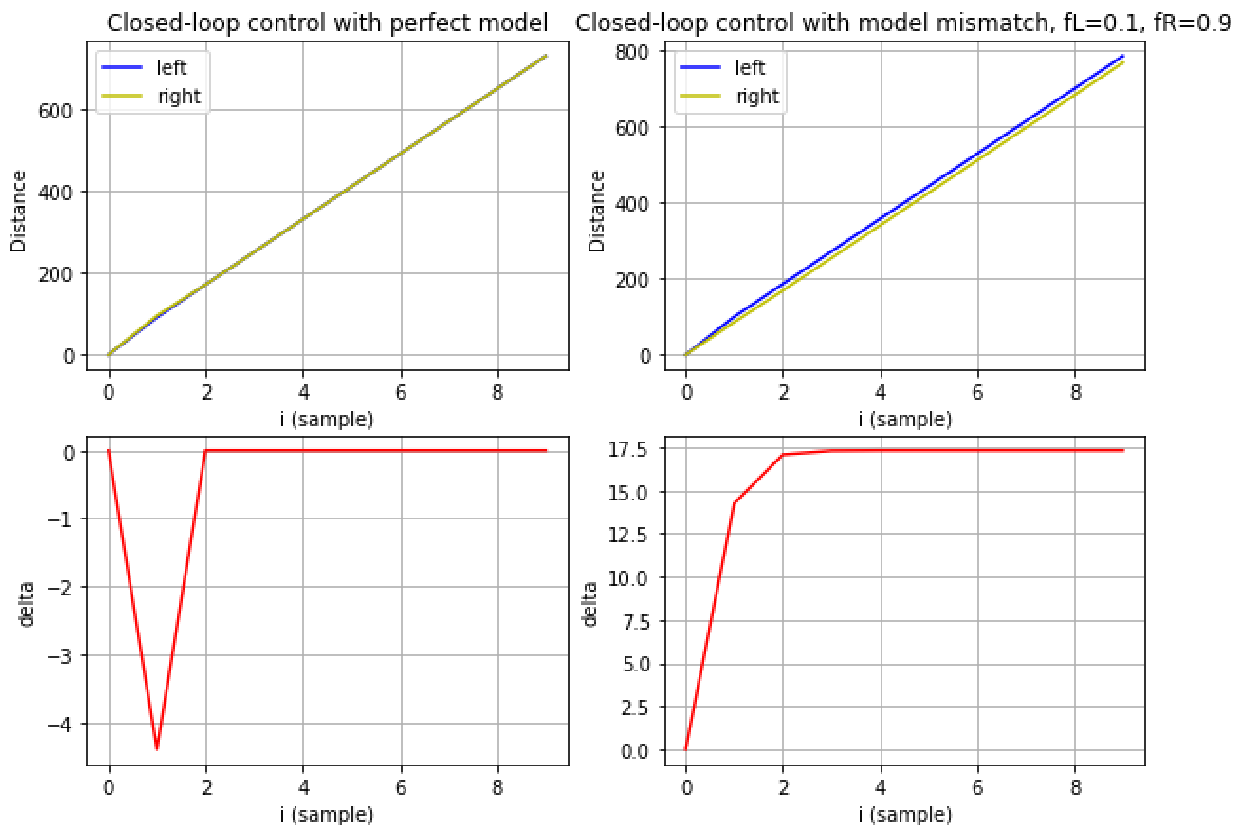
The magnitude of the eigenvalue must be less than 1 for the system to remain in a stable, decaying state due to the relationship of $\delta[i + 1] = \delta[i] * \lambda$ to the geometric series.

