TR-I-0021

High-speed Spectrogram Viewer on X-window System

高精度スペクトログラム表示ルーチン

丸山 活輝　村山 浩一　川端 豪

Katsuteru Maruyama,Koichi Murayama and Takeshi Kawabata

1988.2

音声処理ワークベンチ構築の一環として、X windowワークステーション上に高精度スペクトログラム表示ルーチンを開発した。X windowシステムによって書かれたサブルーチン群は、実数型のパワースペクトラムの配列を受取り、Michael Phillips (C.M.U.)考案のアルゴリズムを用いてXのピクセル配列(BITMAP)に変換する。BITMAP ディスプレイ上に高速かつ高精度にスペクトログラムを表示可能で、表示画面破壊時の復旧も素早く行われる。

ATR Interpreting Telephony Research Laboratories
ATR 自動翻訳電話研究所

目　　次

1. 概要

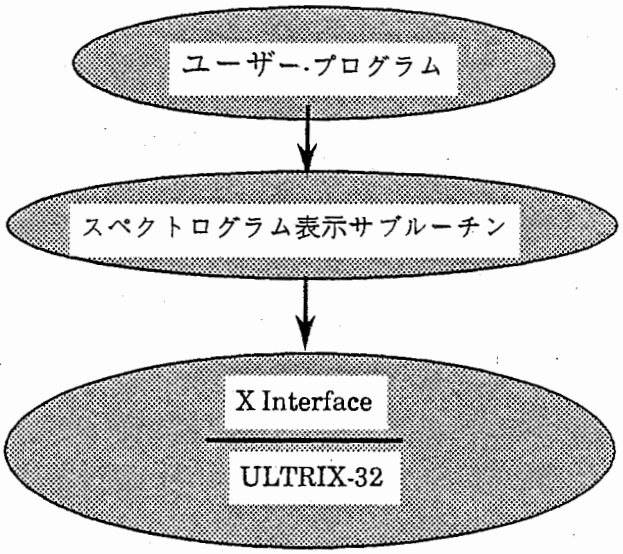
後述するサブルーチン群を使用し、実数型のパワースペクトラムの配列を Michael Phillips (C.M.U.)のアルゴリズムを用いてXのピクセル配列(BITMAP)に変換する。

さらに、その作成された BITMAP をウィンドウに表示しウィンドウのマネージメントを行う。

[動作環境]

OS	言語
ULTRIX-32w (version 1.0) XInterface (Protocol Version 10)	C

[ソフトウェア構成]



2. サブルーチン説明

2.1 SigInit

機能

ピクセルデータを格納する配列のサイズを計算します。

SigCnvFtoPixルーチンで使用する配列(percenton,randarr)に値をセットします。

値のセットの方法は、Michael Phillips のアルゴリズムを用います。

呼び出し形式

```
int SigInit (np, sf, ef, data__xpixels,
             data__ypixels, percenton, randarr)
```

リターン値

正常　0

異常　-1

引数

(1) np

type　: long integer

access　: readonly

mechanism: by value

1フレーム内のエレメント数

(2) sf

type　: long integer

access　: readonly

mechanism: by value

スタートフレーム番号

(3) ef

type : long integer

access : readonly

mechanism: by value

エンドフレーム番号

(4) data__xpixels

type : long integer

access : readonly

mechanism: by value

1エレメントに対応するX方向のピクセル数。

(5) data__ypixels

type : long integer

access : readonly

mechanism: by value

1エレメントに対応するY方向のピクセル数。

(6) bits__x

type : long integer

access : writeonly

mechanism: by address

ピクセルデータ格納配列のX方向のサイズ

(7) bits__y

type : long integer

access : writeonly

mechanism: by address

ピクセルデータ格納配列のY方向のサイズ

(8) percenton

type　　: long integer array

access　　: writeonly

mechanism: by address

SigCnvFtoPixルーチンで使用する配列。

サイズは、MAXPERCENTON。(MAXPERCENTONは、sigdef.hで定義。)

(9) randarr

type　　: long integer array

access　　: writeonly

mechanism: by address

SigCnvFtoPixルーチンで使用する配列。

サイズは、MAXSIZE + 1。(MAXSIZEは、sigdef.hで定義。)

2.2 SigAlloc

機能

ピクセルデータを格納する領域を確保する。

呼び出し形式

short *area

area = SigAlloc (bits__x, bits__y)

