

Third Year Project

Generated by Doxygen 1.9.4

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 Boundary Struct Reference	7
4.2 Bounds Class Reference	7
4.3 Cluster Struct Reference	8
4.3.1 Detailed Description	8
4.4 codes Struct Reference	8
4.5 Dataset Class Reference	8
4.6 Feature_Set Struct Reference	9
4.7 Guess Struct Reference	9
4.7.1 Detailed Description	9
4.8 guess Struct Reference	10
4.9 KNN Class Reference	10
4.10 MyApp Class Reference	10
4.11 MyFrame Class Reference	10
4.12 MyGrid Class Reference	11
4.13 Ph_index Struct Reference	11
4.14 Phoneme Struct Reference	12
4.15 wav_file Struct Reference	12
4.15.1 Detailed Description	12
5 File Documentation	13
5.1 /home/bruh/Documents/GitHub/cleaned_and_commented/Clustering/cluster.c File Reference	13
5.1.1 Detailed Description	13
5.2 cluster.h	14
5.3 /home/bruh/Documents/GitHub/cleaned_and_commented/Clustering/knn.c File Reference	14
5.4 knn.h	14
5.5 knn.h	15
5.6 /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/dtw.c File Reference	15
5.6.1 Detailed Description	18
5.6.2 Function Documentation	18
5.6.2.1 average_mfccs()	18
5.6.2.2 create_mfcc()	19
5.6.2.3 dtw_frame()	19
5.6.2.4 export_device()	19

5.6.2.5 handle_argv()	20
5.6.2.6 init_dtw_matrix()	20
5.6.2.7 interrupt_handler()	20
5.6.2.8 length()	21
5.6.2.9 mean_size_mfccs()	21
5.6.3 Variable Documentation	21
5.6.3.1 glib_paa	21
5.7 dtw.h	22
5.8 dtw.h	24
5.9 /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/jenks.py File Reference	25
5.9.1 Detailed Description	25
5.10 /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/pca.py File Reference	26
5.10.1 Detailed Description	26
5.11 /home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/delta.c File Reference	26
5.11.1 Detailed Description	27
5.11.2 Function Documentation	27
5.11.2.1 delta()	27
5.11.2.2 delta_delta()	28
5.11.2.3 normalise_delta()	28
5.11.2.4 normalise_delta_delta()	29
5.11.2.5 update_delta_deltas_norm()	29
5.11.2.6 update_deltas_norm()	29
5.12 delta.h	30
5.13 /home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/fft.c File Reference	30
5.13.1 Detailed Description	30
5.13.2 Function Documentation	30
5.13.2.1 dct()	31
5.13.2.2 fft()	32
5.13.2.3 fft_chunks()	32
5.14 fft.h	33
5.15 /home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/hanning.c File Reference	33
5.15.1 Detailed Description	34
5.15.2 Function Documentation	34
5.15.2.1 hanning()	34
5.15.2.2 hanning_chunks()	34
5.15.2.3 hanning_chunks_no_overlap()	35
5.15.2.4 hanning_window()	35
5.16 hanning.h	36
5.17 /home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/mel.c File Reference	36
5.17.1 Detailed Description	37

5.17.2 Function Documentation	37
5.17.2.1 mel()	37
5.18 mel.h	37
5.19 /home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/mfcc.c File Reference	38
5.19.1 Detailed Description	38
5.19.2 Function Documentation	38
5.19.2.1 kurtosis()	38
5.19.2.2 log_energy()	39
5.19.2.3 log_entropy()	39
5.19.2.4 mfcc()	40
5.19.2.5 st_energy()	40
5.19.3 Variable Documentation	41
5.19.3.1 max_mfcc	41
5.20 mfcc.h	41
5.21 mfcc.h	41
5.22 /home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/paa.c File Reference	41
5.22.1 Detailed Description	42
5.23 paa.h	42
5.24 /home/bruh/Documents/GitHub/cleaned_and_commented/Misc/realloc.c File Reference	42
5.24.1 Detailed Description	43
5.24.2 Function Documentation	43
5.24.2.1 s_realloc()	43
5.25 realloc.h	43
5.26 mfccs.h	44
5.27 includes.h	44
5.28 mfcc_vars.h	44
5.29 rt.h	44
5.30 /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/bounds.c File Reference	45
5.30.1 Detailed Description	45
5.30.2 Function Documentation	45
5.30.2.1 get_entropy()	45
5.30.2.2 next_boundary()	46
5.30.2.3 shift_and_reduce()	46
5.31 bounds.h	47
5.32 bounds.h	47
5.33 /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/cross_rate.c File Reference	47
5.33.1 Detailed Description	48
5.33.2 Function Documentation	48
5.33.2.1 cross_rate()	48
5.33.2.2 favg_cross_rate()	49
5.33.2.3 is_positive()	49
5.33.2.4 stavg_cross_rate()	49

5.34 cross_rate.h	50
5.35 d.h	50
5.36 /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ste.c File Reference	50
5.36.1 Detailed Description	51
5.36.2 Function Documentation	51
5.36.2.1 f_short_time_energy()	51
5.36.2.2 short_time_energy()	51
5.37 ste.h	52
5.38 /home/bruh/Documents/GitHub/cleaned_and_commented/Testing/test.c File Reference	52
5.38.1 Detailed Description	54
5.38.2 Function Documentation	54
5.38.2.1 bounds_output()	54
5.38.2.2 data_size()	54
5.38.2.3 export_results()	55
5.38.2.4 get_codes()	55
5.38.2.5 matrix_output()	55
5.38.2.6 merge_silences()	56
5.38.2.7 minimum_edit_distance()	56
5.38.2.8 reduce_data()	56
5.38.2.9 reset()	56
5.38.2.10 test()	57
5.38.2.11 test_output()	57
5.38.2.12 test_phoneme()	57
5.38.2.13 test_phoneme_utterance()	57
5.38.2.14 trim_silence()	59
5.38.2.15 utterance_test()	59
5.39 test.h	60
5.40 test.h	60
5.41 /home/bruh/Documents/GitHub/cleaned_and_commented/Training/train.c File Reference	60
5.41.1 Detailed Description	62
5.41.2 Function Documentation	62
5.41.2.1 train()	62
5.41.3 Variable Documentation	62
5.41.3.1 max_sil	62
5.41.3.2 max_sil_dB	62
5.41.3.3 max_silflt	62
5.41.3.4 max_sil_mean	63
5.41.3.5 max_sil_ste	63
5.41.3.6 max_sil_zc	63
5.41.3.7 OBSTR	63
5.41.3.8 p_codes	63
5.41.3.9 p_group	63

5.41.3.10 ph_zc_max	64
5.42 train.h	64
Index	65

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Boundary	7
Cluster	8
codes	8
Feature_Set	9
Guess	9
guess	10
Ph_index	11
Phoneme	12
wav_file	12
App	
MyApp	10
Enum	
Bounds	7
Dataset	8
KNN	10
Screen	
MyFrame	10
MyGrid	11

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Boundary	7
Bounds	7
Cluster	
Cluster structure :: used to hold the generated 1D Jenks Natural Breaks cluster @centroids can be one or more centroids, these are generated from the @clust function @values these are the values from the original data after it has been sorted and duplicates have been removed the i'th array is associated with the i'th centroid in @centroids @sizes is the length of each @values array with the corresponding index @count is the number of centroids in @centroids @gfv is the Goodness of Variance Fit and is a descriptor of how well the centroids describe the dataset . . .	
codes	8
Dataset	8
Feature_Set	9
Guess	
Guess struct is used to store a guess in KNN for sorting and deciding the result @guess index guessed, can be the phoneme, group or voice @diff the difference result used for sorting @ref↔_indx the MFCC index used to produce the result - used for storing the amount of correct and incorrect guesses for the MFCC	
guess	10
KNN	10
MyApp	10
MyFrame	10
MyGrid	11
Ph_index	11
Phoneme	12
wav_file	12

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

/home/bruh/Documents/GitHub/cleaned_and_commented/Clustering/ cluster.c	
A naive Jenks clustering class -	13
/home/bruh/Documents/GitHub/cleaned_and_commented/Clustering/ cluster.h	14
/home/bruh/Documents/GitHub/cleaned_and_commented/Clustering/ knn.c	
Contains the functions comparing phonemes using KNN	14
/home/bruh/Documents/GitHub/cleaned_and_commented/Clustering/ knn.h	14
/home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/ dtw.c	
The main control file of the program which also contains the DTW functions	15
/home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/ dtw.h	22
/home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/ jenks.py	
Creates clusters for each phoneme	25
/home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/ pca.py	
Exports all MFCCs in use for use with	26
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ delta.c	
Functions for creating delta and detla-delta coefficients of MFCCs	26
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ delta.h	30
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ fft.c	
Fast Fourier transform and Discrete Cosine tranform functions	30
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ fft.h	33
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ hanning.c	
Functions for creating and applying a Hanning window	33
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ hanning.h	36
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ mel.c	
Functions for generating a Mel filter bank and applying it to an FFT'ed sequence	36
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ mel.h	37
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ mfcc.c	
Functions for creating MFCCs and additional MFCC features, and normalising MFCCs	38
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ mfcc.h	41
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ paa.c	
Functions to apply Piece-wise aggregation to an input signal	41
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature_Extraction/ paa.h	42
/home/bruh/Documents/GitHub/cleaned_and_commented/Misc/ realloc.c	
Safe reallocation functions for arrays	42
/home/bruh/Documents/GitHub/cleaned_and_commented/Misc/ realloc.h	43
/home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/ rt.h	44

/home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/Boundary/ bounds.h	47
/home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/DTW/ dtw.h	24
/home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/Feature/ mfcc.h	41
/home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/KNN/ knn.h	15
/home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/MFCCs/ mfccs.h	44
/home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/Misc/ includes.h	44
/home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/Misc/ mfcc_vars.h	44
/home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/Test/ test.h	60
/home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ bounds.c	
Functions for finding phoneme boundaries in an audio signal	45
/home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ bounds.h	47
/home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ cross_rate.c	
Zero crossing rate functions, primarily used in boundary detection and voice classification . . .	47
/home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ cross_rate.h	50
/home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ ste.c	
Functions for calculating the short time energy of a signal	50
/home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ ste.h	52
/home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ctest.c/ d.h	50
/home/bruh/Documents/GitHub/cleaned_and_commented/Testing/ test.c	
Contains the functions for classifying test data with the trained model	52
/home/bruh/Documents/GitHub/cleaned_and_commented/Testing/ test.h	60
/home/bruh/Documents/GitHub/cleaned_and_commented/Training/ train.c	60
/home/bruh/Documents/GitHub/cleaned_and_commented/Training/ train.h	64

Chapter 4

Data Structure Documentation

4.1 Boundary Struct Reference

Data Fields

- float **min_best_ste**
- float **min_worst_ste**
- float **min_best_zc**
- float **min_worst_zc**
- float **max_best_ste**
- float **max_worst_ste**
- float **max_best_zc**
- float **max_worst_zc**
- float **avg_best_ste**
- float **avg_worst_ste**
- float **avg_best_zc**
- float **avg_worst_zc**
- int **count**

The documentation for this struct was generated from the following files:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ctest.c/ctest.c
- /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ctest.c/ctest_ma_scnd.c

4.2 Bounds Class Reference

Inheritance diagram for Bounds:

Collaboration diagram for Bounds:

The documentation for this class was generated from the following file:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/main.py

4.3 Cluster Struct Reference

Cluster structure :: used to hold the generated 1D Jenks Natural Breaks cluster @centroids can be one or more centroids, these are generated from the @clust function @values these are the values from the original data after it has been sorted and duplicates have been removed the i'th array is associated with the i'th centroid in @centroids @sizes is the length of each @values array with the corresponding index @count is the number of centroids in @centroids @gfv is the Goodness of Variance Fit and is a descriptor of how well the centroids describe the dataset.

```
#include <cluster.h>
```

Data Fields

- float * **centroids**
- float ** **values**
- int * **sizes**
- int **count**
- float **gfv**

4.3.1 Detailed Description

Cluster structure :: used to hold the generated 1D Jenks Natural Breaks cluster @centroids can be one or more centroids, these are generated from the @clust function @values these are the values from the original data after it has been sorted and duplicates have been removed the i'th array is associated with the i'th centroid in @centroids @sizes is the length of each @values array with the corresponding index @count is the number of centroids in @centroids @gfv is the Goodness of Variance Fit and is a descriptor of how well the centroids describe the dataset.

