Design and Implementation of a Learning Enabled, Voice Controlled Robotic Car

William P. Smith
Dakota Sproch
Pei Jie Sim
Anh Nguyen
Girmai Legese

University of California, Berkeley
Electrical Engineering and Computer Science
Demonstration Link: https://www.youtube.com/watch?v=nk3QCyXg5XI&feature=youtu.be

***Abstract*— The SIXT33N is a mobile robot on 3 wheels (2 drivable) that moves around according to some input. It uses the MSP430 Launchpad as its guts with some circuitry for driving the motor and sensing through a microphone. It also runs on a 9V battery, so we don't have to chase it around as it moves. In Micro and Frond End, we designed SIXT33N's ear. The design uses a microphone front end circuit that converts audio and speech into a signal. The signal is then processed, to include applying some level shift, gain, and filtering higher frequencies (low pass filter circuit) to avoid aliasing. This is then sent to the Launchpad ADC.**

SIXT33N recognizes 4 different voice commands. It will then move forward, slow down, turn left, and turn right based on these commands. Collected voice signals are projected into a lower dimensional subspace based on their principal components (PCA/SVD) and then classified based on a training model. If the classification is within a certain threshold, the command will trigger the controller to send an appropriate control input.

Introduction

The following report outlines the design and implementation of a voice controlled robotic vehicle. We begin with the circuit layout and schematics, describing the circuit analysis and functionality, as well as the prupose of all components, highlighting issues that were found and corrected for along the way. We then transition into the PCA classification implementation, which highlights the algorithms used, data processing performed, and the software level functionality of our system. We describe issues that arose along the way, and how we were able to make this phase of the system robust despite the many free variables (noise, inherent differences in the sample data due to different user voices, correlation between words, etc.). Finally, we discuss our state space control model and controller design using feedback. We conclude with what we learned along the way.

I. Circuit

A. Microphone and Front End

Figure 1 outlines the circuit that amplifies and level-shifts the signal so that the output voltage ranges from 0V to 3.3V when the loudest sound is applied. In order to account for microphone and DC Bias, we tune the potentiometer so that, for very high amplitude sounds near the microphone, the signal at the output is not clipped and is centered around 1.6V (DC voltage at the output = 1.6V).

The input signal is very small. In order to make the output signal big enough, we need to make the gain of 150 by using the non-inverting amplifier :

$$\left(|Gain| = 1 + \frac{300k\Omega}{2k\Omega} \approx 150\right)$$

The input of the amplifier (Node A) is connected with (1.6V from the DC bias) + (the input signal from the MIC). Node B of the amplifier is connected with the output 1.6V from the voltage divider:

$$\left(V_B = \frac{16k\Omega}{16k\Omega + 34k\Omega} \times 5V = 1.6V\right)$$

We add the buffer after the voltage divider to clear out the load and make the signal more constant.

B. Low Pass Filter

The Launchpad will sample every 0.35 ms; hence, in order to remove noise from aliasing, we make a second order filter with cutoff at 1.5kHz.

$$f = \frac{1}{2\pi RC} = \frac{1}{(2\pi)(10 \times 10^{3}\Omega)(10 \times 10^{-9}F)} \approx 1.5kHz$$

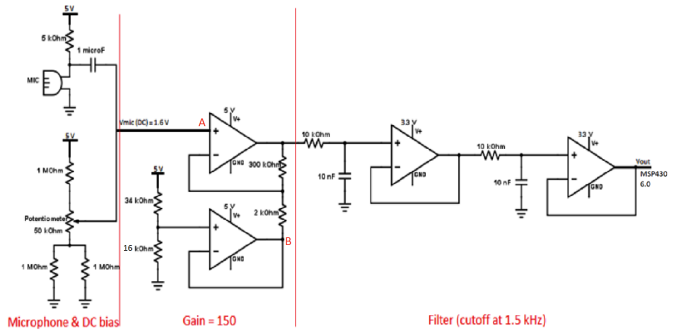

Figure 1. Front End Schematic

C. *Controls*

We use one of the batteries exclusively for the motors, and the other for powering the Launchpad and microphone front end. The reason for this is that the motors consume a great deal of current, but the current is not constant and, in fact, is very noisy. If we were to use the same voltage source for the Launchpad and the motors, the Launchpad would restart every time the motors pulled too much current. Figure 2 contains the block diagram for the power distribution.

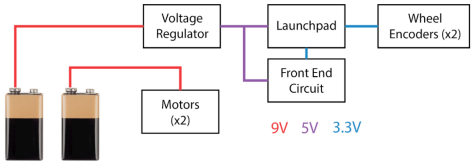

Figure 2. Power Distribution

The Launchpad and the front-end circuit needs a 5V source, so we need to add a voltage regulator circuit. The LM34015 is a voltage regulator that outputs around 5V. Two capacitors help decrease noise even further. The switch helps make it easier and safer to turn the battery on and off.