リターン値

正常　ポインターのアドレス

異常　NULL

引数

(1) bits__x

type : long integer

access : readonly

mechanism: by value

ピクセルデータ格納配列のX方向のサイズ

(2) bits__y

type : long integer

access : readonly

mechanism: by value

ピクセルデータ格納配列のY方向のサイズ

2.3 SigFree

機能

ピクセルデータを格納していた領域を解放する。

呼び出し形式

(void) SigFree(area)

引数

(1) area

type　　: short integer pointer

access　　: readonly

mechanism: by address

ピクセルデータを格納していた領域のポインター

2.4 SigCnvFtoPix

機能

実数型のパワースペクトラムの配列を Michael Phillipsのアルゴリズムを用いて BITMAP に変換する。

呼び出し形式

```
int SigCnvFtoPix (spectrum, n, m, np, sf, ef,
                  data__xpixels, data__ypixels,
                  spect, bits__x, bits__y,
                  percenton, randarr)
```

リターン値

正常　0

異常　-1

引数

(1) spectrum

type　: float array

access　: readonly

mechanism: by address

パワースペクトラムのデータを格納している配列。

(2) n

type　: long integer

access　: readonly

mechanism: by value

パワースペクトラムのデータを格納している配列の全フレーム数。

(3) m

type : long integer

access : readonly

mechanism: by value

パワースペクトラムのデータを格納している配列の全エレメント数。

(4) np

type : long integer

access : readonly

mechanism: by value

1フレーム内のエレメント数。

(5) sf

type : long integer

access : readonly

mechanism: by value

スタートフレーム番号。

(6) ef

type : long integer

access : readonly

mechanism: by value

エンドフレーム番号。

(7) data__xpixels

type : long integer

access : readonly

mechanism: by value

1エレメントに対応するX方向のピクセル数。

(8) data__ypixel type : long integer

access : readonly

mechanism: by value

1エレメントに対応するY方向のピクセル数。

(9) spect

type : short integer pointer

access : writeonly

mechanism: by address

BITMAP データを格納する領域のポインター。

(10) bits__x

type : long integer

access : readonly

mechanism: by value

ピクセルデータ格納配列のX方向のサイズ。

(11) bits__y

type : long integer

access : readonly

mechanism: by value

ピクセルデータ格納配列のY方向のサイズ。

(12) percenton

type　　: long integer array

access　　: readonly

mechanism: by address

データ変換の際、使用する配列。

値は、SigInitルーチンでセットする。

(13) randarr

type　　: long integer array

access　　: readonly

mechanism: by address

データ変換の際、使用する配列。

値は、SigInitルーチンでセットする。

2.5 SigPutSpectrogram

機能

BITMAP をウィンドウに表示する。

別のウィンドウによって壊された領域は、自動的に復旧される。

画像の左右の移動は、マウスのミドル･ボタンが押された位置を中央にして移動させる。

上下の移動はない。

呼び出し形式

```
int SigPutSpectrogram
        (spect, bits__x, bits__y, np,

        topleft__x, topleft__y, topright__x, topright__y,

        sampling__rate, st, sf, d,

        data__xpixels, data__ypixels)
```

リターン値

正常　0

異常　-1

引数

(1) spect

type　　: short integer pointer

access　　: readonly

mechanism: by address

BITMAP データを格納している領域のポインター。

データの格納は SigCnvFtoPix ルーチンで行う。

(2) bits__x

type : long integer

access : readonly

mechanism: by value

BITMAP データを格納している配列のX方向のサイズ。

(3) bits__y

type : long integer

access :reado

mechanism: by value

BITMAP データを格納している配列のY方向のサイズ。

(4) np

type : long integer

access : readonly

mechanism: by value

1フレーム内のエレメント数。

(5) topleft__x

type : long integer

access : readonly

mechanism: by value

スペクトログラム表示ウィンドウの左上隅の相対X座標。

(6) topleft__y

type : long integer

access : readonly

mechanism: by value

スペクトログラム表示ウィンドウの左上隅の相対Y座標。

(7) topright__x

type : long integer

access : readonly

mechanism: by value

スペクトログラム表示ウィンドウの右上隅の相対X座標。

(8) topright__y

type : long integer

access : readonly

mechanism: by value

スペクトログラム表示ウィンドウの右上隅の相対Y座標。

(9) samplingrate

type : long integer

access : readonly

mechanism: by value

最大サンプリング周波数。

周波数目盛の最大値は、samplingrate / 2 となる。

(10) st

type : long integer

access : readonly

mechanism: by value

サンプリングスタート時間。(単位 μS)

(11) sf

type : long integer

access : readonly

mechanism: by value

サンプリングシフト時間。(単位 μS)