The documentation for this struct was generated from the following file:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Clustering/cluster.h

4.4 codes Struct Reference

Data Fields

- int **amount**

The documentation for this struct was generated from the following file:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Testing/test.c

4.5 Dataset Class Reference

Inheritance diagram for Dataset:

Collaboration diagram for Dataset:

The documentation for this class was generated from the following file:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/main.py

4.6 Feature_Set Struct Reference

Data Fields

- float * **entropy**
- float * **kurtosis**
- float * **zc**
- float * **ste**
- int **coeffs**

The documentation for this struct was generated from the following file:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/dtw.h

4.7 Guess Struct Reference

the [Guess](#) struct is used to store a guess in KNN for sorting and deciding the result @guess index guessed, can be the phoneme, group or voice @diff the difference result used for sorting @ref_indx the MFCC index used to produce the result - used for storing the amount of correct and incorrect guesses for the MFCC

```
#include <knn.h>
```

Collaboration diagram for Guess:

Data Fields

- int **guess**
- double **diff**
- int **ref_indx**
- struct [Phoneme](#) * **ref**
- long double **diff**

4.7.1 Detailed Description

the [Guess](#) struct is used to store a guess in KNN for sorting and deciding the result @guess index guessed, can be the phoneme, group or voice @diff the difference result used for sorting @ref_indx the MFCC index used to produce the result - used for storing the amount of correct and incorrect guesses for the MFCC

The documentation for this struct was generated from the following files:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Clustering/knn.h
- /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/KNN/knn.h

4.8 guess Struct Reference

Data Fields

- long **start**
- long **end**
- char * **name**

The documentation for this struct was generated from the following file:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Testing/[test.c](#)

4.9 KNN Class Reference

Inheritance diagram for KNN:

Collaboration diagram for KNN:

The documentation for this class was generated from the following file:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/main.py

4.10 MyApp Class Reference

Inheritance diagram for MyApp:

Collaboration diagram for MyApp:

Public Member Functions

- def **build** (self)

The documentation for this class was generated from the following file:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/main.py

4.11 MyFrame Class Reference

Inheritance diagram for MyFrame:

Collaboration diagram for MyFrame:

Public Member Functions

- def **runProcess** (self)
- def **update_out** (self, text)
- def **run_dtw** (self)
- def **start_dtw_thread** (self)
- def **change** (self, *args)

The documentation for this class was generated from the following file:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/main.py

4.12 MyGrid Class Reference

Inheritance diagram for MyGrid:

Collaboration diagram for MyGrid:

Public Member Functions

- def **update_values** (self)
- def **reset_values** (self)
- def **change** (self, *args)

The documentation for this class was generated from the following file:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/main.py

4.13 Ph_index Struct Reference

Data Fields

- int **i**
- char **group** [5]
- int **group_i**
- char * **name**
- int **voice**

The documentation for this struct was generated from the following files:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/dtw.h
- /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/DTW/dtw.h

4.14 Phoneme Struct Reference

Collaboration diagram for Phoneme:

Data Fields

- struct [Ph_index](#) * **index**
- struct [Cluster](#) ** **clust**
- struct [Feature_Set](#) ** **feats**
- float ** **mfcc_delta**
- float ** **mfcc_delta_delta**
- float ** **norm_mfcc**
- double **score**
- float ** **mfcc**
- long double ** **sequence**
- int * **size**
- int * **used**
- int * **amounts**
- int **size_count**
- int **reduced_count**
- long double **trained**
- float * **correct**
- float * **error**
- float ** **raw_time**
- int * **raw_sizes**
- int **raw_count**
- int * **count**
- int **use_count**

The documentation for this struct was generated from the following files:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/dtw.h
- /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/DTW/dtw.h

4.15 wav_file Struct Reference

Data Fields

- short * **seq**
- int **length**
- int **offset**

4.15.1 Detailed Description

data and length of data

The documentation for this struct was generated from the following file:

- /home/bruh/Documents/GitHub/cleaned_and_commented/Training/train.h

Chapter 5

File Documentation

5.1 /home/bruh/Documents/GitHub/cleaned_and_commented/↵ Clustering/cluster.c File Reference

A naive Jenks clustering class -.

```
#include "cluster.h"  
Include dependency graph for cluster.c:
```

Functions

- float * **breaks** (float *data, int [length](#), int amount, int offset)
- float **mean** (float *data, int [length](#))
- int **remove_dups** (float *data, int [length](#))
- int **remove_zeros** (float *data, int [length](#))
- int **fltcomp** (const void *a, const void *b)
- void * **threadproc** (void *arg)
- struct [Cluster](#) * **clust** (float *data, int [length](#), int amount, int offset, int num)
- float **SDCM** (float *data, int [length](#))

Variables

- int **tries** [64] = {0}
- float **EX** [64] = {0}
- float **tries_mult** [64] = {1}

5.1.1 Detailed Description

A naive Jenks clustering class -.

5.2 cluster.h

```

1 #ifndef CLUSTER_H
2 #define CLUSTER_H
3
4 #include <memory.h>
5 #include <stdlib.h>
6 #include <stdio.h>
7 #include <float.h>
8 #include <math.h>
9 #include <time.h>
10 #include <pthread.h>
11 #include <unistd.h>
12
13 #include "../Misc/realloc.h"
14 #include "../Dynamic_Time_Warping/dtw.h"
15
16 struct Cluster* clust(float* data, int length, int amount, int offset, int num);
17 float SDCM(float* data, int length);
18
19 struct Cluster {
20     float* centroids;
21     float** values;
22     int* sizes;
23     int count;
24     float gfv;
25 } Cluster;
26
27 #endif

```

5.3 /home/bruh/Documents/GitHub/cleaned_and_commented/↵ Clustering/knn.c File Reference

Contains the functions comparing phonemes using KNN.

```
#include "knn.h"
```

Include dependency graph for knn.c:

5.4 knn.h

```

1 #ifndef KNN_H
2 #define KNN_H
3
4 #include <memory.h>
5 #include <float.h>
6 #include <stdlib.h>
7 #include <stdio.h>
8
9 struct Guess {
10     int guess;
11     double diff;
12     int ref_idx;
13     struct Phoneme* ref;
14 };
15
16 #include "../Dynamic_Time_Warping/dtw.h"
17 #include "cluster.h"
18
19 int knn_mfccs(float** test, int test_length, int k, char* ph);
20 int knn_mfccs_size(float** total_test, int test_length, int k, char* ph);
21 int k_means(float* test, int test_length, int k);
22 int knn_mfccs_size_noref(float** test, int test_length, int k, char* ph);
23 int knn_mfccs_voice_time(float* test, int test_length, int k, char* ph);
24 int knn_mfccs_group_time(float* test, int test_length, int k, char* ph);
25
26 #endif

```

5.5 knn.h

```

1 #ifndef KNN_H
2 #define KNN_H
3
4 #include "../Misc/includes.h"
5
6 struct Guess {
7     int guess;
8     long double diff;
9     int ref_idx;
10    struct Phoneme* ref;
11 };
12
13 int knn_mfccs_size(float* test, int test_length, int k);
14
15 #endif

```

5.6 /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic← _Time_Warping/dtw.c File Reference

The main control file of the program which also contains the DTW functions.

```
#include "dtw.h"
```

Include dependency graph for dtw.c:

Functions

- int **max** (int a, int b)
- int **min** (int a, int b)
- int **compar** (const void *a, const void *b)
- void **export_phones** (void)
- double ** **init_dtw_matrix** (int signal_length, int phone_length, int w)
Initialises a zeroed matrix for use in DTW.
- void * **create_mfcc** (void *argv)
Creates all MFCCs for a given phoneme.
- void * **create_clusters** (void *argv)
- void **clean** (void)
Cleans up any used memory before closing the program.
- void **gen_aoo** (void)
- void **export_mfccs** (void)
Exports all MFCCs in use to files in the main directory.
- void **export_clusters** (void)
Exports all clusters in use.
- void **average_mfccs** (int num)
Averages the MFCC coefficients for each phoneme.
- void **export_for_jenks** (char *p, int coef, float *data, int [data_size](#))
- void **read_jenks** (void)
- void **export_for_pca** (void)
- void **normal_all** (void)
Normalises all MFCCs and deltas.
- void **std_test_output** (void)
Produces standard (non-boundary) tests output to the user.
- void **handle_argv** (int argc, char *argv[])