4. Draw the trajectory of SIXT33N with f-values of:

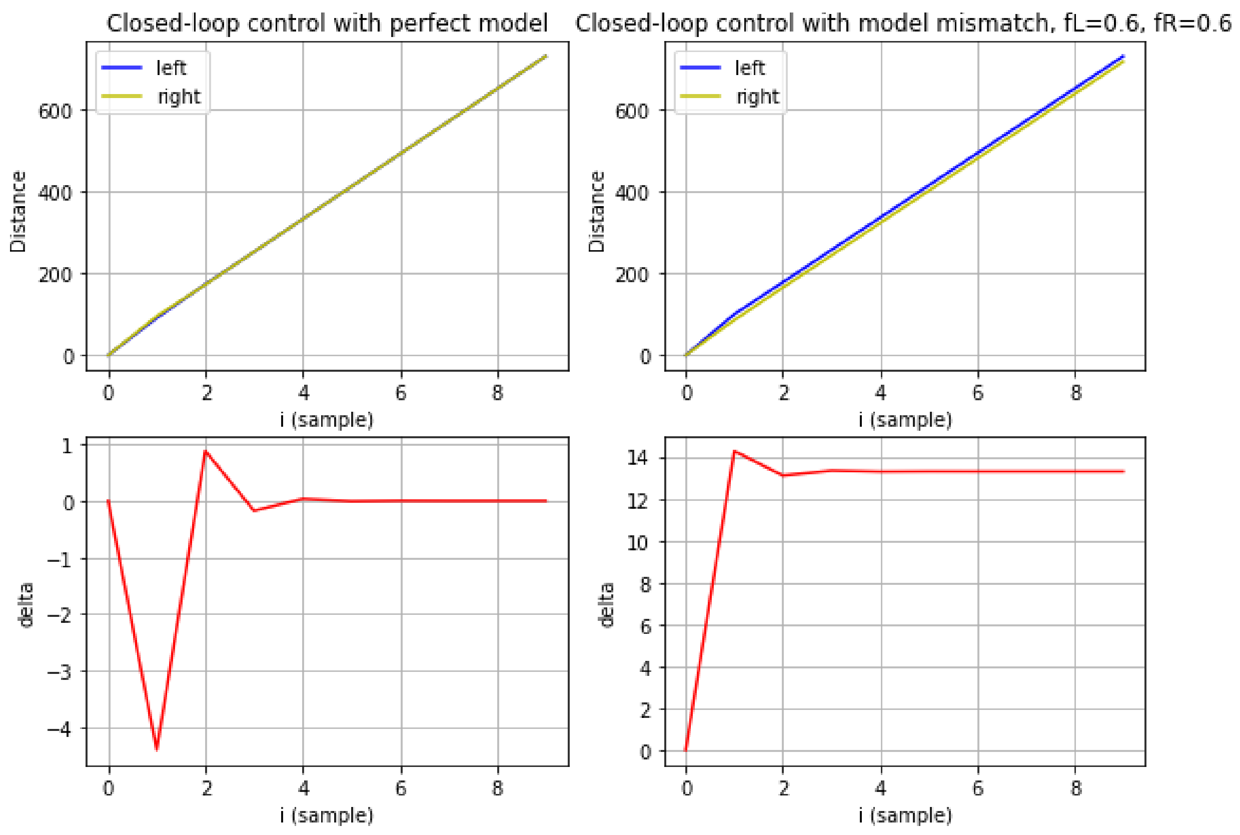
    a) $f_l = 0.5$ $f_r = 0.5$:


Closed-loop control with perfect model
Closed-loop control with model mismatch, fL=0.5, fR=0.5
left
right
Distance
delta
i (sample)

b) $f_l = 0.1$ $f_r = 0.9$:


Closed-loop control with perfect model
Closed-loop control with model mismatch, fL=0.1, fR=0.9
left
right
Distance
delta
i (sample)

c) $f_l = 0.6$ $f_r = 0.6$:



Assume your car has a nonzero, positive delta ss.

5. What is the effect of setting both $f_l = 0$ and $f_r = 0$?

$\lambda = 1 - 0 - 0 = 1$, so the system would be marginally stable, and would remain at $\delta[k]$ for all $\delta[i]$ where $i > k$.

6. What is the purpose of the jolts? Why might we have different jolts for left and right wheels?

The jolts are a very high input value at timestep $i = 0$ that allow the wheels to overcome static friction, which, being noticeably higher than kinetic friction, the wheels may not be able to overcome easily. Two main causes of difference in jolt are varying mechanical efficiencies between the motors and mass imbalance within the car.

7. Why can't we use negative f-values for both wheels? If we wanted to use negative f-values for both wheels, how should we change our closed-loop model equations such that our car goes straight and corrects any errors in its trajectory?

If $f_L < 0$ and $f_R < 0$, then $\lambda > 1$, meaning that the car's wheels' distance difference will blow up/amplify instead of diminishing. If both f-values were negative, we should make the left input's sign in "$- f_L\delta[i]$" positive and the right input's sign in "$+ f_R\delta[i]$" negative, therefore making $1 + f_L + f_R < 1$ for some f_L and $f_R < 0$

8. What does a zero delta ss value tell you about your car's trajectory? What about a non-zero delta ss value? What kind of error is it supposed to correct when we add it to our control scheme? (Hint: Think about the difference between the trajectories for a zero versus a non-zero delta ss value.)

A zero δ_{ss} value tells us that the model doesn't have any steady state error and ends up converging on a totally straight path from where it started.

When it's not zero, it means that due to random error, the car converged on a path that is slightly offset from the ideal path, since one wheel moved more in total than the other.

δ_{ss} is supposed to correct steady-state error, what wheel distance imbalance the car tends to due to circumstances.

9. How did you change the closed-loop model equations to allow the car to turn? Write the equations below and explain how they change for turning left, turning right, and going straight.