(フレームとフレームの間の時間)

(12) d

type : long integer

access : readonly

mechanism: by value

時間目盛の数字表示間隔。。(単位 μS)

(13) data__xpixels

type : long integer

access : readonly

mechanism: by value

1エレメントに対応するX方向のピクセル数。

(14) data__ypixels

type : long integer

access : readonly

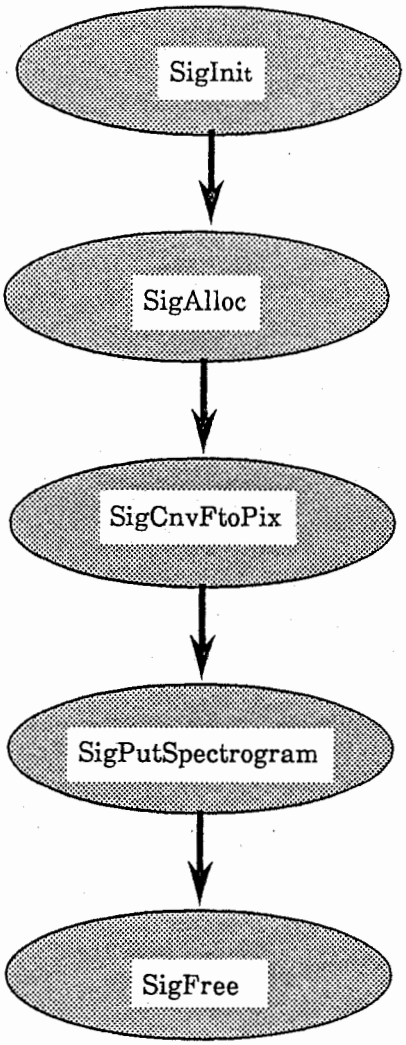
mechanism: by value

1エレメントに対応するY方向のピクセル数。

3. プログラムの構成の仕方

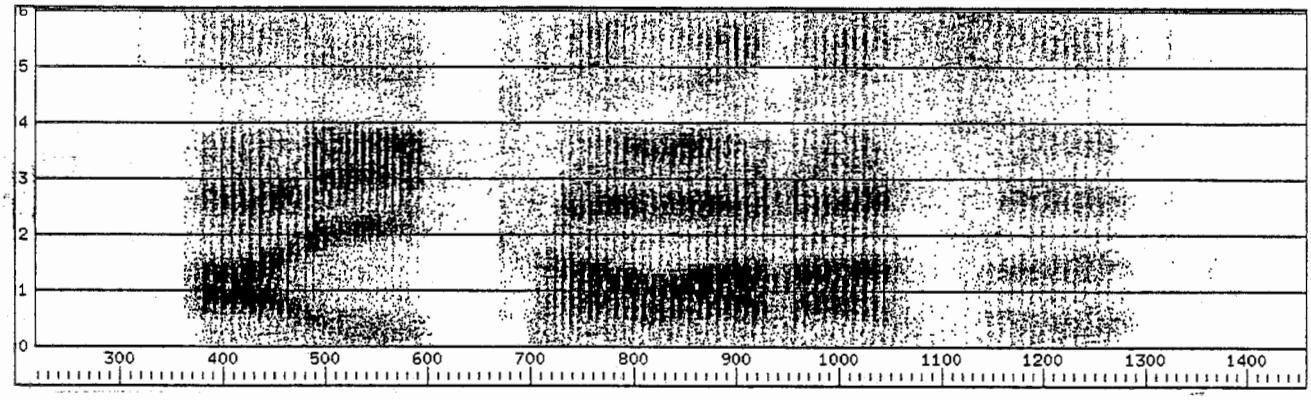
3.1. コール方法

次の図に示す順にサブルーチンをコールする。

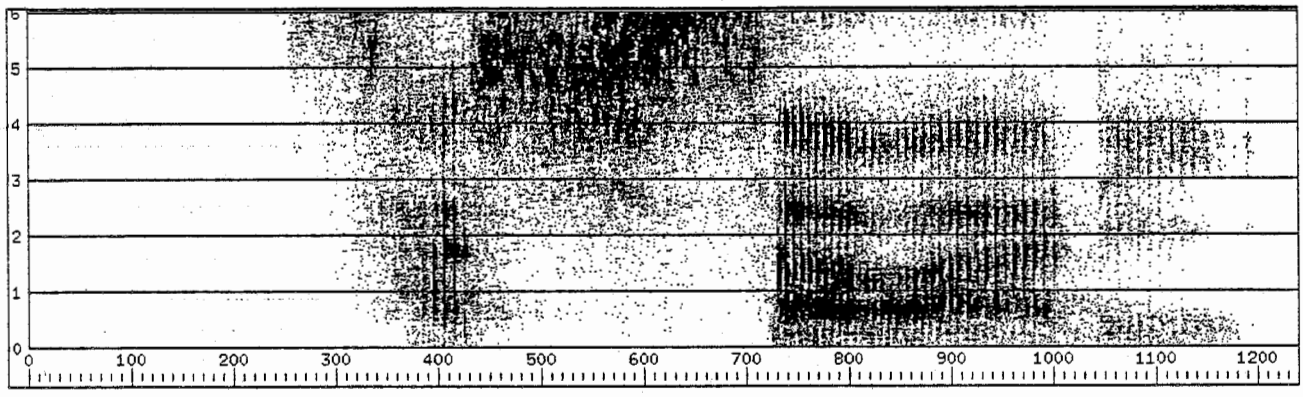


4. 出力例

4.1.アイカワラズ



4.2.サシサワリ



APPENDIX

A-1.対数パワースペクトラム計算プログラム

本プログラムにより、音声波形データから実数型の対数パワースペクトラムを計算し、結果をディスク上に保存する。

```
#include   <stdio.h>
#include   <fcntl.h>
#include   <ctype.h>
#include   <sys/types.h>
#include   <sys/stat.h>

#define    FLENGTH     60      /* in point ( 5 msec )    */
#define    FSHIFT      30      /* in point ( 2.5 msec ) */
#define    FFTLENGTH   128     /* fft size               */
#define    FFTPOW      7       /* 2**FFTPOW = FFTLENGTH  */
#define    MU          0.97    /* preemph coef           */
#define    PMODE       0644

extern int errno;              /* global */

main(argc,argv)
int argc;
char    *argv[];

{
  struct stat    file_stat;

  char  wave_file[256], spec_file[256];
  int        i, j, fw, fo;
  int        nod, noffrs, nofpnt;
  static float    real[FFTLENGTH], imag[FFTLENGTH];
  short *wave;
  float  *pree;
  float  *falloc();
  short *salloc();
  FILE     *fp, *fopen();

  nod = FFTLENGTH/2+1;
```

```
  /* open file and read data */

  if(( fw = open(argv[1], O_RDONLY) ) < 0 ) {
    perror( "Wave File open error" ) ;
    exit(0) ;
  }

  if(( fo = open(argv[2], O_WRONLY | O_TRUNC | O_CREAT, PMODE) ) < 0 ) {
    perror( "Output file open error" ) ;
    exit(0) ;
  }

  if ( ( stat(argv[1], &file_stat) ) < 0) {