- Handles all input arguments provided by the user from the command line.*
- void **cluster** (void)
- void **pca** (void)
- void **threaded_mfccs** (void)
 - Generates a thread for each phoneme to produce all MFCCs for that phoneme.*
- void **min_max_values** (void)
 - Assigns the minimum and maximum values found in all MFCCs.*
- void **mean_size_mfccs** (struct [Phoneme](#) *phone)
 - Averages all MFCCs of the same size by phoneme.*
- void **frame_variance** (void)
 - Calculates and outputs the variance among frames of all MFCCs.*
- void **export_device** (void)
 - Exports all MFCCs and config files for use with the physical device.*
- void **non_threaded_mfccs** (void)
 - Generates a thread for each phoneme to produce all MFCCs for that phoneme, only one thread is used.*
- int **seco** (long time)
- int **minu** (long time)
- int **hour** (long time)
- int **main** (int argc, char *argv[])
- void **dtw_init** (void)
- void **dtw_clust** (float **signal, int signal_length, struct [Phoneme](#) *phoneme, short limith)
- double **dtw_clust_result** (float *signal, int signal_length, struct [Phoneme](#) *phoneme, int t, int clust, short limit)
- void **dtw_frame** (float **signal, int signal_length, struct [Phoneme](#) *phoneme, short limit)
 - Performs DTW frame-by-frame for MFCCs.*
- double **dtw_frame_result** (float *signal, int signal_length, struct [Phoneme](#) *phoneme, int p, short limit)
- double **dtw_frame_result_group** (float **signal, int signal_length, struct [Phoneme](#) *phoneme, int p, short limit)
- double **dtw_frame_result_group_time** (float *signal, int signal_length, struct [Phoneme](#) *phoneme, int p, short limit)
- size_t **length** (short *array)
 - Returns the length of an array terminated with SHRT_MAX.*
- int **is_sil_mean** (short *array, int [length](#))
- int **is_sil** (short *array, int [length](#))
- int **is_sil_ste_chunk** (short *array, int [length](#))
- int **is_sil_ste** (short *array, int [length](#))
- int **is_sil_zc** (short *array, int [length](#))
- int **is_sil_flat** (short *array, int [length](#))
- int **is_sil_db** (short *array, int [length](#))
- void **cancel_threads** (void)
 - Closes all threads in use.*
- void **interrupt_handler** (int signo)
 - Handles a signal interrupt from the user.*
- void **mask_sig** (void)
 - Masks any signals for additional threads other than main.*

Variables

- int **gbl_banks** = 40
- int **gbl_window_width** = 128
- int **gbl_paa_op** = 1
- int **gbl_paa** = 2
 - Applies PAA to the input sequence.*
- int **gbl_dtw_window** = 200
- int **gbl_interval_div** = 2
- int **gbl_nfft** = 512
- int **gbl_mfcc_num** = 25
- int **gbl_clust_num** = 0
- int **gbl_k** = 7
- float **gbl_test_trunc** = 24
- int **no_window** = 0
- int **gbl_test_iter** = 1
- int **gbl_group_k** = 7
- int **gbl_voice_k** = 7
- int **gbl_frame_limit** = INT_MAX
- int **gbl_zc_incr**
- int **gbl_ste_incr**
- int **gbl_entr_incr**
- int **gbl_neg_incr**
- int **gbl_larg_incr**
- int **gbl_larg_entr_incr**
- int **gbl_larg_ste_incr**
- int **largest_mfcc** = 0
- int **largest_index** = 0
- float **largest_value** = 0
- float **smallest_value** = FLT_MAX
- short **failed** = 0
- int **DTW_ERROR**
- int **DELTA** = 0
- int **DELTA_DELTA** = 0
- int **NORM** = 0
- int **SIMPLE** = 0
- int **EXTRA** = 0
- int **CLUST** = 0
- int **BOUNDS** = 0
- int **LOG_E** = 0
- int **AVG** = 0
- int **KNN** = 0
- int **GROUP** = 0
- int **VOICED** = 0
- int **GRAM** = 0
- int **PCA** = 0
- int **ZC** = 0
- int **STE** = 0
- int **MEAN_SIZE** = 0
- int **KURT** = 0
- int **ENTR** = 0
- int **SPKR1** = 0
- int **SPKR1_NOSIL** = 0
- int **WHITE** = 0
- int **STRT** = 0

- int **CAR** = 0
- int **CAFE** = 0
- int **SPLIT_DATA** = 0
- int **SVM** = 0
- int **EXPORT** = 0
- int **THREAD** = 0
- int **MALE** = 0
- int **FEMALE** = 0
- int **trained** = 0
- int **tested** = 0
- int **mfcced** = 0
- int **clustered** = 0
- int **exported** = 0
- double **best_so_far** = DBL_MAX
- time_t **avg_test_time** = 0
- long double **total_test_time** = 0
- long double **total_dtw_tests** = 0
- time_t **start** = 0
- pthread_t * **threads** = NULL
- int **thread_count** = 0
- int **clusters_left** = 0

5.6.1 Detailed Description

The main control file of the program which also contains the DTW functions.

Author

T. Buckingham

Date

Wed May 4 15:41:32 2022

This file contains program dependant configuration variables along with all required dynamic time warping files. This file was decided to contain these and

5.6.2 Function Documentation

5.6.2.1 average_mfccs()

```
void average_mfccs (  
    int num )
```

Averages the MFCC coefficients for each phoneme.

Parameters

<i>num</i>	The phoneme index.
------------	--------------------

5.6.2.2 create_mfcc()

```
void * create_mfcc (
    void * argv )
```

Creates all MFCCs for a given phoneme.

Parameters

<i>argv</i>	The phoneme index to create MFCCs for
-------------	---------------------------------------

Returns

Not used

Creates all MFCCs from all raw data signals for the phoneme index provided.

5.6.2.3 dtw_frame()

```
void dtw_frame (
    float ** signal,
    int signal_length,
    struct Phoneme * phoneme,
    short limit )
```

Performs DTW frame-by-frame for MFCCs.

Parameters

<i>signal</i>	The input signal MFCC to be compared
<i>signal_length</i>	The length of
<i>signal</i>	in frames
<i>phoneme</i>	The phoneme to compare to
<i>limit</i>	The DTW window limit

5.6.2.4 export_device()

```
void export_device (
    void )
```

Exports all MFCCs and config files for use with the physical device.

This method produces a folder to be copied onto the device's SD card, all required information will be loaded by the device

5.6.2.5 `handle_argv()`

```
void handle_argv (
    int argc,
    char * argv[] )
```

Handles all input arguments provided by the user from the command line.

Parameters

<i>argc</i>	The number of arguments provided
<i>argv</i>	The arguments as strings

5.6.2.6 `init_dtw_matrix()`

```
double ** init_dtw_matrix (
    int signal_length,
    int phone_length,
    int w )
```

Initialises a zeroed matrix for use in DTW.

Parameters

<i>signal_length</i>	The length of the test's sequence MFCC in frames
<i>phone_length</i>	The length of the phoneme's sequence MFCC in frames
<i>w</i>	The windowing limit

Returns

The zero initialised matrix

5.6.2.7 `interrupt_handler()`

```
void interrupt_handler (
    int signo )
```

Handles a signal interrupt from the user.

Parameters

<i>signo</i>	The signal flag
--------------	-----------------

When a CTRL+C (SIGINT) is sent to the program the used memory is cleaned and the current stats are displayed before exiting

5.6.2.8 `length()`

```
size_t length (
    short * array )
```

Returns the length of an array terminated with SHRT_MAX.

Parameters

<i>array</i>	The array to find the length of
--------------	---------------------------------

Returns

The length of

Parameters

<i>array</i>	
--------------	--

No longer used in any function as the size is passed instead of finding it.

5.6.2.9 `mean_size_mfccs()`

```
void mean_size_mfccs (
    struct Phoneme * phone )
```

Averages all MFCCs of the same size by phoneme.

Parameters

<i>phone</i>	The phoneme to average MFCCs for
--------------	----------------------------------

5.6.3 Variable Documentation

5.6.3.1 `gbl_paa`

```
length var gbl_paa = 2
```

Applies PAA to the input sequence.

Parameters

<i>sequence</i>	The input signal to be processed
<i>length</i>	The length of
<i>input</i>	

Returns

The new processed signal of size

The input signal is divided by the PAA amount set by the user, with each grouping being averaged into one value

Parameters

<i>sequence</i>	The input signal to be processed
<i>length</i>	The length of
<i>input</i>	

Returns

The new processed signal of size

The input signal is divided by the PAA amount set by the user, with each grouping being averaged into one value.
The same as

5.7 dtw.h

```

1 #ifndef DTW_H
2 #define DTW_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <limits.h>
7 #include <float.h>
8 #include <string.h>
9 #include <math.h>
10 #include <time.h>
11 #include <signal.h>
12 #include <fenv.h>
13 #include <errno.h>
14 #include <pthread.h>
15
16 #include <sys/types.h>
17 #include <sys/stat.h>
18 #include <unistd.h>
19 #include <ftw.h>
20
21 #include "../Clustering/cluster.h"
22
23 struct Ph_index {
24     int i;
25     char group[5];
26     int group_i;
27     char* name;
28     int voice;
29 };
30
31 struct Feature_Set {
32     float* entropy;
33     float* kurtosis;
34     float* zc;

```

```

35     float* ste;
36     int coeffs;
37     // float flatness;
38     // float skewness;
39     // float spread;
40 };
41
42 struct Phoneme {
43     struct Ph_index* index;
44     struct Cluster** clust;
45     struct Feature_Set** feats;
46     float** mfcc_delta;
47     float** mfcc_delta_delta;
48     float** norm_mfcc;
49     double score;
50     float** mfcc;
51     long double** sequence;
52     // int* coeffs;
53     int* size;
54     int* used;
55     int* amounts;
56     int size_count;
57     int reduced_count;
58     long double trained;
59     float* correct;
60     float* error;
61     float** raw_time;
62     int* raw_sizes;
63     int raw_count;
64 } ph;
65
66 #include "../Training/train.h"
67 #include "../Misc/realloc.h"
68 #include "../Testing/test.h"
69 #include "../Feature_Extraction/paa.h"
70 #include "../Feature_Extraction/mfcc.h"
71 #include "../Feature_Extraction/delta.h"
72
73 #define PTHREAD_CANCELED ((void *) -1)
74
75 // Global options - may be best to make them local
76 extern int glbl_banks;
77 extern int glbl_window_width;
78 extern int glbl_paa_op;
79 extern int glbl_paa;
80 extern int glbl_dtw_window;
81 extern int glbl_interval_div;
82 extern int glbl_nfft;
83 extern int glbl_mfcc_num;
84 extern int no_window;
85 extern int glbl_clust_num;
86 extern int glbl_k;
87 extern int glbl_test_iter;
88 extern int glbl_group_k;
89 extern int glbl_voice_k;
90 extern int glbl_frame_limit;
91
92 extern int glbl_zc_incr;
93 extern int glbl_ste_incr;
94 extern int glbl_entr_incr;
95 extern int glbl_neg_incr;
96 extern int glbl_larg_incr;
97 extern int glbl_larg_entr_incr;
98 extern int glbl_larg_ste_incr;
99
100 extern int DELTA;
101 extern int DELTA_DELTA;
102 extern float glbl_test_trunc;
103
104 #define NO_SIG_LEN 1
105 extern int DTW_ERROR;
106 extern int SIMPLE;
107 extern int MALE;
108 extern int FEMALE;
109 extern int EXTRA;
110 extern int CLUST;
111 extern int BOUNDS;
112 extern int Z_ZC;
113 extern int ONE;
114 extern int LOG_E;
115 extern int KNN;
116 extern int GROUP;
117 extern int TO_RUN;
118 extern int GRAM;
119 extern int NORM;
120 extern int ZC;
121 extern int STE;

