Male Female Voice Recognition using MATLAB Software

The Speech signal processing has numerous applications in almost all technical fields. Gender identification is important in speech processing. This project describes a comparative analysis of speech signals in order to produce automatic gender classification. Gender classification by speech signal is a technique that analyses various features of a voice sample to determine the gender of the speaker.

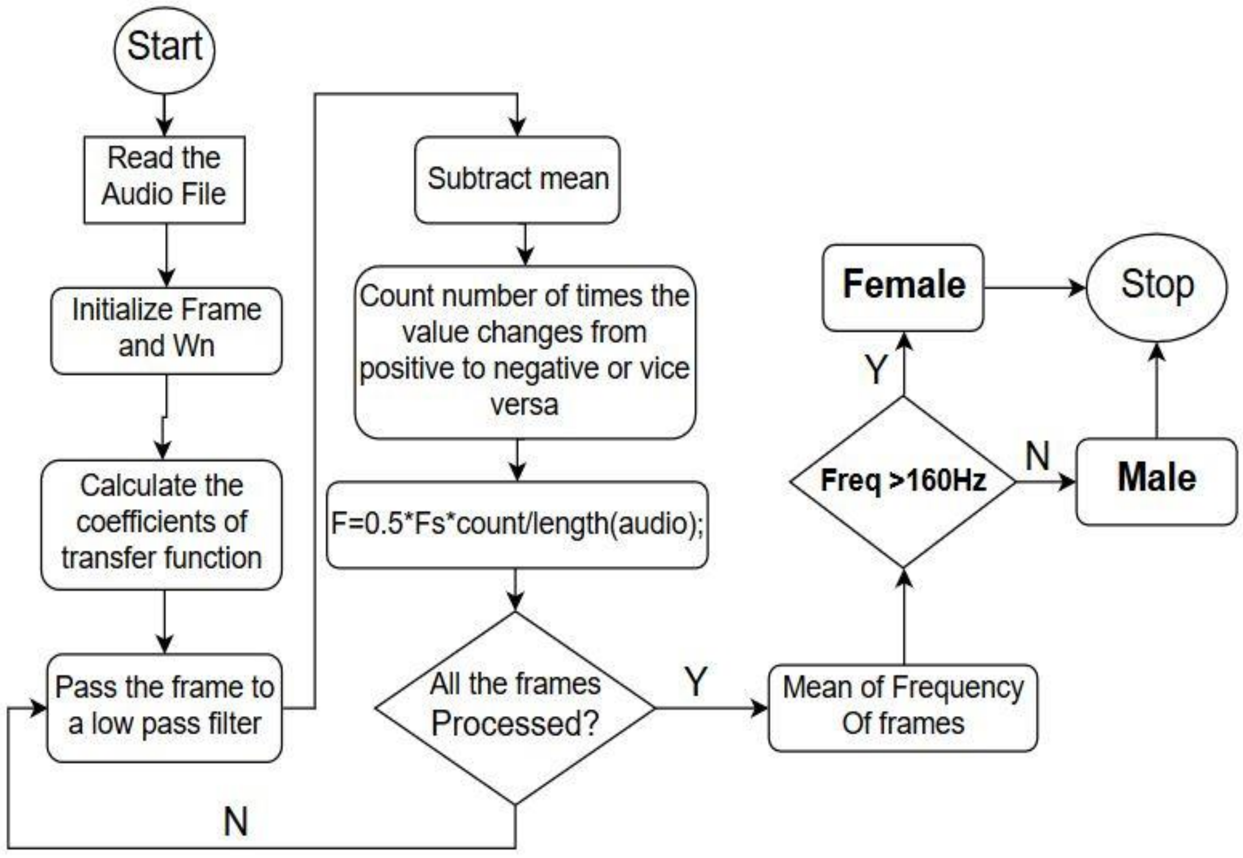
Method 1: Using the built-in pitch function [f0,inx] = pitch(audioIn,fs) It returns the fundamental frequency, f0, of any input audio file with sampling frequency fs right away. As a result, we can directly compare it to 165 Hz and determine whether the voice is male or female.

Method 2: Simulink Implementation Using the From Multimedia File block the sample audio is taken as input with 3500 samples per audio channel.

