Speech Recognition through Mel Frequency Cepstral Coefficients and Dynamic Time Warping

Thomas Smallarz

13 December 2020

Contents

Introduction

Speech is a fundamental mode of communication between people, even in this age of non-verbal communication (email, text, etc.), so the benefits of being able to translate human speech to a digital format is obvious.

This report will explain implementing a method of speech recognition in MATLAB R2020b using Mel Frequency Cepstral Coefficients and Dynamic Time Warping with the intention of implementing this method on a low-cost embedded target.

1 Method

Most modern automatic speech recognition (ASR) systems have the same structure. It is as follows:

1. Speech is recorded and digitized
2. Features of waveform are extracted
3. Strings of phonemes are recognized based on features
4. Words and strings of words are assembled based on language models

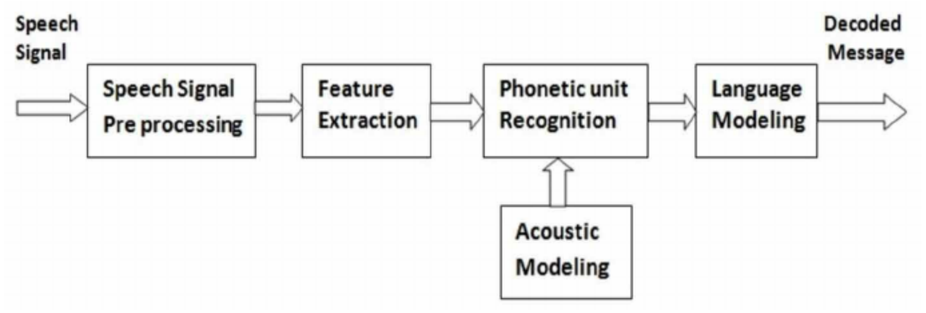


Figure 1: General speech recognition system diagram

As of now, we won't concern ourselves with the first step – recording and digitizing audio. This will be talked about a little in the Implementation section later on. Also, language modeling will not be discussed in this report.

1.1 Mel Frequency Cepstral Coefficients

Before using algorithms to determine the likelihood of speech utterances to be specific words, the data is typically manipulated to gain characteristics that describe what is being said.
In this implementation Mel Frequency Cepstral Coefficients (MFCC) are used. There are numerous ways to extract features or characteristics of speech. This is one method that is used.

1.1.1 Cepstral

To understand how to derive coefficients of the MFC, it is first important to learn what the "cepstrum" is. The cepstrum of a signal is the result of taking the inverse Fourier transform (IFT) of the logarithm of the Fourier transform (FT) of a signal. Or,

$$|\mathcal{F}^{-1}\{\log|(\mathcal{F}\{x[n]\})^2|\}|^2$$

1.1.2 Mel Scale

The Mel scale is a scale based on the perceptions of human hearing. It is a mapping function that separates pitches by what humans consider to be equal distance. A common formula to convert frequency from hertz into mels is:

$$m = 2595 log_{10}(1 + f/700)$$

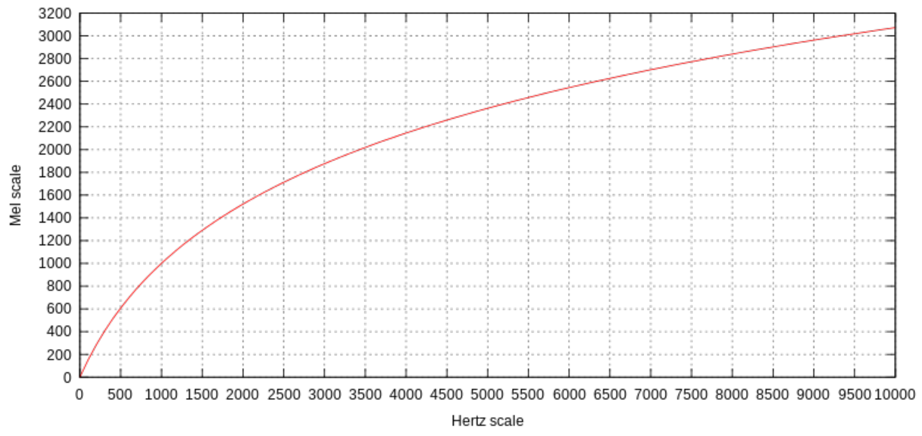


Figure 2: Mel scale vs. hertz

1.1.3 Mel Frequency Cepstral

Including both these concepts, the difference between Cepstral and Mel Frequency Cepstral is that MFC has its initial spectrum mapped to the mel scale before taking the logarithm. The common derivation of MFCC's is as follows:

1. Take the Fourier Transform of signal
2. Find the powers of that spectrum
3. Map onto the mel scale (typically with triangle filters)
4. Take the logs at each mel frequency
5. Take the discrete cosine transform (DCT) of the mel log powers

Then, the MFC coefficients are the amplitudes of the resulting spectrum.