```

```

122 extern int VOICED;
123 extern int ENTR;
124 extern int KURT;
125 extern int SPKR1;
126 extern int SPKR1_NOSIL;
127 extern int WHITE;
128 extern int STRT;
129 extern int CAR;
130 extern int CAFE;
131 extern int SPLIT_DATA;
132 extern int SVM;
133
134 // Testing externs
135 extern float* aaomfcc;
136 extern int aaomfcc_length;
137 extern double against_all[];
138
139 // Output globals
140 extern double best_so_far;
141 extern short failed;
142 extern int trained;
143 extern int tested;
144 extern time_t start;
145
146 extern long double total_test_time;
147 extern long double total_dtw_tests;
148
149 extern pthread_t* threads;
150 extern int thread_count;
151
152 void dtw(float* signal, int signal_length, struct Phoneme* phoneme, short limit);
153 void dtw_init();
154 size_t length(short* array);
155 void export_phone(struct Phoneme* phoneme);
156 void dtw_test(float* signal, float* sequence, int signal_length, int signal_length2, short limit);
157 int seco(long time);
158 int minu(long time);
159 int hour(long time);
160 int is_sil(short* array, int length);
161 void dtw_frame(float** signal, int signal_length, struct Phoneme* phoneme, short limit);
162 int min(int a, int b);
163 int max(int a, int b);
164 void interrupt_handler(int signo);
165 int is_sil_db(short* array, int length);
166 int is_sil_mean(short* array, int length);
167 void dtw_clust(float** signal, int signal_length, struct Phoneme* phoneme, short limit);
168 void mask_sig(void);
169 double dtw_frame_result(float* signal, int signal_length, struct Phoneme* phoneme, int p, short limit);
170 double dtw_clust_result(float* signal, int signal_length, struct Phoneme* phoneme, int t, int clust,
    short limit);
171 double dtw_frame_result_group(float** signal, int signal_length, struct Phoneme* phoneme, int p, short
    limit);
172 double dtw_frame_result_group_time(float* signal, int signal_length, struct Phoneme* phoneme, int p,
    short limit);
173 int is_sil_zc(short* array, int length);
174 int is_sil_ste(short* array, int length);
175 int is_sil_flat(short* array, int length);
176 int is_sil_ste_chunk(short* array, int length);
177
178
179 #endif

```

5.8 dtw.h

```

1 #ifndef DTW_H
2 #define DTW_H
3
4 #include "../Misc/includes.h"
5
6 struct Ph_index {
7     int i;
8     char group[5];
9     char* name;
10 };
11
12 struct Phoneme {
13     struct Ph_index* index;
14     double score;
15     float** mfcc;
16     int* size;
17     int* count;
18     int size_count;
19     int use_count;

```



```
20 } ph;
21
22 long double dtw_frame_result(float* signal, int signal_length, struct Phoneme* phoneme, int p);
23
24 #endif
```

5.9 /home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic_Time_Warping/jenks.py File Reference

Creates clusters for each phoneme.

Functions

- `def main (argv)`

Variables

- string `rootdir` = `"../Jenks/"`

5.9.1 Detailed Description

Creates clusters for each phoneme.

Produces clusters for MFCCs using.

Exports all MFCCs to files so that can process them.

Reads in the Jenks centroids generated by.

A K-means function for use with.

is now used instead

Parameters

<i>argv</i>	Not used
-------------	----------

Returns

Not used

This function is mostly unused due to the use of

Parameters

<i>signal</i>	The signal to be classified
<i>signal_length</i>	The length of

Parameters

<i>signal</i>	
<i>phoneme</i>	The phoneme to be compared to
<i>limit</i>	The window limit for DTW
<i>p</i>	The phoneme code to be exported
<i>coef</i>	The coefficient number to be exported
<i>data</i>	The MFCC data for
<i>p</i>	and
<i>coef</i>	
<i>data_size</i>	The length of the
<i>data</i>	array

5.10 [/home/bruh/Documents/GitHub/cleaned_and_commented/](#) Dynamic_Time_Warping/pca.py File Reference

Exports all MFCCs in use for use with.

Functions

- def **main** (mean, plot)

Variables

- string **rootdir** = "../PCA_Data/"

5.10.1 Detailed Description

Exports all MFCCs in use for use with.

Exports the MFCCs for use with then cleans and exits.

5.11 [/home/bruh/Documents/GitHub/cleaned_and_commented/Feature](#) _Extraction/delta.c File Reference

Functions for creating delta and detla-delta coefficients of MFCCs.

```
#include "delta.h"
Include dependency graph for delta.c:
```

Functions

- void `normalise_delta` (float *`delta`, int `length`)
Normalises a delta coefficient array.
- void `normalise_delta_delta` (float *`delta`, int `length`)
Normalises a delta-delta coefficient array.
- void `update_deltas_norm` (float *`delta`, int `length`)
Update the minimum and maximum values of the delta coefficients.
- void `update_delta_deltas_norm` (float *`delta_delta`, int `length`)
Update the minimum and maximum values of the delta-delta coefficients.
- float * `delta` (float *`mfcc`, int `size`)
Calculate the delta coefficients of from an MFCC.
- float * `delta_delta` (float *`mfcc_delta`, int `size`)
Calculate the delta-delta coefficients of from a delta coefficient array.

Variables

- float `min_delta` = FLT_MAX
- float `max_delta` = FLT_MIN
- float `min_delta_delta` = FLT_MAX
- float `max_delta_delta` = FLT_MIN

5.11.1 Detailed Description

Functions for creating delta and delta-delta coefficients of MFCCs.

Author

T. Buckingham

Date

Wed May 4 23:15:48 2022

5.11.2 Function Documentation

5.11.2.1 `delta()`

```
float * delta (
    float * mfcc,
    int size )
```

Calculate the delta coefficients of from an MFCC.

Parameters

<i>mfcc</i>	The MFCC to process
<i>size</i>	The length of
<i>MFCC</i>	

Returns

The delta coefficient array

5.11.2.2 delta_delta()

```
float * delta_delta (
    float * mfcc_delta,
    int size )
```

Calculate the delta-delta coefficients of from a delta coefficient array.

Parameters

<i>mfcc</i>	The delta coefficient array to process
<i>size</i>	The length of
<i>mfcc_delta</i>	

Returns

The delta coefficient array

5.11.2.3 normalise_delta()

```
void normalise_delta (
    float * delta,
    int length )
```

Normalises a delta coefficient array.

Parameters

<i>delta</i>	The delta array to be normalised
<i>length</i>	The length of @delta

5.11.2.4 normalise_delta_delta()

```
void normalise_delta_delta (
    float * delta,
    int length )
```

Normalises a detal-delta coefficient array.

Parameters

<i>delta</i>	The delta-delta coefficients array to be normalised
<i>length</i>	The length of
<i>delta</i>	

5.11.2.5 update_delta_deltas_norm()

```
void update_delta_deltas_norm (
    float * delta_delta,
    int length )
```

Update the minimum and maximum values of the delta-delta coefficients.

Parameters

<i>delta</i>	The delta-delta sequence to process
<i>length</i>	The length of
<i>delta-delta</i>	

5.11.2.6 update_deltas_norm()

```
void update_deltas_norm (
    float * delta,
    int length )
```

Update the minimum and maximum values of the delta coefficients.

Parameters

<i>delta</i>	The delta sequence to process
<i>length</i>	The length of
<i>delta</i>	

5.12 delta.h

```

1 #ifndef DELTA_H
2 #define DELTA_H
3
4 #include <memory.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <float.h>
8
9 #include "../Clustering/cluster.h"
10 #include "../Dynamic_Time_Warping/dtw.h"
11
12 float* delta(float* mfcc, int size);
13 float* delta_delta(float* mfcc_delta, int size);
14 void update_deltas_norm(float* delta, int length);
15 void update_delta_deltas_norm(float* delta, int length);
16 void normalise_delta_delta(float* delta, int length);
17 void normalise_delta(float* delta, int length);
18
19 #endif

```

5.13 /home/bruh/Documents/GitHub/cleaned_and_commented/Feature_↵ _Extraction/fft.c File Reference

Fast Fourier transform and Discrete Cosine tranform functions.

```
#include "fft.h"
```

Include dependency graph for fft.c:

Functions

- float * [dct](#) (float *array, int width)
DCT function used during the MFCC process.
- float ** [fft_chunks](#) (float **chunks, int n, int [length](#))
Produces an 2 dimensional array of FFT magnitudes.
- float complex * [fft](#) (float complex *chunk, int [length](#))
Performs an FFT on a given array.

5.13.1 Detailed Description

Fast Fourier transform and Discrete Cosine tranform functions.

Author

T. Buckingham

Date

Thu May 5 14:48:03 2022

5.13.2 Function Documentation

5.13.2.1 dct()

```
float * dct (
    float * array,
    int width )
```

DCT function used during the MFCC process.

Parameters

<i>array</i>	The MFCC to be processed
<i>width</i>	The length of
<i>array</i>	

Returns**5.13.2.2 fft()**

```
float complex * fft (
    float complex * chunk,
    int length )
```

Performs an FFT on a given array.

Parameters

<i>chunk</i>	The complex array to be processed
<i>length</i>	The length of
<i>chunk</i>	

Returns

The FFT in the form of a complex number, no magnitude taken

Before processing the input must be in complex form and the return will be also be in complex form and so the magnitude will need to be taken for further MFCC processing.

5.13.2.3 fft_chunks()

```
float ** fft_chunks (
    float ** chunks,
    int n,
    int length )
```

Produces an 2 dimensional array of FFT magnitudes.

Parameters

<i>chunks</i>	The Hanning window sequences from an audio signal
<i>n</i>	The number of Hanning windows to be processed
<i>length</i>	The length of each Hanning window

Returns

A 2D array of FFT arrays

Each 'chunk' passed will have an FFT applied and the magnitude extracted, the same amount of arrays will be returned however, they will all be of length (

Parameters

<i>length</i>	/ 2)
---------------	------

5.14 fft.h

```

1 #ifndef FFT_H
2 #define FFT_H
3
4 #include <stdlib.h>
5 #include <stdio.h>
6 #include <memory.h>
7 #include <math.h>
8 #include <complex.h>
9
10 #ifndef M_PI
11 #define M_PI (3.14159265358979323846)
12 #endif
13
14
15 float** fft_chunks(float** chunks, int n, int length);
16 float complex* fft(float complex* chunk, int length);
17 float* dct(float* array, int width);
18
19 #endif

```

5.15 /home/bruh/Documents/GitHub/cleaned_and_commented/Feature_↵ _Extraction/hanning.c File Reference

Functions for creating and applying a Hanning window.

```
#include "hanning.h"
```

Include dependency graph for hanning.c:

Functions

- float * [hanning_window](#) (int num)
Produces a Hanning window of the given length.
- float ** [hanning_chunks](#) (float *input, int [length](#), int incr)
Applies a Hanning window to each given chunk of an audio input.
- float ** [hanning_chunks_no_overlap](#) (float *input, int [length](#), int incr)
Applies a Hanning window to each given chunk of an audio input with no overlapping of the windows.
- float * [hanning](#) (float *array, int num)
Applies a Hanning function to a given signal.