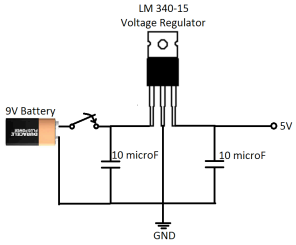

Figure 3. Battery and Voltage Regulator

In order to actuate the motors, we connect the motors to their own separate 9V battery and use an NPN transistor to control switching of the motor power by the Launchpad. When the Launchpad outputs a 0V, there is no current flowing into the base, so the BJT does not allow current to flow through the motor. When the Launchpad outputs a logical high (3.3V), current is allowed to flow through the base and turn the motor on. However, BJTs can be burned by running too much

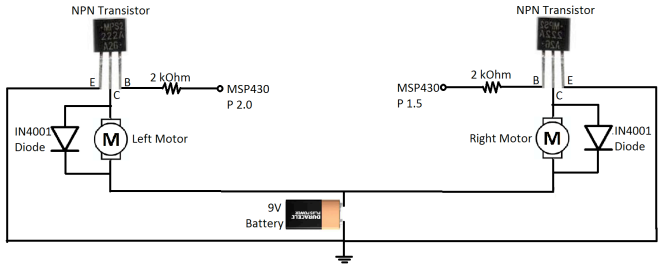

Figure 4. Motor Drivers Schematic

current through them, so a 2KΩ resistor is used to limit the current.

A diode is utilized for robustness and safety of the circuit. If the voltage in the anode is higher than the cathode (see diagram below), it will conduct current from the anode to the cathode and behave somewhat like a closed switch. However, if the voltage in the anode is lower than the cathode, it does not conduct current and behaves like an open switch. This diode is needed because of the inductive behavior of the motor. Without this addition, we risk harming the BJT.

Finally, we use wheel encoders to register the movement of the wheels. In this way, we can track rotation of the wheel and use this as a state variable in the control system. Figure 5 outlines the schematic for the wheel encoders.

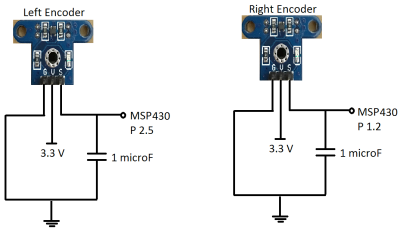

Figure 5. Wheel Encoders Schematic

II. PCA Classification and Controls

A. *Word Choice and PCA Classification*

We define four voice commands for the control of the SIXT33N: “giddy up,” “take it easy,” “port,” and “starboard.” Other options that were explored include “go,” “straight,” and “heel,” but these words yielded a low classification percentage, possibly due to similar intonation as well as the number of syllables of in the words. However, “port” and “starboard” were distinct enough that we were able to obtain 100% classification accuracy in the classification phase. This is likely because the associated amplitude envelopes of their audio signals appear distinct, causing the L2 norm of the resultant data clusters to be quite large.

The audio processing and training model consisted of four stages. Stage one consists of collecting training data sampled from 4 different users giving the listed voice commands. This data is processed by first running an envelopment algorithm to create a waveform, and then a thresholding algorithm in order

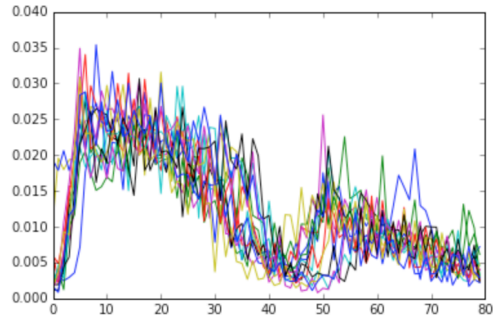

Figure 6. Sample Data Collection Envelope on Training Signals

to throw out noise. This processed data is formed into a matrix in stage 2, where we then take the Singular Value Decomposition. We then perform Principal Component Analysis on the factorized matrix, inspecting the singular values. As shown in figure 7, in the case of the enveloped audio signal data, we found that the signals were sufficiently represented by two principal components. The third phase is to project the training data into the lower (two) dimensional subspace spanned by the two principal vectors from the PCA phase. Finally, phase four is running K-means on the resultant data set in order to classify the data into 4 distinct clusters. These clusters serve as the learning model for the Launchpad, so that new data can be classified.