1.2 Dynamic Time Warping

Dynamic Time Warping (DTW) is an algorithm that measures the similarity between two sequences in time (or other dependent variable) that may vary in speed. It calculates a distance matrix, then the shortest weighted path of that matrix is the ”distance” sequence used to map one of the signals onto the other. So, with A_i and B_i sequences

1. Create distance matrix of size i x j
2. Find shortest weighted distance from opposite corners of distance matrix

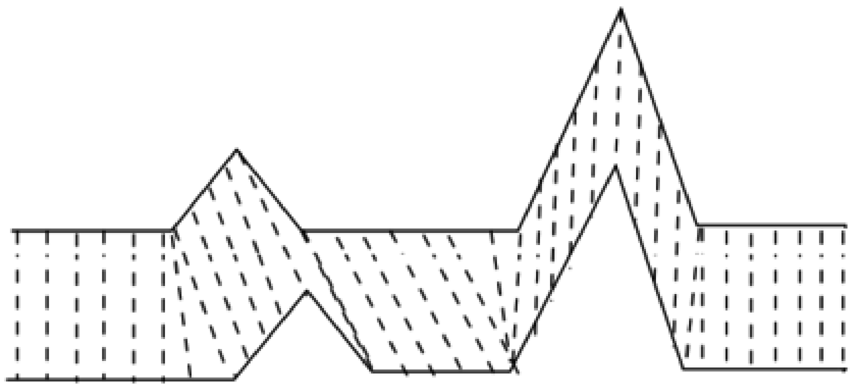


Figure 3: Example of mapping one sequence onto another using DTW

1.2.1 Creating distance matrix

Creating the distance matrix is done with a simple equation. If the values of A_i and B_j are put on row and column side of the matrix respectively, then the values at $D_{i,j}$ are:

$$D(i,j) = |B_j - A_i| + min(D(i-1,j-1), D(i-1,j), D(i,j-1))$$

This is calculated starting at the first place in the sequence ($D_{0,0}$) and working towards the last place ($D_{i,j}$). Values $D(i-1,j-1)$, $D(i-1,j)$, and $D(i,j-1)$ are used when they have been already computed.

There are more computationally simpler methods of computing this distance matrix, but in short, this is how it is calculated.

1.2.2 Finding shortest path

Once a distance matrix is created we need to find the values to warp the two sequences by to to align them. These values are found by starting in the upper corner of the distance matrix ($D(i,j)$) and recursively find the minimum of $D(i-1,j-1)$, $D(i-1,j)$, and $D(i,j-1)$ until reaching the lower corner ($D(i,j)$).

The amplitude values of the distance matrix could be likened to the height of a mountain range, and the path you take to the origin is the valley between the mountain peaks.

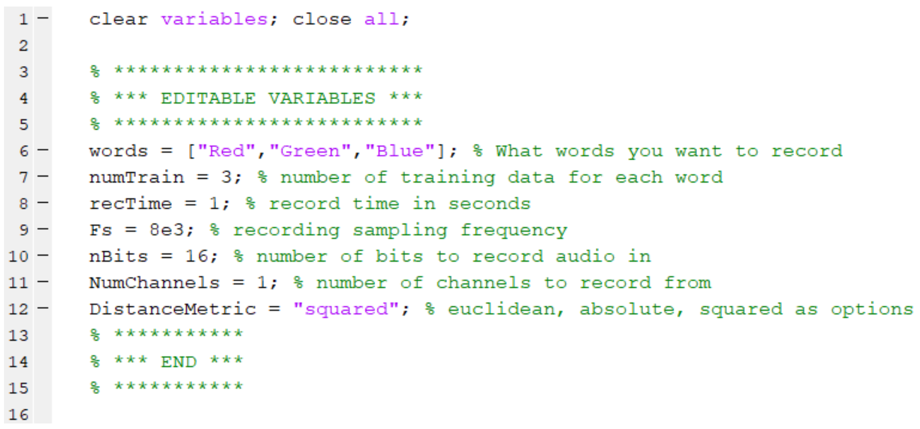
2 Implementation

Now, armed with the means to find the MFCC's and perform a DTW. We can create some training data (multiple recordings of some word) and have an input and see if our system can recognized the inputted word.

2.1 Digitizing Audio

First is to record the audio. In MATLAB there is a audiorecorder( ) function that uses your default system microphone and has parameters: sampling frequency, bit size to store data in, and number of channels to record.

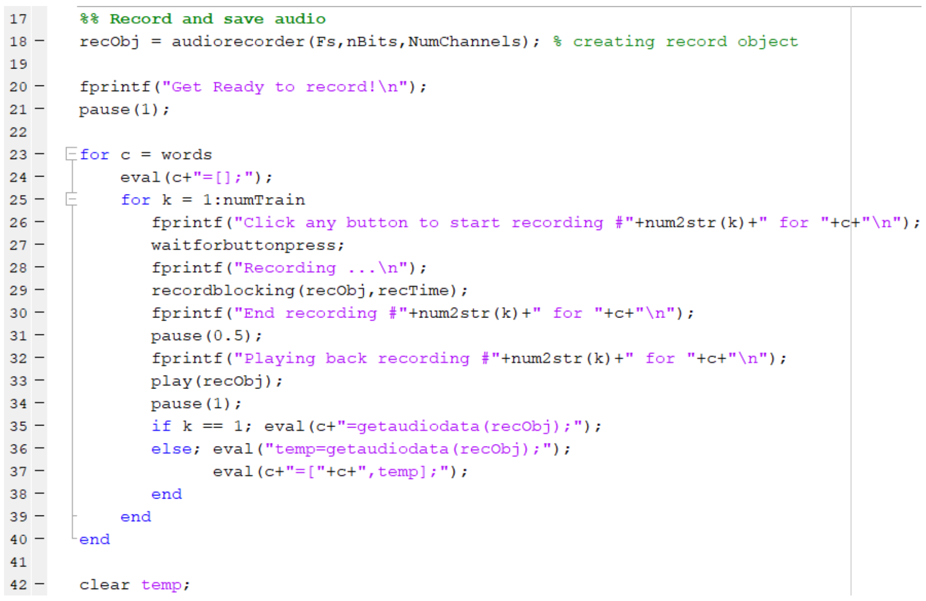
- Sampling Frequency: A quick Google search brings us to the Voice Frequency Wikipedia page that says a sampling rate of 8,000Hz is sufficient (a 4000Hz bandwidth doubled)
- Bit Size: Let's go with 16-bit Integers for now. Implementing on lower-cost hardware may require less size
- Number of Channels: One channel for one microphone



```
clear variables; close all;

% **************************
% *** EDITABLE VARIABLES ***
% **************************
words = ["Red","Green","Blue"]; % What words you want to record
numTrain = 3; % number of training data for each word
recTime = 1; % record time in seconds
Fs = 8e3; % recording sampling frequency
nBits = 16; % number of bits to record audio in
NumChannels = 1; % number of channels to record from
DistanceMetric = "squared"; % euclidean, absolute, squared as options
% ***********
% *** END ***
% ***********