    perror( "Wave data format error" ) ;
    exit(0) ;
  }
  nofpnt = file_stat.st_size / sizeof(short) ;
  noffrs = frame( nofpnt, FLENGTH, FSHIFT ) ;

  if ((wave = (short *)malloc(nofpnt * sizeof(short)) ) == NULL) {
    perror(" Memory allocate error") ;
    exit(0) ;
  }

  if ((pree = (float *)malloc(nofpnt * sizeof(float)) ) == NULL) {
    perror(" Memory allocate error") ;
    exit(0) ;
  }


  /* Open Wave file and read data */
  if ( read(fw,wave,file_stat.st_size) < 0 ) {
    perror( "Wave File read error" ) ;
    exit(0) ;
  }

  preemph ( wave, pree, nofpnt, MU ) ;
  free( wave ) ;

  for ( j=0; j<noffrs; j++ ) {
    printf(" -- %d/%d --%c[A¥n",(j+1),noffrs,'¥033') ;
    logpower(pree, j, real, imag) ;
    write( fo, real, nod*sizeof(float) ) ;
  }

  close( fo ) ;
  close( fw ) ;
}
```

```
logpower(pree, frame, real, imag)
float    *pree, *real, *imag ;
int frame ;

{
    int i, nod ;
    float    a ;
    double  log10() ;

    nod = FFTLENGTH/2+1 ;

    for ( i = 0 ; i < FFTLENGTH ; i++ ) {
        real[i] = 0. ;
        imag[i] = 0. ;
    }
    for ( i = 0 ; i < FLENGTH ; i++ )
        real[i] = pree[frame*FSHIFT+i] ;/* single precision */
    hamu ( real , FLENGTH ) ;
    sfft ( real , imag , FFTLENGTH , FFTPOW , -1 ) ;
    for ( i = 0 ; i < nod ; i++ ) {
        a = real[i]*real[i] + imag[i]*imag[i] ;
        if( a < 1.00e-30 )    a = 1.00e-30 ;
        real[i] = 10. * log10(a) ;
    }
}


frame ( ds , l , s )
int ds , l , s;

{
    int n ;

    n = 0;
    while ( ( l-1+n*s ) <= ( ds - 1 ) )
        n++;
    return ( n );
}
```

```
preemph ( wave, pree, n, mu )
short   *wave ;
float   *pree , mu ;
int n ;

{
    static  float ;
    int i;

    pree[0] = (float)wave[0] ;
    for ( i = 1 ; i < n ; i++ )
        pree[i] = (float)wave[i] - mu * (float)wave[i-1] ;
}
```