B. Control

We determined the open loop model for each wheel separately since both wheels are symmetric. We performed least squares regression on the data collected on each wheel row by row. To perform linear regression, we define:

$$A = u[k]$$

$$T_s = 200ms$$

$$y_1 = d[k+1]\text{-}(d[k] + T_s{*}v[k])$$

$$y_2 = v[k+1]\text{-}(0 * d[k] + v[k])$$

$$b_i = (A^T A)^{-1} A^T y_i \qquad \forall i \in \{1, 2\}$$

To find the closed loop eigenvalues, we define

$$x[k] = \begin{pmatrix} d[k] \\ v[k] \end{pmatrix}$$

$$u[k] = [\text{-}F_1 \ \text{-}F_2]\, x[k]$$

$$x[k+1] = Ax[k]\text{-} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} (F_1 \quad F_2) x[k]$$

$$= \left(A\text{-} \begin{pmatrix} b_1F_1 & b_1F_2 \\ b_2F_1 & b_2F_2 \end{pmatrix} \right) x[k]$$

$$\begin{pmatrix} d[k+1] \\ v[k+1] \end{pmatrix} = \left(\begin{pmatrix} 1 & T_s \\ 0 & 1 \end{pmatrix} \text{-} \begin{pmatrix} b_1F_1 & b_1F_2 \\ b_2F_1 & b_2F_2 \end{pmatrix} \right) \begin{pmatrix} d[k] \\ v[k] \end{pmatrix}$$

$$\begin{pmatrix} d[k+1] \\ v[k+1] \end{pmatrix} = \begin{pmatrix} 1 - b_1F_1 & T_s\text{-}\, b_1F_2 \\ \text{-}b_2F_1 & 1 - b_2F_2 \end{pmatrix} \begin{pmatrix} d[k] \\ v[k] \end{pmatrix}$$

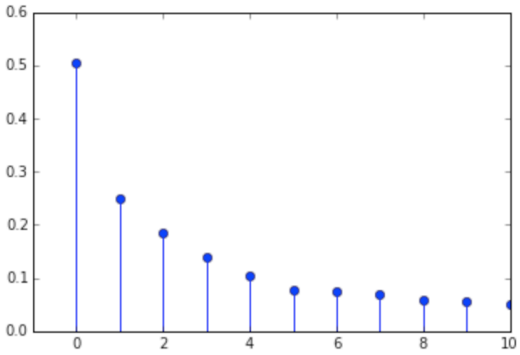

Figure 7. Visualization of Singular Values of Training Matrix

$$\det\left(\begin{pmatrix} 1 - b_1F_1\text{-}\lambda & T_s\text{-}\, b_1F_2 \\ \text{-}b_2F_1 & 1 - b_2F_2\text{-}\lambda \end{pmatrix} \right) = 0$$

$$(1 - b_1F_1\text{-}\lambda)(1 - b_2F_2\text{-}\lambda)\text{-}(T_s\text{-}\, b_1F_2)(\text{-}b_2F_1) = 0$$

$$\lambda^2 + (b_1F_1 + b_2F_2\text{-}2)\lambda\text{-}(b_1F_1 + b_2(T_sF_1\text{-}\, F_2) + 1) = 0$$

Define λ_1 and λ_2 to be the closed loop eigenvalues. Then

$$(\lambda\text{-}\, \lambda_1)(\lambda\text{-}\, \lambda_2) = 0 \quad \rightarrow \quad \lambda^2 + (\lambda_1 + \lambda_2)\lambda + (\lambda_1\lambda_2) = 0$$

By comparing the coefficients, we get

$$b_1F_1 + b_2F_2\text{-}2 = \lambda_1 + \lambda_2$$

$$(b_1F_1 + b_2(T_sF_1\text{-}\, F_2) + 1 = \lambda_1\lambda_2$$

By using the values of b_1 and b_2 found from the last step and selecting the appropriate values for λ_1 and λ_2, we determine the controller gains F_1 and F_2.

C. Control Command Implementation

The vehicle has two modes of operation: Drive Mode, and Listen Mode. The default mode is Listen Mode, during which SIXT33N is idle and periodically listening for voice commands. During listen mode, the speaker circuit is active, any audio signals recorded are captured and compared to a loudness threshold. This eliminates attempted classification of noise. If a signal is above the loudness threshold, then the data is enevoloped, projected into the subspace of its principal components, and classified.

The classification distance from the centroids of the training set are compared by L2 norm. If the L2 norm of the signal vector is outside of a certain threshold from any training centroid, the signal isagain assumend to be noise, and no action is taken. If the classification yields an L2 norm within the threshold, then the appropriate control function is activated. Implementation of straight and slow movement is straightforward, and accomplished through appropriate pulse width moduation signals to the motors.