```

Figure 4:

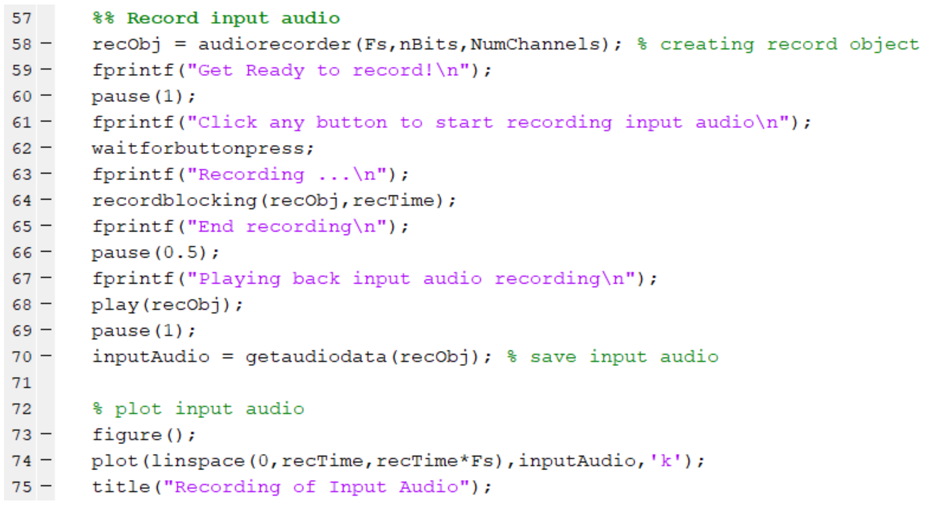
Now, we create an instance of the audiorecorder and record each word "numTrain" times with a one second duration to record.



```
%% Record and save audio
recObj = audiorecorder(Fs,nBits,NumChannels); % creating record object

fprintf("Get Ready to record!\n");
pause(1);

for c = words
    eval(c+"=[];");
    for k = 1:numTrain
       fprintf("Click any button to start recording #"+num2str(k)+" for "+c+"\n");
       waitforbuttonpress;
       fprintf("Recording ...\n");
       recordblocking(recObj,recTime);
       fprintf("End recording #"+num2str(k)+" for "+c+"\n");
       pause(0.5);
       fprintf("Playing back recording #"+num2str(k)+" for "+c+"\n");
       play(recObj);
       pause(1);
       if k == 1; eval(c+"=getaudiodata(recObj);");
       else; eval("temp=getaudiodata(recObj);");
             eval(c+"=["+c+",temp];");
       end
    end
end

clear temp;
```

Figure 5:

After this we can record our input audio using the same method.



```
%% Record input audio
recObj = audiorecorder(Fs,nBits,NumChannels); % creating record object
fprintf("Get Ready to record!\n");
pause(1);
fprintf("Click any button to start recording input audio\n");
waitforbuttonpress;
fprintf("Recording ...\n");
recordblocking(recObj,recTime);
fprintf("End recording\n");
pause(0.5);
fprintf("Playing back input audio recording\n");
play(recObj);
pause(1);
inputAudio = getaudiodata(recObj); % save input audio

% plot input audio
figure();
plot(linspace(0,recTime,recTime*Fs),inputAudio,'k');
title("Recording of Input Audio");
```

Figure 6:

Plotting recorded audio:

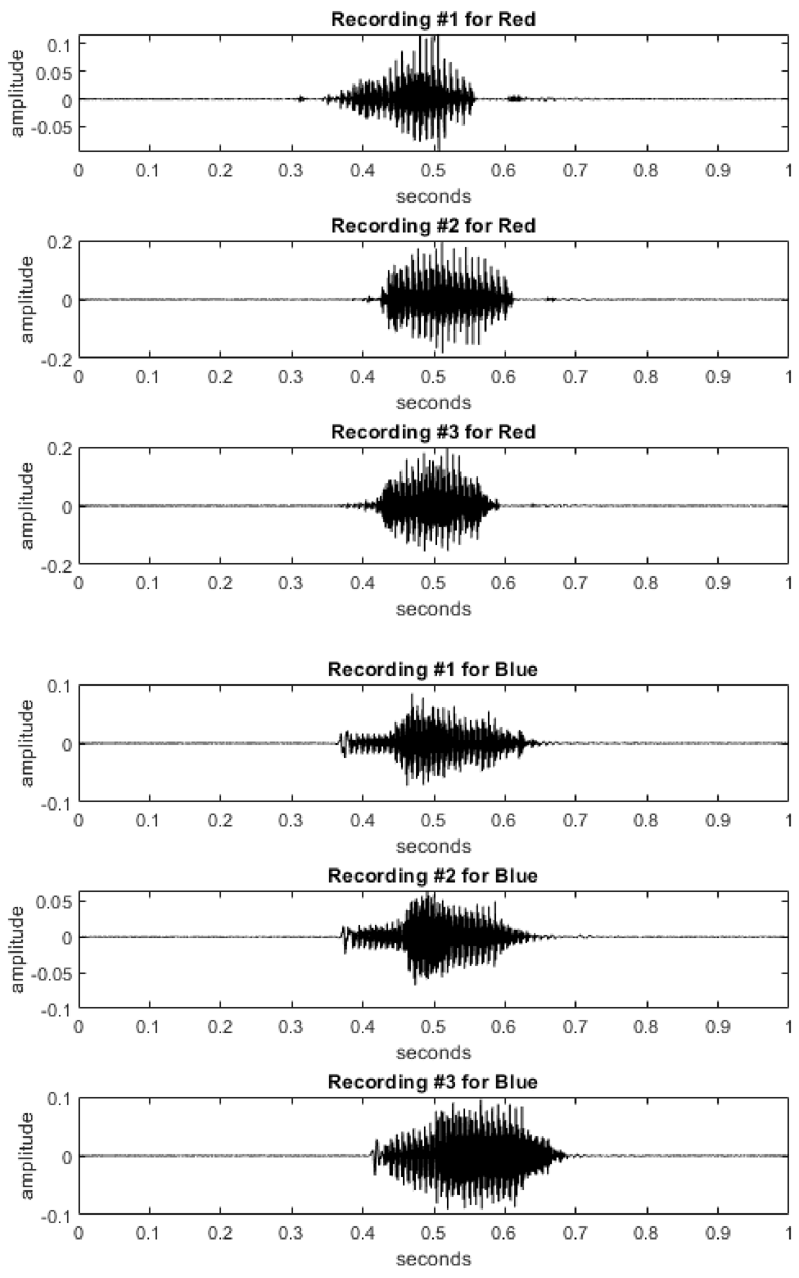


Figure 7: "Red" and "Blue" recorded audios

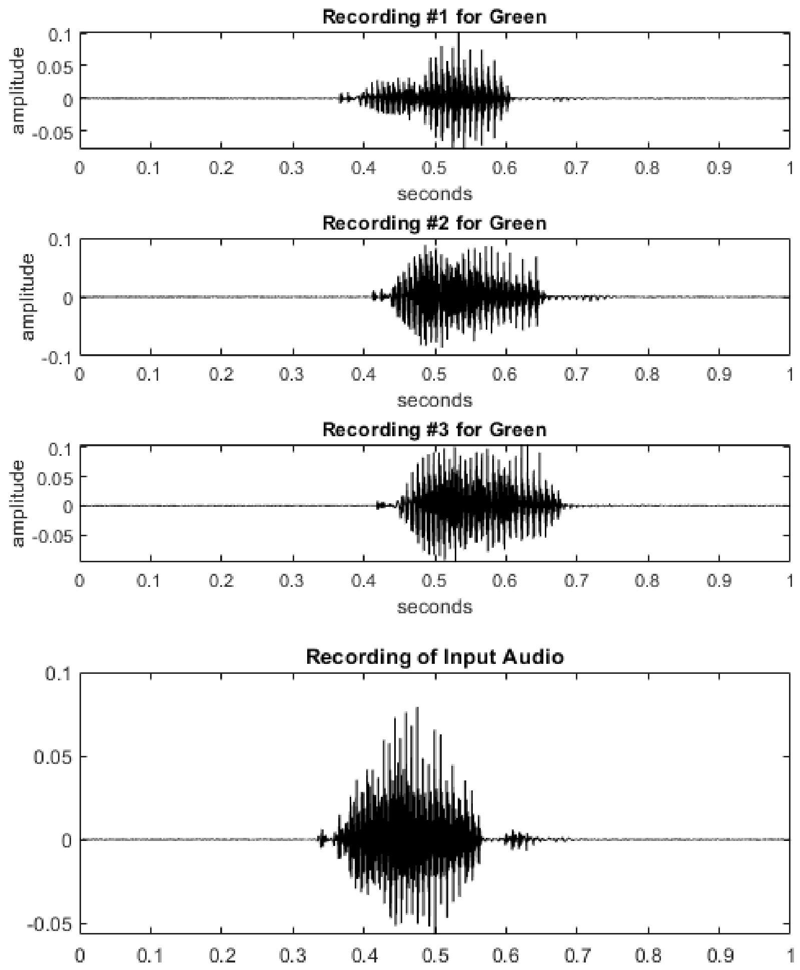


Figure 8: "Green" and input recorded audios

Now that we have all our "training" audio and input audio saved it's time to calculate our MFC coefficients.

