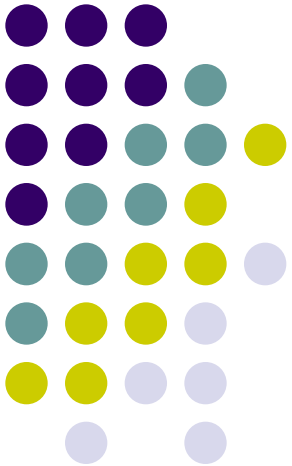
Voice2Text

CS 293 Final Project Demo

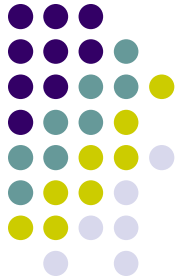
29th November, 2012

Tanmay Randhavane, 110050010

Nishit Bhandari, 110050026



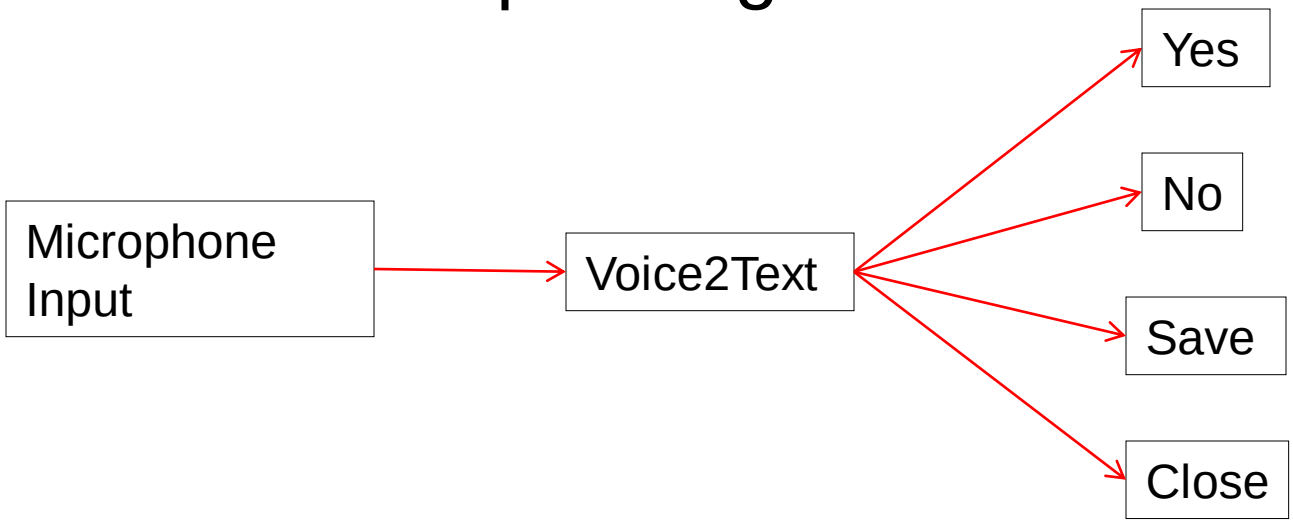
Outline <for a total of 30 mins>



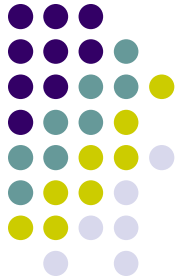
- Aim of project (1 mins)
- Demo (5 mins)
- Teamwork Details (0.5 min)
- Design Details –Algorithm (5 mins)
- Design Details – Implementation (8 mins)
- Viva (9 mins)
- Transition time to next team (2 mins)

Aim of the project

- To create a program that takes words from {yes, no, save, close} as voice input and outputs the corresponding word.



Overview



- Our work was mainly divided in following steps:

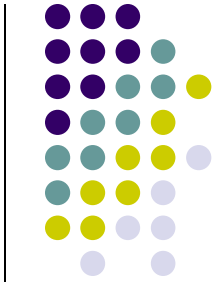
1. Take voice input from the user and convert it into *.wav* file.
2. The sound wave thus obtained is in the time domain, convert it in frequency domain using *fast fourier transform*.
3. Find the characteristics of sound wave like peaks of frequency, energy and frequency distributions, etc.
4. Compare these characteristics with the known data and decide to which word they match closest. This is our final output.