5.15.1 Detailed Description

Functions for creating and applying a Hanning window.

Author

T. Buckingham

Date

Thu May 5 17:41:08 2022

5.15.2 Function Documentation

5.15.2.1 hanning()

```
float * hanning (
    float * array,
    int num )
```

Applies a Hanning function to a given signal.

Parameters

<i>array</i>	The input signal to apply the Hanning function to
<i>num</i>	The width of the window

Returns

The result of applying the Hanning function to the given input signal

5.15.2.2 hanning_chunks()

```
float ** hanning_chunks (
    float * input,
    int length,
    int incr )
```

Applies a Hanning window to each given chunk of an audio input.

Parameters

<i>input</i>	The audio signal to be processed
<i>length</i>	The length of
<i>length</i>	
<i>incr</i>	The window width to use

Returns

The array of Hanning windows

The input signal will be blocked into chunks of size

Parameters

<i>incr</i>	then each chunk will have a Hanning window applied.
-------------	---

5.15.2.3 hanning_chunks_no_overlap()

```
float ** hanning_chunks_no_overlap (
    float * input,
    int length,
    int incr )
```

Applies a Hanning window to each given chunk of an audio input with no overlapping of the windows.

Parameters

<i>input</i>	The audio signal to be processed
<i>length</i>	The length of
<i>length</i>	
<i>incr</i>	The window width to use

Returns

The array of Hanning windows

The input signal will be blocked into chunks of size

Parameters

<i>incr</i>	then each chunk will have a Hanning window applied.
-------------	---

5.15.2.4 hanning_window()

```
float * hanning_window (
    int num )
```

Produces a Hanning window of the given length.

Parameters

<i>num</i>	The length of the Hanning window to produce
------------	---

Returns

The Hanning window of length

Parameters

<i>num</i>	
------------	--

5.16 hanning.h

```

1 #ifndef HANNING_H
2 #define HANNING_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <math.h>
7 #include <memory.h>
8 #include "../Training/train.h"
9 #include "../Seperation/cross_rate.h"
10
11 #ifndef M_PI
12 #define M_PI (3.14159265358979323846)
13 #endif
14
15 float* hanning(float* array, int num);
16 float* hanning_window(int num);
17 float** hanning_chunks_no_overlap(float* input, int length, int incr);
18 float** hanning_chunks(float* input, int length, int incr);
19
20 #endif

```

5.17 /home/bruh/Documents/GitHub/cleaned_and_commented/Feature_↵ _Extraction/mel.c File Reference

Functions for generating a Mel filter bank and applying it to an FFT'ed sequence.

```
#include "mel.h"
```

Include dependency graph for mel.c:

Functions

- float ** **mel_fb** (int width, int banks)
- float * **mel** (float *array, int width, int banks)

Applies a generated Mel filter bank to the input array.

5.17.1 Detailed Description

Functions for generating a Mel filter bank and applying it to an FFT'ed sequence.

Author

T. Buckingham

Date

Thu May 5 14:56:25 2022

5.17.2 Function Documentation

5.17.2.1 mel()

```
float * mel (
    float * array,
    int width,
    int banks )
```

Applies a generated Mel filter bank to the input array.

Parameters

<i>array</i>	The FFT'ed sequence the filter bank is to be applied to
<i>width</i>	The length of
<i>array</i>	
<i>banks</i>	The number of desired filter banks

Returns

The result of applying the Mel filter bank to the input FFT'ed sequence.

5.18 mel.h

```
1 #ifndef MEL_H
2 #define MEL_H
3
4 #include <memory.h>
5 #include <stdlib.h>
6 #include <stdio.h>
7 #include <math.h>
8 #include <float.h>
9
10 #include "../Clustering/cluster.h"
11 #include "../Dynamic_Time_Warping/dtw.h"
12
13 float** mel_fb(int width, int banks);
14 float* mel(float* array, int width, int banks);
15
16 #endif
```

5.19 /home/bruh/Documents/GitHub/cleaned_and_commented/Feature_↵ _Extraction/mfcc.c File Reference

Functions for creating MFCCs and additional MFCC features, and normalising MFCCs.

```
#include "mfcc.h"
```

Include dependency graph for mfcc.c:

Functions

- float `log_entropy` (float *sequence, int inc)
Calculates the log entropy of a given signal.
- float `log_energy` (float *chunk, int `length`)
Calculates the log energy of a given signal.
- float `st_energy` (float *chunk, int `length`)
Calculates the short time energy of a given signal.
- float `kurtosis` (float *chunk, int `length`)
Calculates the kurtosis of a given signal.
- void `normalise_mfcc` (float *`mfcc`, int `length`)
- void `update_mfcc_norm` (float *`mfcc`, int `length`)
- float * `mfcc` (float *sequence, int width, int incr, int banks, int paa)
The control function that generates an MFCC from the given input audio sequence.

Variables

- float `min_mfcc` = FLT_MAX
- float `max_mfcc` = FLT_MIN
Normalises an MFCC using.

5.19.1 Detailed Description

Functions for creating MFCCs and additional MFCC features, and normalising MFCCs.

Author

T. Buckingham

Date

Thu May 5 16:06:24 2022

5.19.2 Function Documentation

5.19.2.1 kurtosis()

```
float kurtosis (
    float * chunk,
    int length )
```

Calculates the kurtosis of a given signal.

Parameters

<i>sequence</i>	The signal to be processed.
<i>inc</i>	The length of
<i>sequence.</i>	

Returns

The log kurtosis value.

5.19.2.2 log_energy()

```
float log_energy (
    float * chunk,
    int length )
```

Calculates the log energy of a given signal.

Parameters

<i>sequence</i>	The signal to be processed.
<i>inc</i>	The length of
<i>sequence.</i>	

Returns

The log energy value.

5.19.2.3 log_entropy()

```
float log_entropy (
    float * sequence,
    int inc )
```

Calculates the log entropy of a given signal.

Parameters

<i>sequence</i>	The signal to be processed.
<i>inc</i>	The length of
<i>sequence.</i>	

Returns

The log entropy value.

5.19.2.4 mfcc()

```
float * mfcc (
    float * sequence,
    int width,
    int incr,
    int banks,
    int paa )
```

The control function that generates an MFCC from the given input audio sequence.

Parameters

<i>sequence</i>	The audio sequence to be processed
<i>width</i>	The length of
<i>sequence</i>	
<i>incr</i>	The window width used for Hanning
<i>banks</i>	The number of Mel filter banks to use
<i>paa</i>	The Piece-wise aggregation divisor

Returns

The generated MFCC.

5.19.2.5 st_energy()

```
float st_energy (
    float * chunk,
    int length )
```

Calculates the short time energy of a given signal.

Parameters

<i>sequence</i>	The signal to be processed.
<i>inc</i>	The length of
<i>sequence.</i>	

Returns

The short time energy value.

5.19.3 Variable Documentation

5.19.3.1 max_mfcc

min_mfcc and var max_mfcc = FLT_MIN

Normalises an MFCC using.

Parameters

<i>mfcc</i>	The MFCC sequence to be normalised.
<i>length</i>	The length of
<i>mfcc</i>	

5.20 mfcc.h

```
1 #ifndef MFCC_H
2 #define MFCC_H
3
4 #include "mel.h"
5 #include "paa.h"
6 #include "hanning.h"
7 #include "fft.h"
8
9 #include "../Clustering/cluster.h"
10
11 float* mfcc(float* sequence, int size, int window, int banks, int paa);
12 float log_energy(float* chunk, int length);
13 void update_mfcc_norm(float* mfcc, int length);
14 void normalise_mfcc(float* mfcc, int length);
15 float kurtosis(float* chunk, int length);
16 float log_entropy(float* sequence, int inc);
17
18 #endif
```

5.21 mfcc.h

```
1 #ifndef MFCC_H
2 #define MFCC_H
3
4 #include "../Misc/includes.h"
5
6 int frame_amount(int signal_length);
7 int mfcc_size(int signal_length);
8 float* mfcc_quick(float* sequence, int width, int incr, int banks, int paa);
9
10 #endif
```

5.22 /home/bruh/Documents/GitHub/cleaned_and_commented/Feature_↵ _Extraction/paa.c File Reference

Functions to apply Piece-wise aggregation to an input signal.

```
#include "paa.h"
```

Include dependency graph for paa.c:

Functions

- short * **paa** (short *sequence, int [length](#))
- float * **f_paa** (float *sequence, int [length](#))

5.22.1 Detailed Description

Functions to apply Piece-wise aggregation to an input signal.

Author

T. Buckingham

Date

Thu May 5 15:50:21 2022

5.23 paa.h

```
1 #ifndef PAA_H
2 #define PAA_H
3
4 #include <stdlib.h>
5 #include <stdio.h>
6 #include <math.h>
7 #include <memory.h>
8 #include "../Training/train.h"
9 #include "../Misc/realloc.h"
10
11 short* paa(short* sequence, int length);
12 float* f_paa(float* sequence, int length);
13
14 #endif
```

5.24 /home/bruh/Documents/GitHub/cleaned_and_commented/↵ Misc/realloc.c File Reference

Safe reallocation functions for arrays.

```
#include "realloc.h"
Include dependency graph for realloc.c:
```

Functions

- short * **s_realloc** (short *input, int [length](#))
Reallocates @input to size @length.
- float * **f_realloc** (float *input, int [length](#))
- int * **i_realloc** (int *input, int [length](#))

5.24.1 Detailed Description

Safe reallocation functions for arrays.

Author

T. Buckingham

Date

Fri Apr 29 01:07:49 2022

A range of reallocation functions which contain checks before return the new array. Contains a int, short and float variant.

5.24.2 Function Documentation

5.24.2.1 s_realloc()

```
short * s_realloc (
    short * input,
    int length )
```

Reallocates @input to size @length.