2.2 Feature Extraction

Using the MFCC method we talked about earlier, we can use the built-in mfcc( ) MATLAB function. This function has input parameters for the audio amplitude values, and sampling frequency, and returns the mel frequency cepstral coefficients. The algorithm from MATLAB's documentation for this function is:

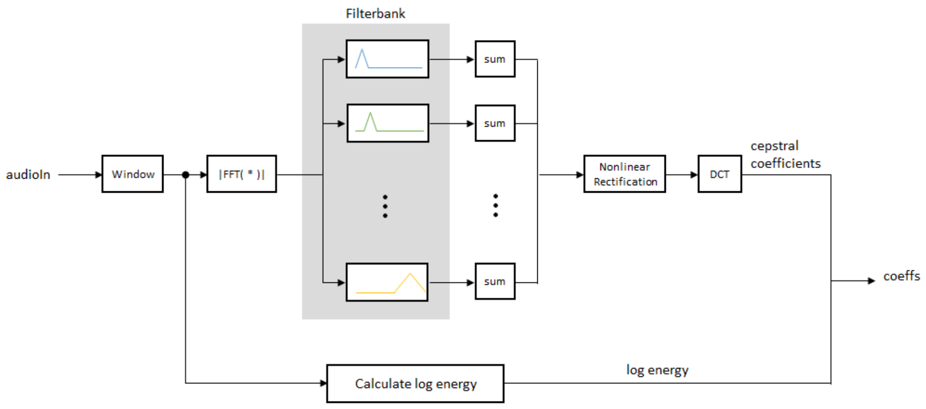
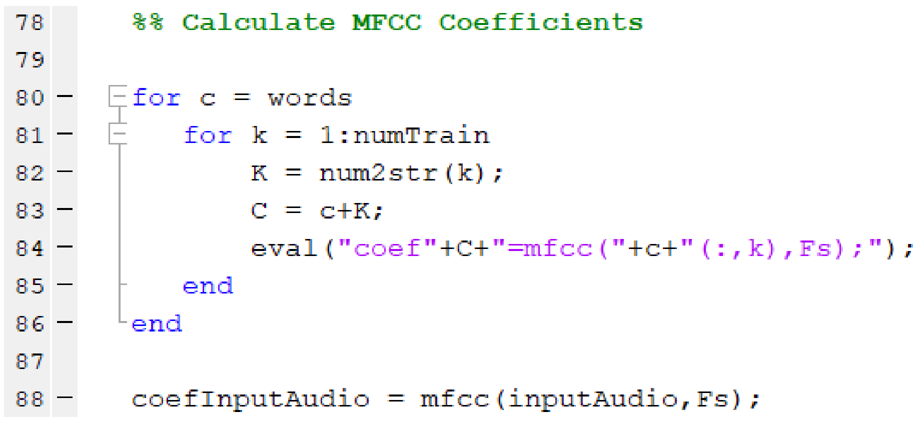


Figure 9:

This is implemented in MATLAB as:



```
%% Calculate MFCC Coefficients

for c = words
    for k = 1:numTrain
        K = num2str(k);
        C = c+K;
        eval("coef"+C+"=mfcc("+c+"(:,k),Fs);");
    end
end

coefInputAudio = mfcc(inputAudio,Fs);
```

Figure 10:

2.3 Feature Recognition

Now that we've calculated the MFC coefficients for each audio sequence we can compare the inputted audio against the templates using dynamic time warping to see which template has the lowest distance when compared to our inputted audio.

For this, MATLAB has a built-in dtw( ) function that has input parameters of an input vector "x" and an input vector "y". The order of inputting "x" and "y" does not matter. This function outputs: the total calculated euclidean distance of "x" and "y", the warping path for "x", and the warping path for "y".

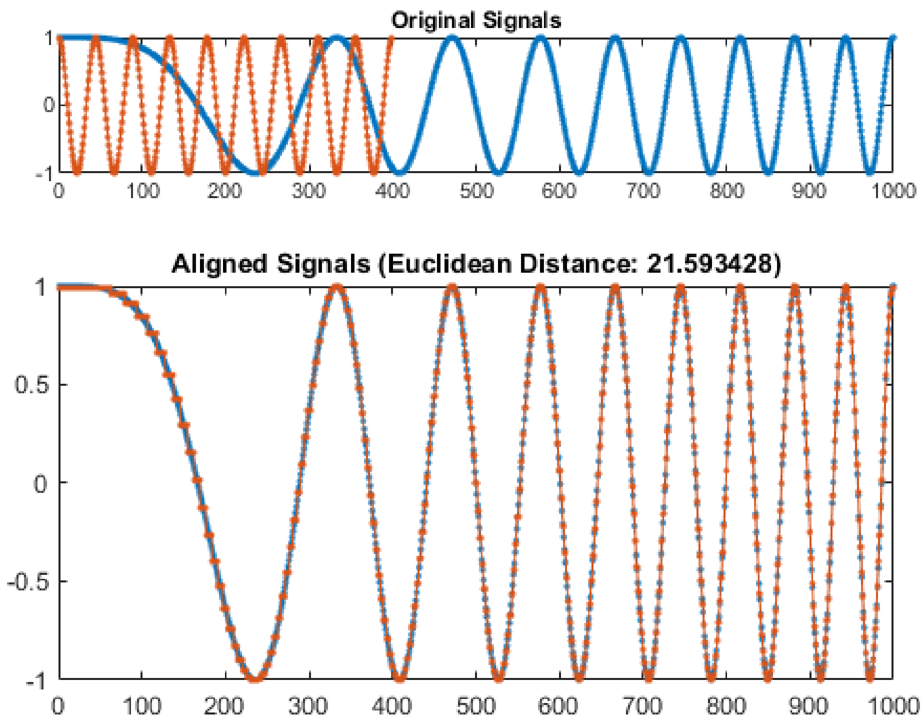
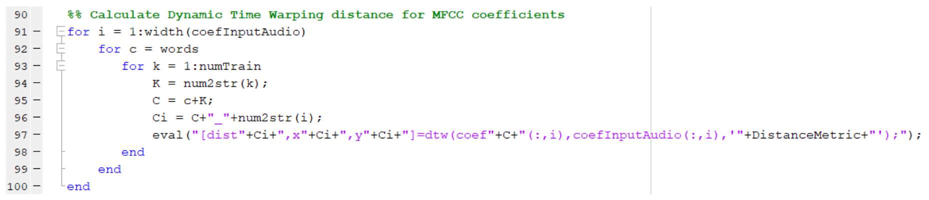