Teamwork Details

- Division of work
  - Tanmay Randhavane: Algorithm for peak finding, voice recording
  - Nishit Bhandari: data extraction from wave file, Cooley-Turkey algorithm for fast fourier transform, phoneme matching

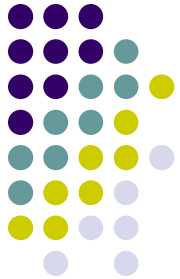
Overall Contribution by Team Member 1	Overall Contribution by Team Member 2
~50%	~50%

Design Details



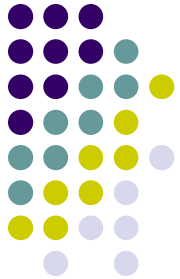
- Algorithm 1: *Author - Tanmay*
  - Find the peaks in a data.
- Structure
  - **Input** – Vector of size N
  - **Output** – List of indices of peaks in vector
  - **Time Complexity** – O(N) where N is the size of vector

Algorithm Details



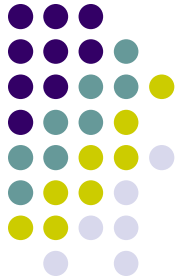
- Problem
  - Find the peaks in a sound vector block of size N.
- Solution
  - Loop through the entire sound vector to calculate standard deviation and maximum values. This is of O(N).
  - Create a list of elements which are above a certain threshold (O(N)).

Algorithm Details



- For each element in this list, check whether 8 out of 10 previous elements show increasing trend. If yes, then this is a peak else move forward. This step is of O(Number of elements in the list) = O(N)(in the worst case).
- In the above step, if there are peaks in the 10 previous elements, then they are not considered as peaks. This is done because if there are clustered peaks then any one of them is suffice.
- Thus, the overall function is of O(N).

Design Details



- Algorithm 2: *Author - Nishit*
  - **Cooley-Turkey** Algorithm, to implement fast fourier transform
- Structure
  - **Input** – blocks of sound data,
  - **Output** – Fast Fourier transformed data,
  - **Time Complexity –** *O(N logN)* where N is input data size

Background

- We required the frequencies of which the sound data was composed of so as to recognize which phonemes were present.

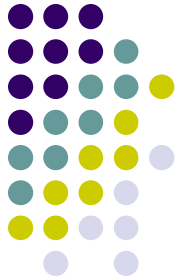
- We applied Fast Fourier Transformation on the sound data. After the transformation, data which was in time domain is converted to frequency domain and peaks occur at those frequencies which were present in the sound data.

Algorithm Design

- Problem
  - A sound vector A is given(size N).We need to make a complex vector C(size N) whose elements are-
  - $C[k] = \sum_{0}^{N-1} A[j] * e^{2\pi * j * k/N} \equiv FFT(A,N)$

- Solution
  - Doing this by two for loops would take $O(N^2)$ time. So, we found a way to implement it in O(N logN) by Cooley-Turkey Algorithm.

Algorithm Details



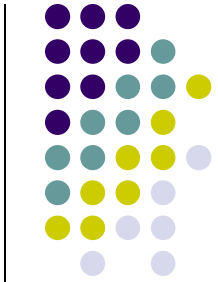
- Consider $N = 2^p$, for all even $j(=2t)$,
  $j/N = t/2^{p-1}$

- So $C[k] = \sum_0^{N-1} A[j] * e^{2\pi * j * k/N}$

- $C[k] = (\sum_{even} A[2t] * e^{2\pi * t * k/(N/2)}) +$

  $(e^{2\pi * k/(N/2)}) * (\sum_{odd} A[2t+1] * e^{2\pi * t * k/(N/2)})$

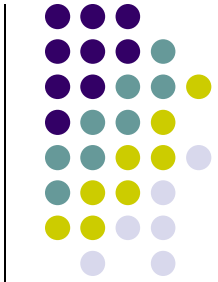
Algorithm Details



- FFT(A,N) = FFT(even terms of A, N/2) +
  $(e^{i\theta})$*FFT(odd terms of A, N/2)

- Hence recursively calling FFT on two vectors of size N/2 would result in order of FFT to be O(N logN).

Class Design



- Two classes created:

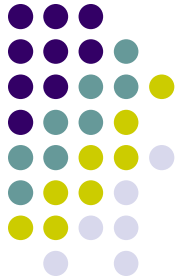
1. Complex class

2. Block class

Class Design

Class Name	Brief Description	Author
Block	Implements the blocks of wave file, contains functions to apply Fast Fourier transform and peak finding.	Tanmay
Complex	Implements basic functions about Complex numbers like add, subtract, multiply, polar transformation.	Nishit

Block Class



- Phonemes appear as blocks in the sound waveform.