To allow the car to turn, we add an offset of $\delta_{ref}[i] = vli/r$ to the total δ, which prompts the car to fix the perceived error by turning in the opposite direction. However, we don't want the car to do a pivot turn, so we add parts of δ_{ref} over multiple timesteps. If $\delta_{ref} > 0$, then the car would turn left, and if $\delta_{ref} < 0$, then the car would turn right. If $\delta_{ref} = 0$, then δ doesn't change and the car keeps its path.

10. Describe how the trajectory of the car would look if we had a constant δ_{ref}, rather than our δ_{ref}, which changes as a function of the timestep.

If we had a constant δ_{ref} is all added at once to δ at timestep i, then the car would pivot very abruptly because the car input takes f * $\delta[i]$ as an argument. We wouldn't have any control of the angle at which we'd want it to turn.

3 SVD/PCA

Summary

In Lab 8: SVD/PCA, the objective was to use Principal Component Analysis (PCA) to analyze the main components of the audio data we fed into the Arduino. We were prompted to rehearse 4 distinct-sounding words (based on syllable count and ending sound), one for each of the four drive modes (straight far, left, straight close, right), and record them into the Arduino. We used PCA because it reduces memory size, allowing better storage by the limited-memory Arduino, and also de-noises the data. Once we project the data onto Col(U) of the SVD ($A = U\Sigma V^T$) in two or three dimensions, we can geometrically visualize the clusters and analyze the effectiveness of our word choices. We used the serial monitor of the Arduino IDE to determine if each word was properly classified or not and the oscilloscope to ensure that our voices were picked up. An issue that we ran into is that we both said each word 20 times, leading to some discrepancies between the waveforms of the word instances recorded. This required that we carefully pronounce each word in a very specific, distinct way that corresponds to both ways in which we pronounced it when recording: for example, if James' voice was slightly lower and mine was higher in specific words, we'd need to replicate that during execution and checkoff.

Questions

1. What 4 words did you choose for classification? What characteristics of this set make your words good for classification? Provide at least two features. Compared to other sets of words with similar ideally good characteristics, why is this set preferable to others?

The four words we ended up choosing for classification are pizza, helicopter, watermelon, and bee. For the word bee, it was a short one syllable word that had a strong ending noise to it from a phonetics perspective, and for pizza, it had more of a soft "ah" ending that is two syllables instead of two. As for watermelon and helicopter we choose 3 syllable words that have a softer ending than the other. Since these words for the most part were distinct in characteristics, the peaks when saying the word were more distinguishable to see a clear difference in them. In terms of data display for the centroids of these words from the six words we originally instituted, they had centroid clusters further apart from them, making classification of the word a little easier for the machine to recognize (despite our slight difficulties during this process of registering the words).

2. Why is taking the envelope of your voice signals a good choice for classification, especially considering that this classifier is implemented on an Arduino?

The envelope gives us a visual representation of the shape of the voice signal's magnitude. In relation to classification, this is highly beneficial for us because we can more easily differentiate specific words apart from each other by using their peaks and low points and seeing when those points in the data of the word occur at. Considering that the Aurdino itself has a limited data storage capacity, finding the envelope helps make the waveform of the word in a simpler way that takes up less memory but also in a way where data is not compromised to a point where the differentiation of each word is indistinguishable, the envelope does it in a simple and still distinguishable way to find differences in the given words for the Audrino to register.

3. Why do we need to use SVD/PCA to represent our data set?

SVD is used so we can use PCA to perform data compression for a working memory size considering that the Audrino is not ideal in terms of memory storage, we also use SVD/PCA to ensure that we do not lose a lot of crucial information while doing data compression

4. If we were to use the transpose of our data matrix for SVD, which rows/columns of which matrix correspond to the principal component vectors representing the recorded words? Why?

Transposing the data matrix for SVD means that the rows would have the different time steps and the columns would have the recordings of the different words (for saying multiple times of the same word). In the "u" matrix the columns would be utilized because it contains the column vectors that are corresponded to the PCA vectors that are the input matrix's columns respectively represented by A. The columns of the U are relational to the recorded word's component vectors (the principal ones).

5. How many basis vectors are you using, and how did you choose this number? What are the benefits vs. tradeoffs of increasing or decreasing the number of basis vectors by a small amount? What about increasing the number of basis vectors by a large amount?

For the sake of the limited memory storage that the Aurdino can hold, we ended up just using 3 basis vectors. Increasing the number of basis vectors can be detrimental in the sense that the possibility of picking up more (unnecessary) noise is more plausible, even more so from a logistical end, the limited memory of the Audrino for us to even use an increased amount of basis vectors. In contrast decreasing the amount of basis vectors makes it a little more practical in relation to the memory storage of the Aurdino and makes it faster to classify and process the data due to less vectors needing to be processed but a possible con is that it can potentially undermine/underfit the data.

6. What are length, prelength, and threshold for our data pre-processing? How does changing them affect your alignment? Include both the definitions and the values you chose. What kinds of words are better suited for our pre-processing method with live classification?