Parameters

<i>input</i>	The array to reallocate
<i>length</i>	The resulting size of the new array

Returns

An array of size @length

5.25 realloc.h

```
1 #ifndef REALLOC_H
2 #define REALLOC_H
3
4 #include <memory.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 short* s_realloc(short* input, int length);
9 float* f_realloc(float* input, int length);
10 int* i_realloc(int* input, int length);
11
12 #endif
```

5.26 mfccs.h

```
1 #ifndef MFCCS_H
2 #define MFCCS_H
3
4 #include "../Misc/includes.h"
5
6 struct Phoneme* get_mfcc(struct Phoneme* phone, int length);
7 void dtw_init(void);
8
9 extern int truncation;
10 extern struct Phoneme** phones;
11 extern char* p_codes[];
12
13 #endif
14
```

5.27 includes.h

```
1 #include <memory.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <float.h>
5 #include <stdlib.h>
6 #include <limits.h>
7 #include <string.h>
8 #include <complex.h>
9 #include <stdio.h>
10 #include <unistd.h>
11 #include <pthread.h>
12 #include <dirent.h>
13
14 #include "../MFCCs/mfccs.h"
15 #include "../Misc/mfcc_vars.h"
16 #include "../Test/test.h"
17 #include "../DTW/dtw.h"
18 #include "../Feature/mfcc.h"
19 #include "../Boundary/bounds.h"
20 #include "../KNN/knn.h"
```

5.28 mfcc_vars.h

```
1 #ifndef MFCC_VARS_H
2 #define MFCC_VARS_H
3
4 #include "includes.h"
5
6 extern int num_ph;
7 extern int glbl_paa;
8 extern int glbl_paa_op;
9 extern int glbl_window_width;
10 extern int glbl_banks;
11 extern float glbl_dtw_window;
12 extern int glbl_interval_div;
13 extern int glbl_nfft;
14 extern float glbl_test_trunc;
15
16 #endif
```

5.29 rt.h

```
1 #ifndef RT_H
2 #define RT_H
3
4 #include "../Misc/includes.h"
5
6 void record(void);
7 int check_in(void);
8
9 #endif
```

5.30 /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/bounds.c File Reference

Functions for finding phoneme boundaries in an audio signal.

```
#include "bounds.h"
```

Include dependency graph for bounds.c:

Functions

- float [get_entropy](#) (short *sequence, int inc)
Calculates the entropy value of a given audio chunk.
- short * [shift_and_reduce](#) (short *sequence, int [length](#), int shift)
Creates a new array and moves the remaining values to the new array.
- int [next_boundary](#) (short *sequence, int [length](#))
Finds and returns the next boundary in audio signal.

5.30.1 Detailed Description

Functions for finding phoneme boundaries in an audio signal.

Author

T. Buckingham

Date

Thu May 5 18:05:53 2022

5.30.2 Function Documentation

5.30.2.1 [get_entropy\(\)](#)

```
float get_entropy (  
    short * sequence,  
    int inc )
```

Calculates the entropy value of a given audio chunk.

Parameters

<i>sequence</i>	The audio chunk to process
<i>inc</i>	The length of
<i>sequence</i>	

Returns

The entropy value of

Parameters

<i>sequence</i>	
-----------------	--

5.30.2.2 next_boundary()

```
int next_boundary (
    short * sequence,
    int length )
```

Finds and returns the next boundary in audio signal.

Parameters

<i>sequence</i>	The audio signal to process
<i>length</i>	The length of
<i>sequence</i>	

Returns

The next boundary found in the audio sequence

The boundary is found using entropy, zero cross and short time energy change thresholds which are defined by the user when running the program.

5.30.2.3 shift_and_reduce()

```
short * shift_and_reduce (
    short * sequence,
    int length,
    int shift )
```

Creates a new array and moves the remaining values to the new array.

Parameters

<i>sequence</i>	The sequence to procces
<i>length</i>	The length of
<i>sequence</i>	
<i>shift</i>	The amount of data indexed from 0 to remove

Returns

The new array with the data from 0 ->

Parameters

<i>shift</i>	
--------------	--

Once a boundary has been found and processed this data needs to be removed from the array, this removes this data and returns a new array to save memory and prevent a memory leak.

5.31 bounds.h

```
1 #ifndef BOUNDS_H
2 #define BOUNDS_H
3
4 #include "../Misc/includes.h"
5
6 int shift_and_reduce(short* sequence, int length, int shift);
7 int next_boundary(short* sequence, int length);
8 float get_entropy(short* sequence, int inc);
9 int next_boundary(short* sequence, int length);
10 int is_positive(short num);
11 int f_is_positive(float num);
12
13 #endif
```

5.32 bounds.h

```
1 #ifndef BOUNDS_H
2 #define BOUNDS_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <math.h>
7 #include <memory.h>
8 #include <float.h>
9 #include "cross_rate.h"
10 #include "ste.h"
11 #include "../Feature_Extraction/fft.h"
12
13 int next_boundary(short* sequence, int length);
14 short* shift_and_reduce(short* sequence, int length, int shift);
15
16 #endif
```

5.33 /home/bruh/Documents/GitHub/cleaned_and_commented/↵ Seperation/cross_rate.c File Reference

Zero crossing rate functions, primarily used in boundary detection and voice classification.

```
#include "cross_rate.h"
```

Include dependency graph for cross_rate.c:

Functions

- int `is_positive` (short num)
Check if a short is positive.
- int `f_is_positive` (float num)
- int `cross_rate` (short *signal, int signal_length)
Calculates the crossing rate of the given signal.
- float `favg_cross_rate` (short *signal, int signal_length)
Calculates the average cross rate of the given signal.
- float `f_cross_rate` (float *signal, int signal_length)
- float `stavg_cross_rate_no_overlap` (float *signal, int signal_length)
- float `stavg_cross_rate` (float *signal, int signal_length)
Calculates the short time zero cross average of a signal signal.

5.33.1 Detailed Description

Zero crossing rate functions, primarily used in boundary detection and voice classification.

Author

T. Buckingham

Date

Tue May 17 00:23:35 2022

5.33.2 Function Documentation

5.33.2.1 `cross_rate()`

```
int cross_rate (
    short * signal,
    int signal_length )
```

Calculates the crossing rate of the given signal.

Parameters

<i>signal</i>	The signal to process
<i>signal_length</i>	The length of @signal

Returns

The total crossing rate of the signal

5.33.2.2 favg_cross_rate()

```
float favg_cross_rate (
    short * signal,
    int signal_length )
```

Calculates the average cross rate of the given signal.

Parameters

<i>signal</i>	The signal to process
<i>signal_length</i>	The length of @signal

Returns

The averaged crossing rate as a floating point

5.33.2.3 is_positive()

```
int is_positive (
    short num )
```

Check if a short is postivie.

Parameters

<i>num</i>	The number to check
------------	---------------------

Returns

1 if positive, 0 if negative or zero.

5.33.2.4 stavg_cross_rate()

```
float stavg_cross_rate (
    float * signal,
    int signal_length )
```

Calculates the short time zero cross average of a signal signal.

Parameters

<i>signal</i>	The signal to process
<i>signal_length</i>	The length of @signal

Returns

The short time averaged crossing rate as a floating point

The signal is windowed using a Hanning window, the result from each window is summed then the average of these windows is taken.

5.34 cross_rate.h

```
1 #ifndef CROSS_RATE_H
2 #define CROSS_RATE_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <math.h>
7 #include <memory.h>
8
9 #include "../Feature_Extraction/fft.h"
10 #include "../Feature_Extraction/hanning.h"
11 #include "../Dynamic_Time_Warping/dtw.h"
12
13 int cross_rate(short* signal, int signal_length);
14 float f_cross_rate(float* signal, int signal_length);
15 float stavg_cross_rate(float* signal, int signal_length);
16 float stavg_cross_rate_no_overlap(float* signal, int signal_length);
17 float favg_cross_rate(short* signal, int signal_length);
18
19 #endif
20
```

5.35 d.h

```
1 #ifndef D_H
2 #define D_H
3
4 extern int glbl_banks;
5 extern int glbl_window_width;
6 extern int glbl_paa_op;
7 extern int glbl_paa;
8 extern int glbl_dtw_window;
9 extern int glbl_interval_div;
10
11 #endif

```

5.36 /home/bruh/Documents/GitHub/cleaned_and_commented/↵ Seperation/ste.c File Reference

Functions for calculating the short time energy of a signal.

```
#include "ste.h"
```

Include dependency graph for ste.c:

Functions

- float [short_time_energy](#) (short *signal, int signal_length, int window_length)
Calculates the short time energy of a signal.
- float [f_short_time_energy](#) (float *signal, int signal_length, int window_length)
Calculates the short time energy of a signal; this version is for a float input signal.

5.36.1 Detailed Description

Functions for calculating the short time energy of a signal.

Author

T. Buckingham

Date

Thu May 5 17:59:44 2022

5.36.2 Function Documentation

5.36.2.1 `f_short_time_energy()`

```
float f_short_time_energy (
    float * signal,
    int signal_length,
    int window_length )
```

Calculates the short time energy of a signal; this version is for a float input signal.

Parameters

<i>signal</i>	The signal to process
<i>signal_length</i>	The length of
<i>signal</i>	
<i>window_length</i>	The length of the window to use

Returns

The calculated short time energy

5.36.2.2 `short_time_energy()`

```
float short_time_energy (
    short * signal,
    int signal_length,
    int window_length )
```

Calculates the short time energy of a signal.

Parameters

<i>signal</i>	The signal to process
<i>signal_length</i>	The length of
<i>signal</i>	
<i>window_length</i>	The length of the window to use

Returns

The calculated short time energy

5.37 ste.h

```

1 #ifndef STE_H
2 #define STE_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <memory.h>
7 #include <float.h>
8
9 #include "../Feature_Extraction/hanning.h"
10
11 float short_time_energy(short* signal, int signal_length, int window_length);
12 float f_short_time_energy(float* signal, int signal_length, int window_length);
13
14 #endif

```

5.38 /home/bruh/Documents/GitHub/cleaned_and_commented/Testing/test.c File Reference

Contains the functions for classifying test data with the trained model.

```
#include "test.h"
```

Include dependency graph for test.c:

Data Structures

- struct [guess](#)
- struct [codes](#)

Functions

- void **export_results_pca** (char *ph_code)
- int **test_phoneme_utterance** (short *h, int signal_length)

Tests files without knowing the boundaries, will use the boundary detection to find them.
- void **minimum_edit_distance** (char *res_filename, char *phn_filename)

Determines the minimum edit distance between a reference and result file.
- void **merge_silences** (char *filename)

Merges any contiguous silence classifications into one.
- struct [codes](#) * **get_codes** (char *filename)

- Extracts the labels (start, end and code) from a given file.*

 - void `reset` (void)

Resets all values used for determining accuracy and outputting results.
- void `test_phoneme` (short *h, int signal_length, char *p)

Tests files knowing the boundary.
- void `export_results` (char *ph_code)

Exports the results of a test. Used for one-to-one testing, not boundary testing.
- void `utterance_test` (char *filename, short *sequence, int length, int offset)

Tests a found phoneme using boundary detection.
- float `data_size` (void)

Calculates the size of all MFCCs currently in use.
- float `reduce_data` (void)
- void `bounds_output` (FILE *m_fp)

Outputs the boundary test results to the screen, and the results with which parameters to a file.
- void `matrix_output` (FILE *m_fp)

Outputs the confusion matrix and percentage correct files for use with the given Python scripts.
- void `test_output` (FILE *m_fp)

Outputs the one-to-one testing to the user.
- struct `wav_file` * `trim_silence` (FILE *fp, short *sequence)

Trims the silence data from a given .wav file.
- void `test` (void)

The main control function of the testing process.