- This class is used to implement blocks of the sound waveform.

- This class contains the functions to apply Fast Fourier Transform and to find peaks in that particular block.

- The data obtained from the blocks is used to detect the presence of phonemes in those blocks.

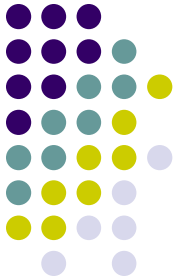
Block Class - Variables

Variable	Type	Details
vec	Vector of float	A block of sound wave is stored in this vector
compVector	Array of complex type of objects	Float data from vec is converted into complex object and saved into compVector to pass as an input in FFT function
blockSize	int	Stores the size of the block

Block Class - Variables

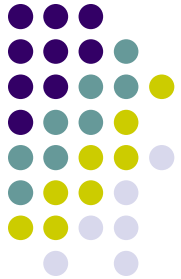
Variable	Type	Details
totalEnergy	float	Stores the total energy of the sound wave in the block(sum of squares of data values in vec)
zeroCrossings	int	Stores the number of times the vec crosses zero. This is directly proportional to the average frequency in the block
Peaks	List of int	Stores the values of the peaks in the sound block

Block Class – Important Functions



Functions	Details
FFT_SimpleH	Applies fast fourier transform on the wave block. This is a private helper function called through FFT_Simple.
convertToComplex	Takes a float sound vector and converts the data into an array of complex objects
mergeBlock	Merges two block objects into one
findPeaks	Finds the peaks in the sound data obtained after applying FFT.

Complex Class



• The output of Fast Fourier Transformation has real as well as imaginary parts.

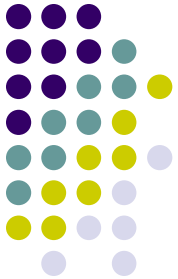
• So a class is required to implement basic operations like add, multiply, subtract and polar representations of complex numbers

• This class contains the functions to apply these operations.

Complex Class - Variables

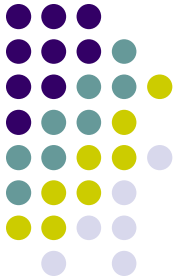
Variable	Type	Details
re	double	Stores the real part of the complex number
im	double	Stores the imaginary part of complex number

Complex Class - Functions



Functions	Details
assign	assign values to real and imaginary part
complex_from_polar	form the complex number using radius and angle
Mathematical operations	Functions to perform operations like addition, subtraction, multiplication, magnitude

Data Structures Used



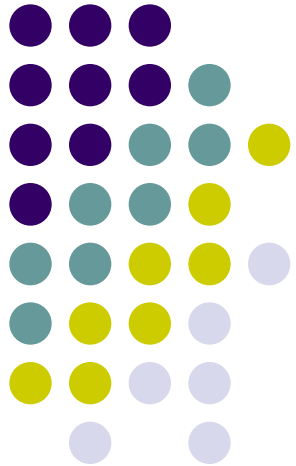
No specific data structures were used other than vectors and lists. These were used to store the sound data values, complex objects, block objects, etc.

Brief Conclusion

- Our program identifies words (specifically yes, no, save and close) spoken by various users with an accuracy of 83%(45 out of 54 test cases worked correctly).
- Some test cases(6 out of 54) did not work either due to bad pronunciation or some other unknown factors.

Note : Our inputs contained only male voices, so accuracy with female voices may be less.