Length - The total number of samples the registered word takes for saying it in our recordings
Pre-Length - Before a certain threshold is reached this is the number of samples that will be integrated for a certain word
Threshold - In simple terms it is a limit. More specifically, a max value's fraction that the preprocessor utilizes in order to indicate when the action of saying a certain word is considered as "started". EX: If the loudness threshold (LT) is 8 and we say something at LT 7, it won't start registering as anything in terms of starting to say the whole word and in fact it will not pick up, but if we say something at LT 9 it will indicate to the threshold that it has been met and it can start processing the word (noise) we input into it.
In short a threshold is utilized in a way where the threshold multiplied by the max value is equal to the value of the sample itself. The way we change our threshold manipulates the mic board in a way where it affects the way how a word/noise is picked up whether it gets registered or not (misclassified or not). When a word according to the mic board is "found" it refers back to the pre-length amount of samples to try to find the start of saying the word. Changing the amount of pre-length values affects how much further back to find the start of a registered word, if the amount is too much than too many samples can occur, and if the amount is too small then it is possible that it will not go far enough to find the start of the word. The length of a word is more self-explanatory in which it is simply the length of the word. The length and pre-length work hand in hand to scavenge through all samples of a word to indicate a possible match of a registered word/sound. To maintain the same length, we need to make sure that we have the preprocessed data. Changing the length value will change the number of samples involved when trying to classify a specific word.

7. Why can we simply take the dot product when projecting our recorded data vector onto the principal component vectors?

If we take the dot product of the recorded data against the principal component vectors we can see how much the data of the recorded vectors and principal component vectors are aligned amongst each other. To capture the maximum data from the dataset we need to make a visual representation of the max projection.

8. What is EUCLIDEAN THRESHOLD? What is LOUDNESS THRESHOLD? Include both the definitions and the values you chose. During live classification, which threshold was more difficult to satisfy, and how do you know?

The distance between the nearest centroid is the Euclidean Threshold (ET), and the Loudness Threshold (LT) is used to see if a sound or noise that is made has a possibility for classifying a word, and any sound that is above this threshold acts like an activator for the start of classifying a word. For example, if the LT is 8 and I talk at a LT of 9, the mic board will try to classify based on the data what word I most likely have said. In contrast, if I talk below the threshold it will not start to classify a word and register it as "background noise".

Euclidean Threshold: 0.3
Loudness Threshold: 150

I think both were hard to satisfy in terms of finding a balance between the ET and LT for finding the most ideal situation where we can minimize background noise and maximize accuracy of classification. But more so the Euclidean Threshold was much harder to find mainly because we had to match a perfect-like pitch for the word to register properly and execute the correct command which was in some ways frustrating because any background noise can cause the distance of the nearest centroid in the data set to be further apart from the dataset of the actual word we are trying to say. For example, for the word pizza, we need to say the syllable pi (soft), and the last zza (in a more demanding and stronger way) in order to classify the word correctly to execute the desired command of driving straight for a far distance. The LT was a little easier to figure out as we just started from a high number of 180 than narrowed down to 150 as the most ideal for our situation.

9. How different was live classification in practice from what you found in the SVD/PCA lab? Why do you think that is?

The way that the classifying component of this lab worked in terms of the live classification aspect of it is that the classiger was matching the sound/word trying to be said to analytic graphs of a word that was stored in the mic board as we registered data (the words we registered to the micboard) based on the graphs the words were classified as. During the times we mispitched our voices to say a certain word or if there was background noise, it usually misclassified the word we were trying to say, so the prevent this we went to a spot with more ideal conditions with less background noise to minimize possibility of misclassification of a word, although it was not always perfect for us.

4 Feedback

1. Extra Credit: To receive extra credit, please provide 1-2 sentences for each of the following parts:

   a) How well do you feel that this assignment evaluated your understanding of labs this semester?

   This assignment really forced us to think about what we did during the labs and the circuitry/processing under the hood. Led us in a helpful direction of thought!

   b) How much time do you think this assignment took you to complete? (Just a number is fine for this part.)

   It probably took around 7 hours for each of us.

   c) What changes to lab reports would improve your experience?

   The length of this report is quite reasonable. I think the midterm lab report was way too long since we had 5 labs instead of 3. I would definitely shorten the questions for each question in the MT report to keep around the same length.

2. Additionally, please feel free to provide any feedback you have about the 16B lab or anything we can do to better support you.

The labs are pretty hard, but I think overall what we built was really cool. Certain TAs, like Minyang and Andrew, really helped us out and we couldn't have grasped what we were doing without them. Would definitely try to provide more support for the homework too: office hours are often backed up and it can just take forever if you're falling behind, leading to a slippery slope of confusion. Maybe more intuitive hints or examples on the assignments?