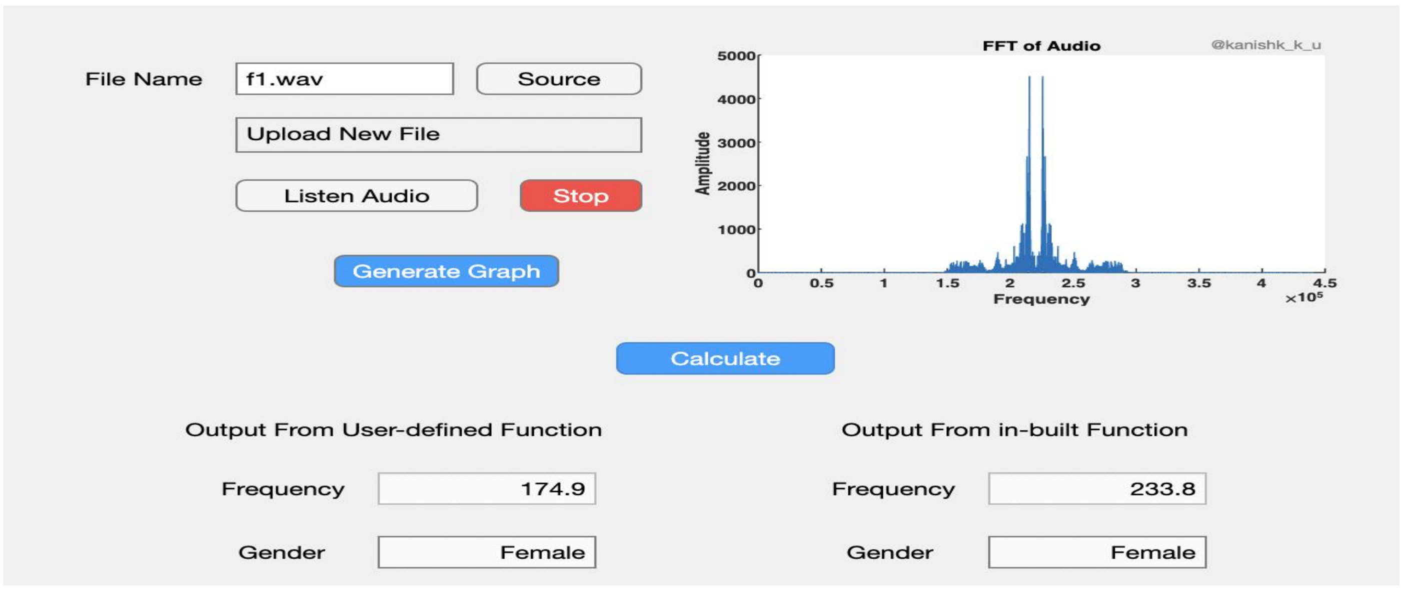
Proposed Methodology

Firstly, male and female voice samples are recorded in a file and saved in a recorded file. The feature (energy) is then extracted from the voice sample and designated as a known value. The unknown voice sample is then analysed and the feature extracted. Unknown value refers to the extracted feature. The unknown and known values are compared. If it matches, we can conclude whether the speaker is male or female. The unknown and known values are compared. Extraction of Characteristics As a feature, energy is extracted by estimating the power spectrum. A file contains samples of both male and female voices. The voice samples are then fed into the recognition system. The voice samples are then subjected to FFT. The power spectrum is then estimated using the FFT signal. Following that, the energy is extracted from the power spectrum. The threshold energy is derived from this. Energy is extracted from the unknown voice sample using the same method and compared to the estimated frequency.

Flowchart


Start
Read the Audio File
Initialize Frame and Wn
Calculate the coefficients of transfer function
Pass the frame to a low pass filter
Subtract mean
Count number of times the value changes from positive to negative or vice versa
F=0.5*Fs*count/length(audio);
All the frames Processed?
Y
N
Mean of Frequency Of frames
Freq >160Hz
Y
N
Female
Male
Stop

Using GUI based Matlab App


File Name f1.wav

Output From User-defined Function

Frequency 174.9

Gender Female

Output From in-built Function

Frequency 233.8

Gender Female

Source code analysis

Myfilter.m:

```
function filtered = myfilter(b, a, raw)

filtered = zeros(size(raw));

   for n = 3:size(raw,1)

      filtered(n,1) = b(1)* raw(n,1)   + b(2)* raw(n-1,1) + b(3)* raw(n-2,1) ...
                 - a(1)*filtered(n,1)   - a(2)*filtered(n-1,1) - a(3)*filtered(n-2,1) ;

   end
```

Here's a breakdown of how the code works:

1. **Input Parameters:**
   - **b**: This is the feedforward coefficient vector of the filter. It contains the coefficients that multiply the input values.
   - **a**: This is the feedback coefficient vector of the filter. It contains the coefficients that multiply the filtered output values from previous time steps.
   - **raw**: This is the input signal that you want to filter.
2. **Output:**
   - **filtered**: This is the output of the filtering process. It's the result of applying the filter to the **raw** input signal.
3. **Filtering Loop:** The core of the code is a loop that iterates through the rows of the **raw** signal starting from the third row (index 3). This makes sense since the filtering operation depends on previous samples, and the first two rows do not have enough previous samples to compute the filter.
4. **Filter Equation:** The filter operation is implemented using the difference equation of an IIR filter. The equation calculates the filtered output at each time step (**n**) based on the current and previous input samples and the current and previous filtered output samples.
5. Here, **b** coefficients are multiplied with the current and two previous input samples, and **a** coefficients are multiplied with the current and two previous filtered output samples.

The result is assigned to **filtered(n, 1)**.
Overall, this code is implementing a third-order IIR filter using the given coefficients **b** and **a** to process an input signal **raw**. It's a basic example of digital signal processing where filtering is used to modify the frequency content of a signal. Keep in mind that the performance and characteristics of the filter are determined by the values of the coefficients **b** and **a**.

Mybutter.m:

```
V  = tan(W * 1.5707963267948966);
Sg = V ^ 2;
Sp = V * [-1-1i, -1+1i] / sqrt(2);

P = (1 + Sp) ./ (1 - Sp);
G = real(Sg / prod(1 - Sp));

B = G * [1, 2, 1];
A = real(poly(P));
end
```

1. **Calculation of V:**
In this line, **W** seems to be a variable that is not shown in the provided code snippet. **V** is calculated by taking the tangent of **W** multiplied by π/2. This operation is effectively scaling and transforming the value of **W**.
2. **Calculation of Sg (Squared Magnitude of V):**
This line calculates the squared magnitude of **V**, which is simply the square of the value calculated in the previous step. This might be used as a measure of signal strength or energy.

3. **Calculation of Sp (Spectral Parameter):**
Here, **Sp** is calculated by multiplying **V** with a complex vector **[-1-1i, -1+1i]** and then dividing by the square root of 2. This is likely used in some filter or transfer function calculations.

4. **Calculation of P (Transfer Function Denominator Polynomial):**
P is calculated by element-wise division of two complex vectors **(1 + Sp)** and **(1 - Sp)**. This is related to the transfer function's denominator polynomial coefficients.

5. **Calculation of G (Transfer Function Constant):**
G is calculated as the real part of the division between **Sg** (the squared magnitude of **V**) and the product of **(1 - Sp)** for both elements in the **Sp** vector. This could be a scaling factor or a constant in the transfer function.

6. **Calculation of B (Numerator Polynomial Coefficients) and A (Denominator Polynomial Coefficients):**
B is calculated by multiplying the constant **G** with the vector **[1, 2, 1]**. This is likely related to the numerator coefficients of a transfer function.
A is calculated by converting the complex **P** vector into its polynomial representation and then taking the real part. This is likely related to the denominator coefficients of a transfer function.

Gender_determine.m:

```
clc;clear all;close all;

[y ,Fs]=audioread('Voice/amina-1.wav'); % Read the audio sample
frame=3500; % Set the frame rate
[b0,a0]=mybutter(350/(Fs/2)); % Get teh coefficient of the filter matrix
%% Identify the frequency of each frame
for i=1:length(y)/frame
    x=y(1+(i-1)*frame:i*frame);
    xin = abs(x);
    xin=myfilter(b0,a0,xin);
    xin = xin-mean(xin);
    x_out(1+(i-1)*frame:i*frame,1)=xin;
    x2=zeros(length(xin),1);
    x2(1:length(x)-1)=xin(2:length(x));
    zc=length(find((xin>0 & x2<0) | (xin<0 & x2>0)));
    F0(i)=0.5*Fs*zc/length(x);

end
Fx=mean(F0); % Take mean of all the frequency for each frame

%% Display the output frequency
fprintf('Estimated frequency is %3.2f Hz.\n',Fx);
fprintf('Estimated frequency by in-built function is %3.2f
Hz.\n',mean(freq(y))); % Use the function freq to find the frequency

%% Display the final Gender
if Fx>165   % set the threshold
    fprintf('Female Voice\n');
else
    fprintf('Male Voice\n');
end
```

Conclusion:
This process of translating speech in systems is known as gender recognition using voice. It was created to allow a person to authenticate or verify the identity of a speaker as part of a security measure. In this project, the speaker is identified by using energy estimation as a threshold value. This calculated energy is then compared to the threshold energy. If the energy is greater than the threshold, the male produces the voice sample. If it is less than the threshold, the female produces the voice sample.