Variables

- int ** `matrix`
- float * `per_correct`
- int `group_matrix` [7][7] = {0}
- int `voice_matrix` [3][3] = {0}
- int `sil_v_matrix` [3][3]
- int `sil_corr`
- int * `shorted`
- int * `removed`
- int `correct` = 0
- int `fails` = 0
- int `group` = 0
- int `done` = 0
- int `zero_fails` = 0
- long double `min_edit` = 0
- long double `ins` = 0
- long double `delts` = 0
- long double `subs` = 0
- int `tested_files` = 0
- long double `corr` = 0
- long double `WER` = 0
- long double `refs` = 0
- long double `avg_wer` = 0.0
- long double `low_wer` = 0.0
- long double `high_wer` = 0
- long double `worst`
- long double `total_lengths` = 0
- long double `total_dists` = 0

- long double **total_edits** = 0
- long double **worst_case** = 0
- int **iteration_limit** = 0
- int **current_chunk** = 0
- char * **best_file**
- char * **worst_file**

5.38.1 Detailed Description

Contains the functions for classifying test data with the trained model.

Author

T. Buckingham

Date

Wed May 4 20:20:52 2022

5.38.2 Function Documentation

5.38.2.1 bounds_output()

```
void bounds_output (
    FILE * m_fp )
```

Outputs the boundary test results to the screen, and the results with which parameters to a file.

Parameters

m_{fp}	The file to output the results to.
----------	------------------------------------

5.38.2.2 data_size()

```
float data_size (
    void )
```

Calculates the size of all MFCCs currently in use.

Returns

The current size of all MFCCs

5.38.2.3 export_results()

```
void export_results (
    char * ph_code )
```

Exports the results of a test. Used for one-to-one testing, not boundary testing.

Parameters

<i>ph_code</i>	The code of the phoneme which has been tested.
----------------	--

A result may be exported to correct, group or fail. If the classification is correct then 'correct' will be used, if it is within the same group then 'group' will be used, if neither then 'fail' will be used. This export will contain what the correct phoneme's score was and what the classifications score was at the top with a list of all tested phonemes with their scores below.

5.38.2.4 get_codes()

```
struct codes * get_codes (
    char * filename )
```

Extracts the labels (start, end and code) from a given file.

Parameters

<i>filename</i>	The name of the filename to extract the labels from
-----------------	---

Returns

A codes structure containing a list of phoneme codes in the same order as written in the file.

5.38.2.5 matrix_output()

```
void matrix_output (
    FILE * m_fp )
```

Outputs the confusion matrix and percentage correct files for use with the given Python scripts.

Parameters

<i>m_fp</i>	The file to output the matrix and percentages to.
-------------	---

The percentages correct are defined as the amount correct over the amount observed.

5.38.2.6 merge_silences()

```
void merge_silences (
    char * filename )
```

Merges any contiguous silence classifications into one.

Parameters

<i>filename</i>	The name of the file to find and merge silences
-----------------	---

Any classifications of subsequent silences results in the same outcome for the user but a different WER and so it is decided that as one or more silences acts the same then they can be combined together.

5.38.2.7 minimum_edit_distance()

```
void minimum_edit_distance (
    char * res_filename,
    char * phn_filename )
```

Determines the minimum edit distance between a reference and result file.

Parameters

<i>res_filename</i>	The result file produced by the testing process
<i>phn_filename</i>	The reference file given by the user in the test data

5.38.2.8 reduce_data()

```
float reduce_data (
    void )
```

Removes any sil MFCCs above a certain frame length as they do not vary much among frames, removes coefficients with minimal variance among phonemes and reduces coefficients with low, but not minimal, variance to a short rather than a float to save space.

Returns

The amount of data reduced by the function

5.38.2.9 reset()

```
void reset (
    void )
```

Resets all values used for determining accuracy and outputting results.

This function's primary use is to reset the values when SPLIT_DATA is in use as the accuracy values and confusions matrices should be per test loop

5.38.2.10 test()

```
void test (
    void )
```

The main control function of the testing process.

Determines and sets the test dataset file path, iterates of the files in the dataset file path, produces and outputs the results of the testing process.

5.38.2.11 test_output()

```
void test_output (
    FILE * m_fp )
```

Outputs the one-to-one testing to the user.

Parameters

<i>m_↔_fp</i>	The file to output the results to.
-------------------------	------------------------------------

5.38.2.12 test_phoneme()

```
void test_phoneme (
    short * h,
    int signal_length,
    char * p )
```

Tests files knowing the boundary.

Parameters

<i>h</i>	The signal to be classified
<i>signal_length</i>	The length of
<i>h</i>	
<i>p</i>	The previous phoneme, used for the grammar functions

This is the main one-to-one testing function and can test phonemes using basic DTW, KNN, K-means, etc. A 'complete_signal' can be produced which stores the STE and ZC values for each frame to be used in voice type KNN classification

5.38.2.13 test_phoneme_utterance()

```
int test_phoneme_utterance (
    short * h,
    int signal_length )
```

Tests files without knowing the boundaries, will use the boundary detection to find them.

Parameters

<i>h</i>	The audio signal to test
<i>signal_length</i>	The length of
<i>h</i>	

Returns

The classified phoneme's index

5.38.2.14 trim_silence()

```
struct wav_file * trim_silence (
    FILE * fp,
    short * sequence )
```

Trims the silence data from a given .wav file.

Parameters

<i>fp</i>	The reference file associated with the .wav file
<i>sequence</i>	The associated .wav file in an array

Returns

The new wav structure with silence removed, the size updated and the offset

When using a NOSIL dataset this function returns the structure with the silence removed and provides the offset so that the boundary .res files are correct.

5.38.2.15 utterance_test()

```
void utterance_test (
    char * filename,
    short * sequence,
    int length,
    int offset )
```

Tests a found phoneme using boundary detection.

Parameters

<i>filename</i>	The filename of the reference file
<i>sequence</i>	The found phoneme audio sequence
<i>length</i>	The length of
<i>sequence</i>	
<i>offset</i>	The offset of the file, used when NOSIL is used as the start of testing is no longer the start of the .wav file due to the removal of the leading and trailing silence.

5.39 test.h

```

1 #ifndef TEST_H
2 #define TEST_H
3
4 #include "../Misc/includes.h"
5
6 extern struct Phoneme** phones;
7
8 int test_phoneme_utterance(short* h, int signal_length);
9
10 #endif

```

5.40 test.h

```

1 #ifndef TEST_H
2 #define TEST_H
3
4 #include <math.h>
5 #include "../Clustering/cluster.h"
6 #include "../Dynamic_Time_Warping/dtw.h"
7 #include "../Training/train.h"
8 #include "../Seperation/cross_rate.h"
9 #include "../Seperation/bounds.h"
10 #include "../Clustering/knn.h"
11
12 void export_results(char* ph_code);
13 void export_results_aao(char* ph_code);
14 void test(void);
15 void test_phoneme(short* h, int signal_length, char* p);
16 void test_phoneme_aao(short* h, long start_end[2], char* p);
17 void test_phoneme_pca(short* h, long start_end[2], char* p);
18 float reduce_data(void);
19 float data_size(void);
20 void means(void);
21
22 extern int correct;
23 extern int fails;
24 extern int group;
25 extern int sil_corr;
26 extern int zero_fails;
27
28 extern float* per_correct;
29
30 extern int* shorted;
31 extern int* removed;
32 extern int shted;
33 extern int rmved;
34
35 extern long double refs;
36 extern long double ins;
37 extern long double delts;
38 extern long double subs;
39
40 extern int group_matrix[7][7];
41 extern int voice_matrix[3][3];
42 extern int sil_v_matrix[3][3];
43
44 #endif
45

```

5.41 /home/bruh/Documents/GitHub/cleaned_and_commented/↵ Training/train.c File Reference

#include "train.h"
Include dependency graph for train.c:

Functions

- void **update_zc** (float *signal, int signal_length, int n)

- void **update_ste** (short *signal, int signal_length, int n)
- void **update_sil** (short *signal, int signal_length)
- void **update_sil_db** (short *signal, int signal_length)
- void **update_sil_mean** (short *signal, int signal_length)
- void **update_sil_flat** (short *signal, int signal_length, char *filename)
- void **update_sil_ste_frame** (short *array, int **length**, char *filename)
- int **mfcc_size** (int signal_length)
- int **frame_amount** (int signal_length)
- double **cubic_interpolate** (short y0, short y1, short y2, short y3, double mu)
- long double **ld_cubic_interpolate** (long double y0, long double y1, long double y2, long double y3, long double mu)
- void **train** (void)
is the main control function for training
- short * **resize** (short *shorter, size_t s, size_t l)
- long double * **ld_resize** (long double *shorter, size_t s, size_t l)
- short * **train_ph_mfcc** (int new, short *sequence, struct **Phoneme** *phone)
- void **allocate_ph** (FILE *fp, short *wav, unsigned char t_t)
- struct **wav_file** * **read_wav** (FILE *fp)
- void **update_sil_ste** (short *array, int **length**, char *filename)
- void **update_sil_zc** (short *array, int **length**, char *filename)
- float **flatness** (float *chunk, int **length**)

Variables

- struct **Phoneme** ** **phones**
- char * **p_codes** []
Initlialises all phonemes found in.
- char * **p_group** [] = {"STOP", "AFRI", "FRIC", "NASL", "SEMV", "VOWL", "OTHR"}
- int **OBSTR** = 0
- int **SONOR** = 1
- int **OTHER** = 2
- float * **ph_zc_max**
- float * **ph_zc_min**
- float * **ste_min**
- float * **ste_max**
- int **limit_changed** = 0
- int **num_tt** = 0
- int **num_ph** = 0
- int **max_sil** = 0
used when testing numerous silence detection methods
- int **min_sil** = INT_MAX
- float **max_sil_db** = 0
used when testing numerous silence detection methods
- float **min_sil_db** = INT_MAX
- float **max_sil_zc** = 0
used when testing numerous silence detection methods
- float **min_sil_zc** = INT_MAX
- float **max_sil_ft** = 0
used when testing numerous silence detection methods
- float **min_sil_ft** = INT_MAX
- float **max_sil_ste** = 0
used when testing numerous silence detection methods
- float **min_sil_ste** = INT_MAX
- float **max_sil_mean** = 0
used when testing numerous silence detection methods
- float **min_sil_mean** = INT_MAX

5.41.1 Detailed Description

\Handles training of the phonemes and reading of .wav files

5.41.2 Function Documentation

5.41.2.1 train()

```
train (
    void )
```

is the main control function for training

.PHN file found. These are passed to

If training then the signal is passed to

5.41.3 Variable Documentation

5.41.3.1 max_sil

```
int max_sil = 0
```

used when testing numerous silence detection methods

@max_sil @min_sil are values that store the maximum and minimum raw signal values found in testing

5.41.3.2 max_sil_dB

```
float max_sil_dB = 0
```

used when testing numerous silence detection methods

@max_sil_dB @min_sil_dB are values that store the maximum and minimum raw signal values in decibels found in testing

5.41.3.3 max_sil_flt

```
float max_sil_flt = 0
```

used when testing numerous silence detection methods

@max_sil_flt @min_sil_flt are values that store the maximum and minimum flatness values found in testing

5.41.3.4 max_sil_mean

```
float max_sil_mean = 0
```

used when testing numerous silence detection methods

@max_sil_mean @min_sil_mean are values that store the maximum and minimum mean values found in testing

5.41.3.5 max_sil_ste

```
float max_sil_ste = 0
```

used when testing numerous silence detection methods

@max_sil_ste @min_sil_ste are values that store the maximum and minimum short time energy values found in testing

5.41.3.6 max_sil_zc

```
float max_sil_zc = 0
```

used when testing numerous silence detection methods

@max_sil_zc @min_sil_zc are values that store the maximum and minimum zero cross values found in testing

5.41.3.7 OBSTR

```
int OBSTR = 0
```

@OBSTR @SONOR @OTHER are types of voiceness a phoneme (or sil) can be

5.41.3.8 p_codes

```
char* p_codes[]
```

Initial value:

```
= {"\0", "b", "d", "k", "p", "t", "g",
    "jh", "ch",
    "s", "sh", "th", "v", "f", "dh", "z",
    "m", "n", "ng",
    "l", "r", "hh", "w", "y",
    "aa", "ae", "ah", "aw", "er", "ay", "eh", "ey", "ih", "iy", "ow", "oy", "uh", "uw",
    "sil",
    "\0"}
```

Initialises all phonemes found in.