Figure 11: Example of how dtw( ) ”maps” one sequence to another

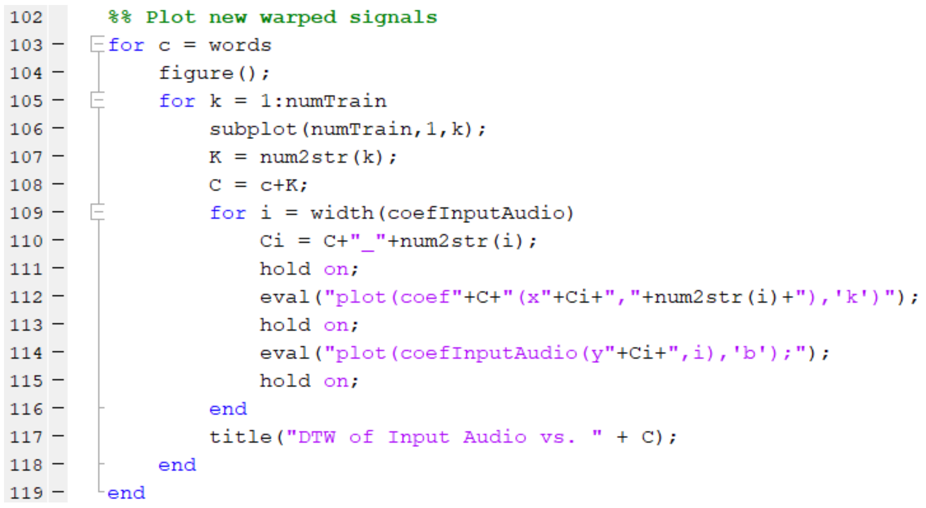
Implementing this in MATLAB:



```
%% Calculate Dynamic Time Warping distance for MFCC coefficients
for i = 1:width(coefInputAudio)
    for c = words
       for k = 1:numTrain
           K = num2str(k);
           C = c+K;
           Ci = C+"_"+num2str(i);
           eval("[dist"+Ci+",x"+Ci+",y"+Ci+"]=dtw(coef"+C+"(:,i),coefInputAudio(:,i),'"+DistanceMetric+"');");
       end
    end
end
```

Figure 12:

Then plotting results gives us:



```
%% Plot new warped signals
for c = words
    figure();
    for k = 1:numTrain
        subplot(numTrain,1,k);
        K = num2str(k);
        C = c+K;
        for i = width(coefInputAudio)
            Ci = C+"_"+num2str(i);
            hold on;
            eval("plot(coef"+C+"(x"+Ci+","+num2str(i)+"),'k')");
            hold on;
            eval("plot(coefInputAudio(y"+Ci+",i),'b');");
            hold on;
        end
        title("DTW of Input Audio vs. " + C);
    end
end
```

Figure 13:

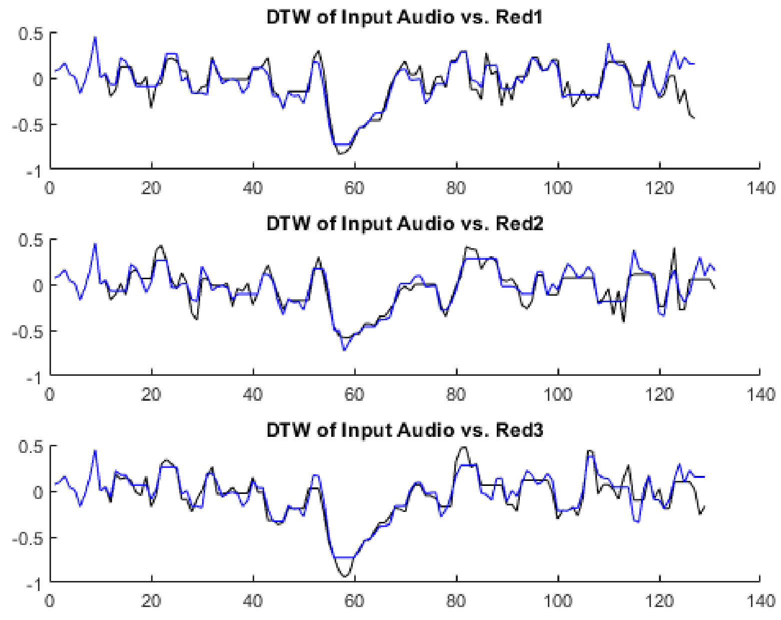


Figure 14: DTW of Input vs. Red MFC Coefficients

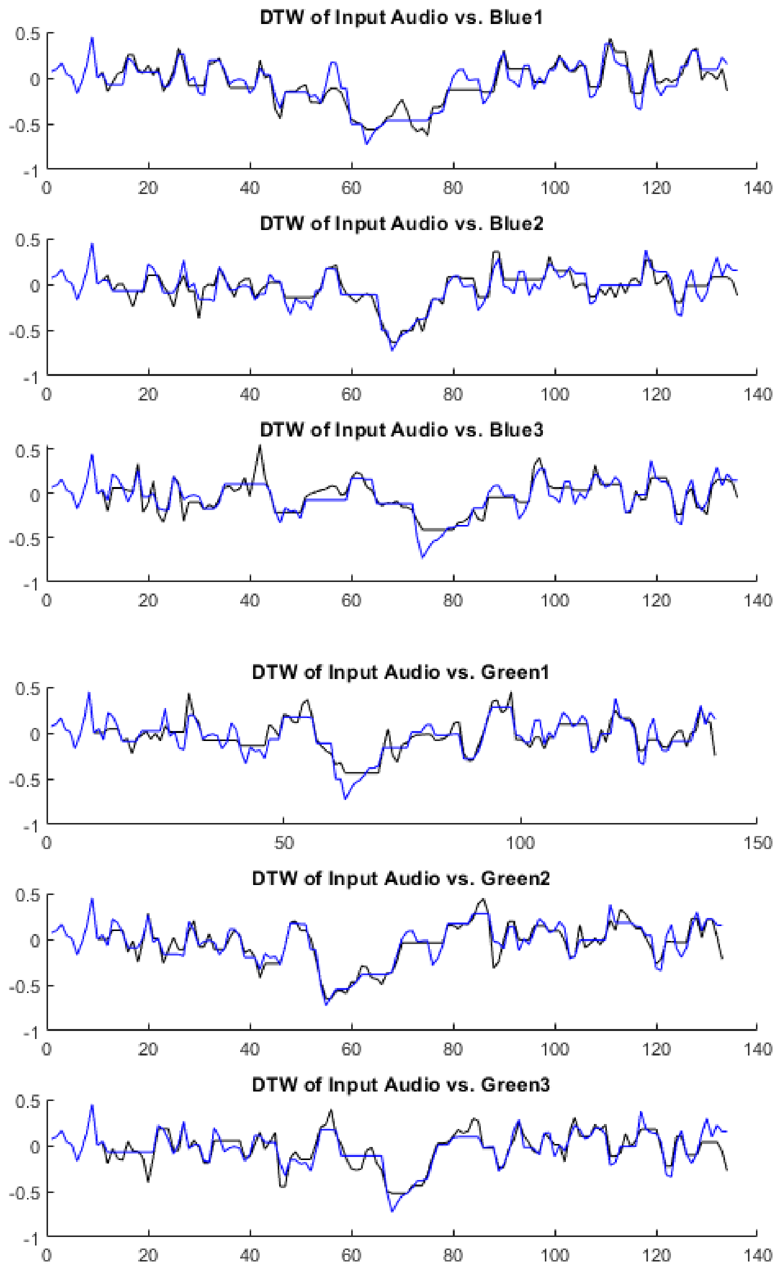
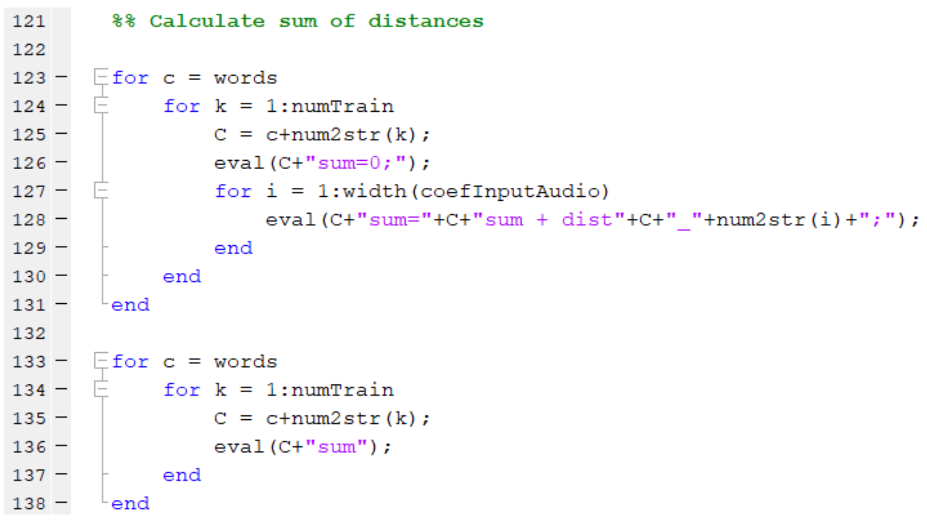


Figure 15: DTW of Input vs. Blue/Green MFC Coefficients

Then, summing all the distances from the Red, Blue, and Green coefficient matrices gives us:



```
%% Calculate sum of distances

for c = words
    for k = 1:numTrain
        C = c+num2str(k);
        eval(C+"sum=0;");
        for i = 1:width(coefInputAudio)
            eval(C+"sum="+C+"sum + dist"+C+"_"+num2str(i)+";");
        end
    end
end

for c = words
    for k = 1:numTrain
        C = c+num2str(k);
        eval(C+"sum");
    end
end
```

Figure 16:

Summing each gives us:

Color	Set	Distance
Red	1	95.78
	2	180.09
	3	139.43
Blue	1	179.83
	2	172.95
	3	292.08
Green	1	208.80
	2	241.63
	3	201.73

From this we can see that training sets from the color red have the lowest two distances. This is a good thing because in this case I did say ”Red” for my input recording.