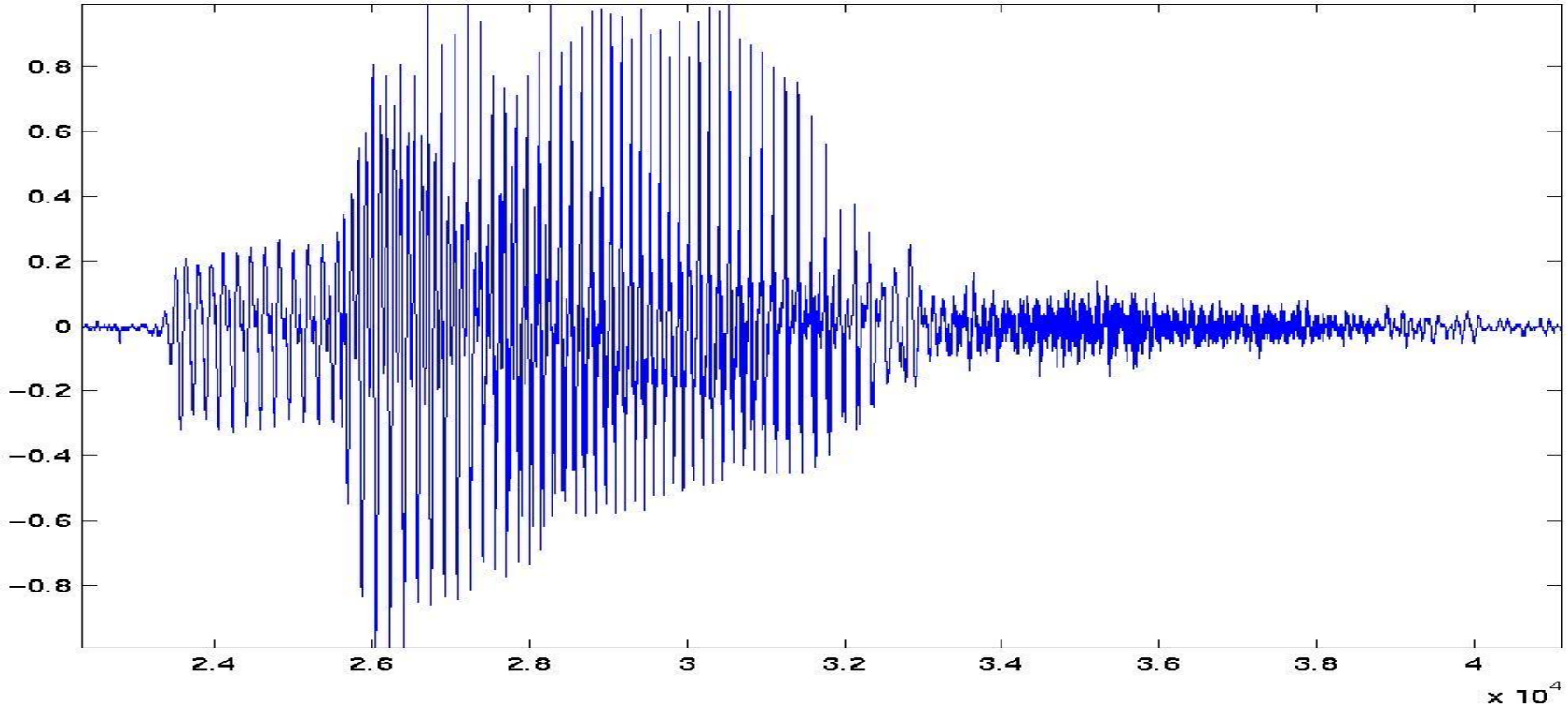
Thank You



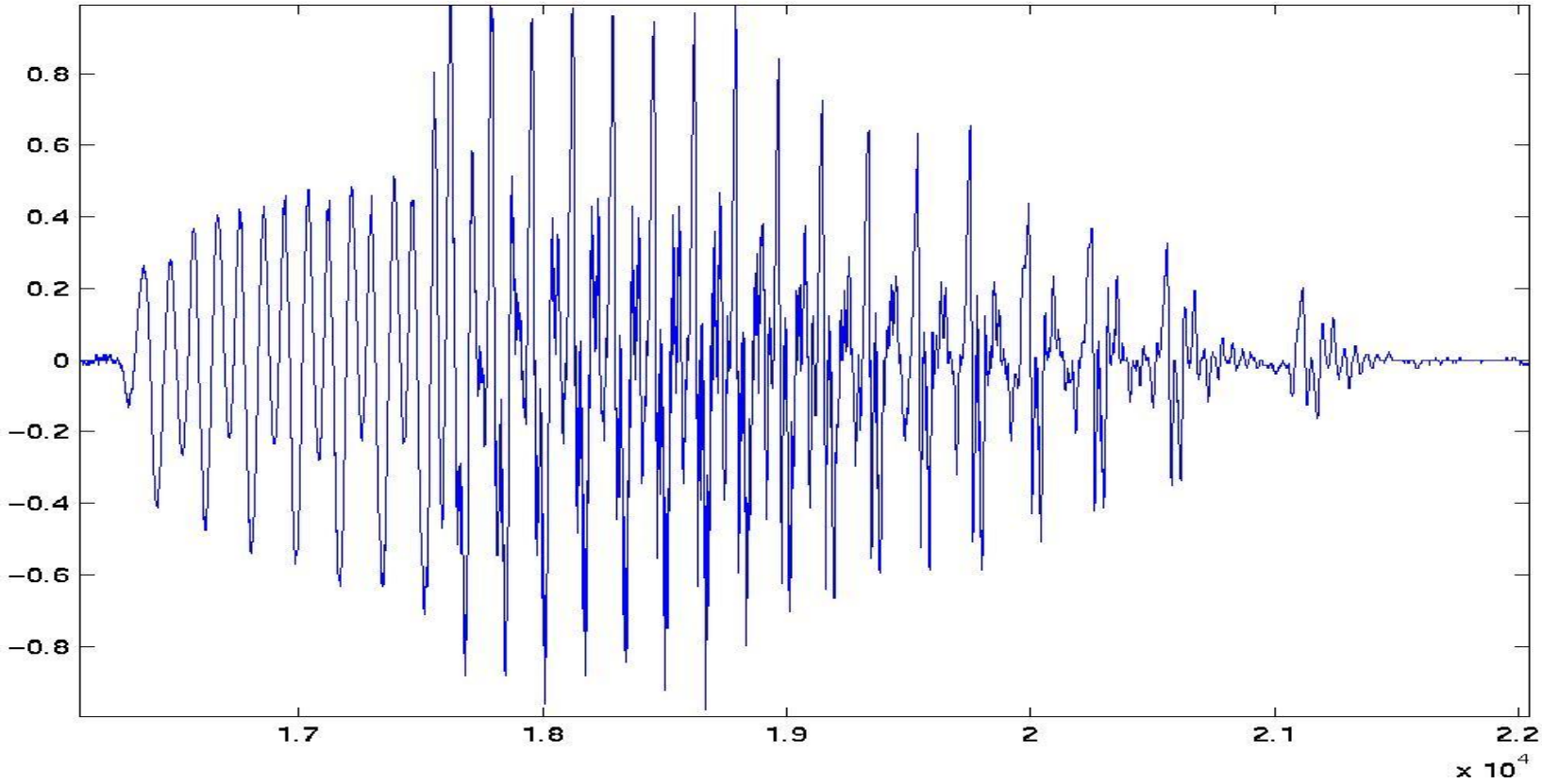
Back Up Slides

Source : http://www.cs.dartmouth.edu/~dwagn/aiproj/speech.html

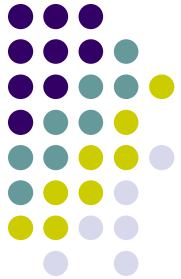
- *"Yes" Waveform :*



“No” Waveform:



Differentiating Features



- The beginning of each sound (wherein the 'y' is spoken) is characterized by a quite regular sound wave, while the middle section (wherein the 'e' is spoken) appears to contain a combination of multiple frequencies. The trailing section (wherein the 's' is spoken) is in fact quite distinctive, characterized by a very high frequency, low energy sound wave. This distinctiveness is what motivated the decision to use the 's' as a reliable factor differentiating 'yes' from 'no'.
- Notice here that the 'n' is characterized by a single frequency, while the 'o' is a combination of freqencies.

Other Features

- Nasals (n, m, ng), are often characterized by a single formant of low frequency, and if followed by a vowel, their formants tend to have a wide spectrum. The 'h' is characterized by a building unvoiced sound followed by a sudden sustained increase in energy at the formants of the vowel which follows. Unvoiced fricatives (th, s, sh, f), are characterized by a low energy, wide band, high frequency spectrum. Their voiced counterparts (dh, z, zh, v) have an additional formant in the low frequency spectrum.

Other Features

- Affricatives (j, ch) are often described as a plosive which turns into a fricative (d - zh and t - sh respectively). Glides, or semivowels (w, l, r, y) may be the most difficult to characterize, because they are highly situation dependent. They are followed by vowels, unless they appear at the end of a word, and behave much like a transition from another vowel into the vowel which follows it. In this project we noted how the 'y' transitions from it's characteristic frequency into the frequencies of the 'e' which follows it. There may be no clear distinction where one ends and the other begins.