All phonemes are initialised with the index, string code, MFCC array, raw data array, size array and MFCC count.

5.41.3.9 p_group

```
char* p_group[] = {"STOP", "AFRI", "FRIC", "NASL", "SEMV", "VOWL", "OTHR"}
```

@p_group contains a list of phoneme groups

5.41.3.10 ph_zc_max

float* ph_zc_max

@ph_zc_max @ph_zc_min @ste_min @ste_max All were used to determine if a phoneme should be tested using DTW or KNN if the test's values were not within the range of the phoneme's values then no test would be done

5.42 train.h

```

1 #ifndef TRAIN_H
2 #define TRAIN_H
3
4 #include <dirent.h>
5 #include <unistd.h>
6 #include <stdlib.h>
7 #include <stdio.h>
8 #include <string.h>
9 #include <limits.h>
10 #include <math.h>
11 #include <sys/types.h>
12
13 #include "../Clustering/cluster.h"
14 #include "../Dynamic_Time_Warping/dtw.h"
15 #include "../Misc/realloc.h"
16 #include "../Testing/test.h"
17 #include "../Feature_Extraction/paa.h"
18 #include "../Feature_Extraction/mfcc.h"
19 #include "../Seperation/cross_rate.h"
20 #include "../Seperation/ste.h"
21
22 #define TRAIN 0
23 #define TEST 1
24
25 struct wav_file {
26     short* seq;
27     int length;
28     int offset;
29 };
30
31 double cubic_interpolate(short y0, short y1, short y2, short y3, double mu);
32 void train(void);
33 struct wav_file* read_wav(FILE* fp);
34 void allocate_ph(FILE* fp, short* wav, unsigned char t_t);
35 short* train_ph(int new, short* sequence, struct Phoneme* ph);
36 short* init_new_phone(struct Phoneme* phone, short* sequence, int new);
37 short* resize(short* shorter, size_t s, size_t l);
38 int mfcc_size(int signal_length);
39 int frame_amount(int signal_length);
40 long double* ld_resize(long double* shorter, size_t s, size_t l);
41 void update_sil_zc(short* array, int length, char* filename);
42 void update_sil_ste(short* array, int length, char* filename);
43 float flatness(float* chunk, int length);
44
45 extern struct Phoneme** phones;
46 extern char* p_codes[];
47 extern char* p_group[];
48 extern int num_ph;
49 extern int prev_ph;
50
51 extern int limit_changed;
52
53 extern float* ph_zc_max;
54 extern float* ph_zc_min;
55 extern long double* zc_st_avg_no_oc;
56 extern float* ste_min;
57 extern float* ste_max;
58
59 extern int max_sil;
60 extern int min_sil;
61 extern float max_sil_dB;
62 extern float min_sil_dB;
63 extern float max_sil_zc;
64 extern float min_sil_zc;
65 extern float max_sil_ste;
66 extern float min_sil_ste;
67 extern float max_silflt;
68 extern float min_silflt;
69 extern float max_sil_mean;
70 extern float min_sil_mean;
71
72 #endif

```


Index

/home/bruh/Documents/GitHub/cleaned_and_commented/Clustering41
13 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/KN
/home/bruh/Documents/GitHub/cleaned_and_commented/Clustering13
14 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/MF
/home/bruh/Documents/GitHub/cleaned_and_commented/Clustering41
14 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/Mis
/home/bruh/Documents/GitHub/cleaned_and_commented/Clustering41
14 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/Mis
/home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic 41
15 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/Test
/home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic 50
22 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/rt.h,
/home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic 41
25 /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/bou
/home/bruh/Documents/GitHub/cleaned_and_commented/Dynamic 45
26 /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/bou
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 47
26 /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/cros
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 47
30 /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/cros
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 50
30 /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ctes
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 50
33 /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ste.
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 50
33 /home/bruh/Documents/GitHub/cleaned_and_commented/Seperation/ste.
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 50
36 /home/bruh/Documents/GitHub/cleaned_and_commented/Testing/test.c,
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 50
36 /home/bruh/Documents/GitHub/cleaned_and_commented/Testing/test.h,
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 60
37 /home/bruh/Documents/GitHub/cleaned_and_commented/Training/train.c,
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 60
38 /home/bruh/Documents/GitHub/cleaned_and_commented/Training/train.h,
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 64
41
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 64
41 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/KN
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 64
42 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/KN
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 64
42 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/KN
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 64
43 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/KN
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 64
47 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/KN
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 64
24 /home/bruh/Documents/GitHub/cleaned_and_commented/Real_Time/KN
/home/bruh/Documents/GitHub/cleaned_and_commented/Feature 64
Cluster 8
codes, 8

- create_mfcc
 - dtw.c, 19
- cross_rate
 - cross_rate.c, 48
- cross_rate.c
 - cross_rate, 48
 - favg_cross_rate, 48
 - is_positive, 49
 - stavg_cross_rate, 49
- data_size
 - test.c, 54
- Dataset, 8
- dct
 - fft.c, 30
- delta
 - delta.c, 27
- delta.c
 - delta, 27
 - delta_delta, 28
 - normalise_delta, 28
 - normalise_delta_delta, 28
 - update_delta_deltas_norm, 29
 - update_deltas_norm, 29
- delta_delta
 - delta.c, 28
- dtw.c
 - average_mfccs, 18
 - create_mfcc, 19
 - dtw_frame, 19
 - export_device, 19
 - glbl_paa, 21
 - handle_argv, 20
 - init_dtw_matrix, 20
 - interrupt_handler, 20
 - length, 21
 - mean_size_mfccs, 21
- dtw_frame
 - dtw.c, 19
- export_device
 - dtw.c, 19
- export_results
 - test.c, 54
- f_short_time_energy
 - ste.c, 51
- favg_cross_rate
 - cross_rate.c, 48
- Feature_Set, 9
- fft
 - fft.c, 32
- fft.c
 - dct, 30
 - fft, 32
 - fft_chunks, 32
- fft_chunks
 - fft.c, 32
- get_codes
 - test.c, 55
- get_entropy
 - bounds.c, 45
- glbl_paa
 - dtw.c, 21
- Guess, 9
- guess, 10
- handle_argv
 - dtw.c, 20
- hanning
 - hanning.c, 34
- hanning.c
 - hanning, 34
 - hanning_chunks, 34
 - hanning_chunks_no_overlap, 35
 - hanning_window, 35
- hanning_chunks
 - hanning.c, 34
- hanning_chunks_no_overlap
 - hanning.c, 35
- hanning_window
 - hanning.c, 35
- init_dtw_matrix
 - dtw.c, 20
- interrupt_handler
 - dtw.c, 20
- is_positive
 - cross_rate.c, 49
- KNN, 10
- kurtosis
 - mfcc.c, 38
- length
 - dtw.c, 21
- log_energy
 - mfcc.c, 39
- log_entropy
 - mfcc.c, 39
- matrix_output
 - test.c, 55
- max_mfcc
 - mfcc.c, 41
- max_sil
 - train.c, 62
- max_sil_dB
 - train.c, 62
- max_silflt
 - train.c, 62
- max_sil_mean
 - train.c, 62
- max_sil_ste
 - train.c, 63
- max_sil_zc
 - train.c, 63

mean_size_mfccs
 dtw.c, 21
mel
 mel.c, 37
mel.c
 mel, 37
merge_silences
 test.c, 55
mfcc
 mfcc.c, 40
mfcc.c
 kurtosis, 38
 log_energy, 39
 log_entropy, 39
 max_mfcc, 41
 mfcc, 40
 st_energy, 40
minimum_edit_distance
 test.c, 56
MyApp, 10
MyFrame, 10
MyGrid, 11

next_boundary
 bounds.c, 46
normalise_delta
 delta.c, 28
normalise_delta_delta
 delta.c, 28

OBSTR
 train.c, 63

p_codes
 train.c, 63
p_group
 train.c, 63
Ph_index, 11
ph_zc_max
 train.c, 63
Phoneme, 12

realloc.c
 s_realloc, 43
reduce_data
 test.c, 56
reset
 test.c, 56

s_realloc
 realloc.c, 43
shift_and_reduce
 bounds.c, 46
short_time_energy
 ste.c, 51
st_energy
 mfcc.c, 40
stavg_cross_rate
 cross_rate.c, 49

ste.c
 f_short_time_energy, 51
 short_time_energy, 51

test
 test.c, 56
test.c
 bounds_output, 54
 data_size, 54
 export_results, 54
 get_codes, 55
 matrix_output, 55
 merge_silences, 55
 minimum_edit_distance, 56
 reduce_data, 56
 reset, 56
 test, 56
 test_output, 57
 test_phoneme, 57
 test_phoneme_utterance, 57
 trim_silence, 59
 utterance_test, 59
test_output
 test.c, 57
test_phoneme
 test.c, 57
test_phoneme_utterance
 test.c, 57
train
 train.c, 62
train.c
 max_sil, 62
 max_sil_dB, 62
 max_silflt, 62
 max_sil_mean, 62
 max_sil_ste, 63
 max_sil_zc, 63
 OBSTR, 63
 p_codes, 63
 p_group, 63
 ph_zc_max, 63
 train, 62
trim_silence
 test.c, 59

update_delta_deltas_norm
 delta.c, 29
update_deltas_norm
 delta.c, 29
utterance_test
 test.c, 59

wav_file, 12