A MATLAB

```
clear variables; close all;

% *************************
% *** EDITABLE VARIABLES ***
% *************************
words = ["Red","Green","Blue"]; % What words you want to
    record
```

```
numTrain = 3; % number of training data for each word
recTime = 1; % record time in seconds
Fs = 8e3; % recording sampling frequency
nBits = 16; % number of bits to record audio in
NumChannels = 1; % number of channels to record from
DistanceMetric = "squared"; % euclidean, absolute,
    squared as options
% ***********
% *** END ***
% ***********

%% Record and save audio
recObj = audiorecorder(Fs,nBits,NumChannels); % creating
    record object

fprintf("Get Ready to record!\n");
pause(1);

for c = words
    eval(c+"=[];");
    for k = 1:numTrain
        fprintf("Click any button to start recording #"+
            num2str(k)+" for "+c+"\n");
        waitforbuttonpress;
        fprintf("Recording ...\n");
        recordblocking(recObj,recTime);
        fprintf("End recording #"+num2str(k)+" for "+c+"\n
            ");
        pause(0.5);
        fprintf("Playing back recording #"+num2str(k)+"
            for "+c+"\n");
        play(recObj);
        pause(1);
        if k == 1; eval(c+"=getaudiodata(recObj);");
        else; eval("temp=getaudiodata(recObj);");
              eval(c+"=["+c+",temp];");
        end
    end
end

clear temp;

%% Plot saved audio
close all;

for c = words
```

```
    figure();
    for k = 1:numTrain
        subplot(numTrain,1,k);
        eval("plot(linspace(0,"+num2str(recTime)+","+
            num2str(recTime*Fs)+"),"+c+"(:,k),'k');");
        title("Recording #"+num2str(k)+" for "+c);
        xlabel("seconds"); ylabel("amplitude");
    end
end

%% Record input audio
recObj = audiorecorder(Fs,nBits,NumChannels); % creating
    record object
fprintf("Get Ready to record!\n");
pause(1);
fprintf("Click any button to start recording input audio\
    n");
waitforbuttonpress;
fprintf("Recording ...\n");
recordblocking(recObj,recTime);
fprintf("End recording\n");
pause(0.5);
fprintf("Playing back input audio recording\n");
play(recObj);
pause(1);
inputAudio = getaudiodata(recObj); % save input audio

% plot input audio
figure();
plot(linspace(0,recTime,recTime*Fs),inputAudio,'k');
title("Recording of Input Audio");


%% Calculate MFCC Coefficients

for c = words
    for k = 1:numTrain
        K = num2str(k);
        C = c+K;
        eval("coef"+C+"=mfcc("+c+"(:,k),Fs);");
    end
end

coefInputAudio = mfcc(inputAudio,Fs);

%% Calculate Dynamic Time Warping distance for MFCC
```

```
    coefficients
for i = 1:width(coefInputAudio)
    for c = words
        for k = 1:numTrain
            K = num2str(k);
            C = c+K;
            Ci = C+"_"+num2str(i);
            eval("[dist"+Ci+",x"+Ci+",y"+Ci+"]=dtw(coef"+C
                +"(:,i),coefInputAudio(:,i),'"+
                DistanceMetric+"');");
        end
    end
end

%% Plot new warped signals
for c = words
    figure();
    for k = 1:numTrain
        subplot(numTrain,1,k);
        K = num2str(k);
        C = c+K;
        for i = width(coefInputAudio)
            Ci = C+"_"+num2str(i);
            hold on;
            eval("plot(coef"+C+"(x"+Ci+","+num2str(i)+"),
                'k')");
            hold on;
            eval("plot(coefInputAudio(y"+Ci+",i),'b');");
            hold on;
        end
        title("DTW of Input Audio vs. " + C);
    end
end

%% Calculate sum of distances

for c = words
    for k = 1:numTrain
        C = c+num2str(k);
        eval(C+"sum=0;");
        for i = 1:width(coefInputAudio)
            eval(C+"sum="+C+"sum + dist"+C+"_"+num2str(i)
                +";");
        end
    end
end
```

```

for c = words
    for k = 1:numTrain
        C = c+num2str(k);
        eval(C+"sum");
    end
end
```

References

1. Bhadragiri, J.M., and Ramesh, B.N., *Speech Recognition using MFCC and DTW*, VIT University, Vellore, India, 2014

2. Kare, C.B., and Navale, V.S., *Speech recognition by Dynamic Time Warping*, IOSR Journal of Electronics and Communication Engineering, Pune, 2015

3. Kesarkar, M.P., *Feature Extraction for Speech Recognition*, IIT Bombay, Mumbai, India, 2003

4. Narang, S., and Gupta, D., *Speech Feature Extraction Techniques: A Review*, Amity University, Noida, India, 2015

5. Rao, K.S., and Manjunath K.E., *Speech Recognition Using Articulatory and Excitation Source Features*, 2017