Each wheel controller has two inputs, namely distance of the opposing wheel and the desired velocity. That is, the left

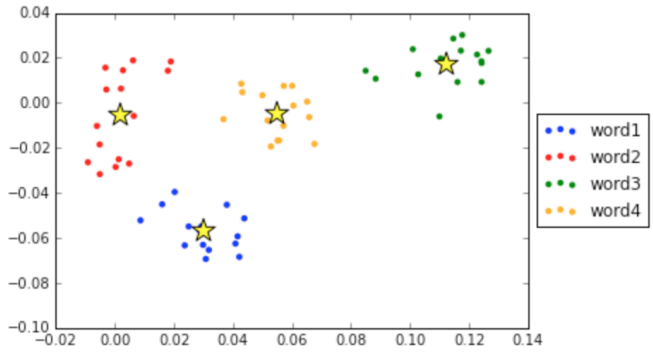

Figure 8. K-Means Classification of Training Data

wheel has as input the right wheel distance $x_{right}[k]$ and the desired velocity $v_d[k]$, while the right wheel controller has two inputs, the left wheel distance and the desired velocity. To implement left and right turn functionality, we manipulate its position global variable to make the car "believe" that it has turned more than it really has, thus causing it to straighten using the feedback mechanism implemented in the previous step. This causes assymetric inputs to the two wheel motors, thereby causing a turn.

III. Conclusions

The project as a whole was very interesting and we definitely learned a great deal from it. First of all, being able to control a vehicle using only voice commands is remarkable, but understanding the underlying concepts associated with getting the car to recognize and process those commands is

even more enlightening. Through completing this project, we had the opportunity to put the concepts that have been taught in 16B to use and that certainly enforces what we already know as well as helping us incorporate such concepts in real-life applications.

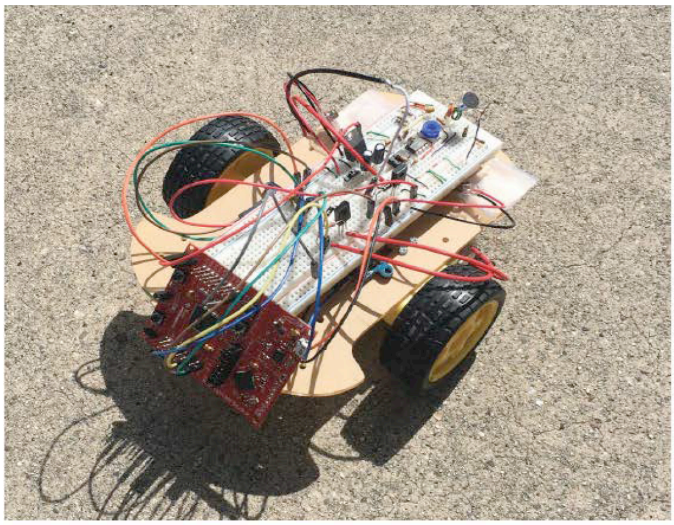

Figure 9. SIXT33N Robotic Car

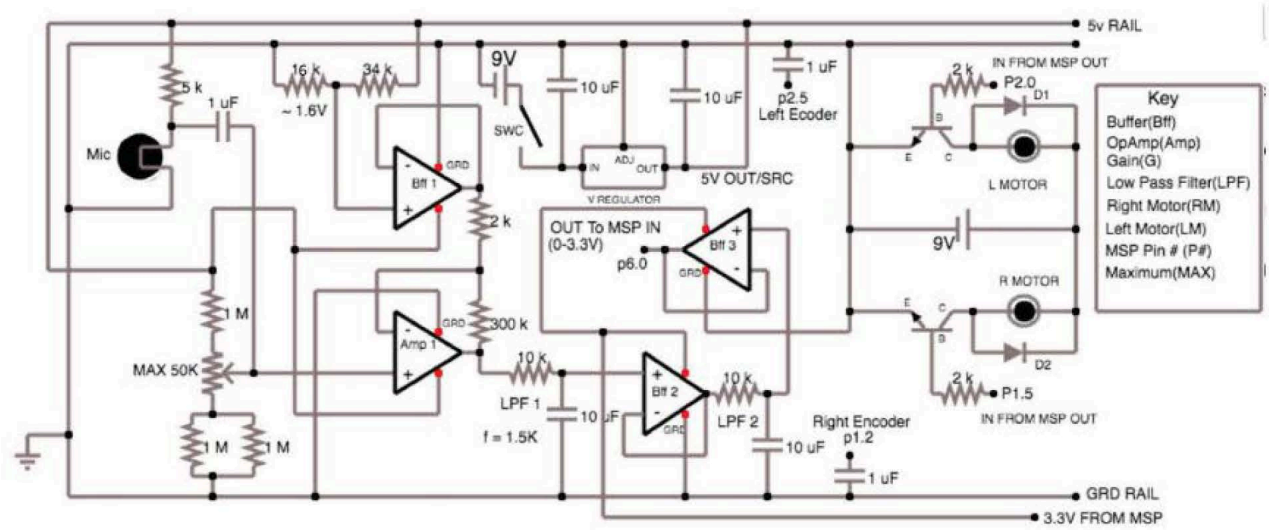

Figure 8. Full System Circuit Schematic