A-2. スペクトログラム表示サンプルプログラム

以下にスペクトログラム表示ルーチンをコールするメインプログラムを紹介する。

```
/* サンプルプログラム */

#include    <stdio.h>
#include    <sys/file.h>
#include    <sys/types.h>
#include    <sys/stat.h>

/* スペクトラム表示ルーチンのためのinclude file */
#include    "sigdef.h"

#define     DATA_XPIXELS    8       /* 1エレメントに対するX方向のピクセル数 */
#define     DATA_YPIXELS    8       /* 1エレメントに対するY方向のピクセル数 */
#define     TOPLEFT_X       0       /* ウインドゥ左上隅X座標 */
#define     TOPLEFT_Y       10      /* ウインドゥ左上隅Y座標 */
#define     TOPRIGHT_X      1000    /* ウインドゥ右上隅X座標 */
#define     TOPRIGHT_Y      10      /* ウインドゥ右上隅Y座標 */
#define     SAMPLINGRATE    12      /* サンプリング周波数 (kHz) */
#define     ST              0       /* サンプリングスタート時間 (mS) */
#define     SF              1       /* サンプリングシフト時間 (mS) */
#define     D               10      /* 時間軸数字表示間隔 (mS) */
#define     DATA_SIZE       65

main    (argc,argv)
long    argc;
char    *argv[];        /* データファイル名を得る */
{
  int       i, j, k;        /* 添字 */
  float *spectrum;          /* スペクトラムのデータ */
  short *spect;             /* ピクセルデータ */
  long  bits_x,             /* ピクセルデータ X方向サイズ */
        bits_y;             /* ピクセルデータ Y方向サイズ */
  long  percenton[MAXPERCENT+1];
  long  randarr[MAXSIZE];   /* 乱数表 */
  long  fp;
  struct stat   file_stat;
  long  n;                  /* 全フレーム数 */
  long  sf,                 /* スタートフレーム番号 */
        ef;                 /* エンドフレーム番号 */
```

```
/* データファイルのサイズを調べる */
  if ( (stat   (argv[1], &file__stat) ) < 0)
    perror  ("Stat Failure");

/* ファイルサイズをレコード長で割り全フレーム数を得る */
  n = file__stat.st__size / ((DATA__SIZE + FILLER__SIZE) * sizeof(float));

/* スペクトラムデータ配列の領域を確保する */
  spectrum = (float *)calloc    (n*DATA__SIZE+FILLER__SIZE, sizeof(float) );

/* データファイルをオープンする */
  if ( (fp= open (argv[1], O__RDONLY) ) < 0)
    perror  ("Open Failure");

/* ピクセル配列のサイズと乱数表を得る */
  sf= 0;
  ef= n - 1;
  if ( (SigInit   (DATA__SIZE,
            sf,
            ef,
            DATA__XPIXELS,
            DATA__YPIXELS,
            &bits__x,
            &bits__y,
            percenton,
            randarr) ) < 0)
    perror  ("Fail SigInit");

/* ピクセル配列の領域を確保する */
  if ( (spect = SigAlloc(bits__x,bits__y) ) == NULL)
    perror  ("Fail : Cannot allocate space for spect");

/* データをスペクトラム配列に取り込む */
  j  = 0;
  for    (i = 0; i < n; i++) {
    if   ( (read  (fp,
            &spectrum[j],
            DATA__SIZE*sizeof(float) ) ) < 0)
        perror  ("Read Error");
    j += DATA__SIZE;
  }

/* データファイルをクローズする */
  close  (fp);
```

```
/* スペクトラムデータをピクセルデータに変換する */
  if((SigCnvFtoPix    (spectrum,
            n,
            DATA_SIZE,
            DATA_SIZE,
            sf,
            ef,
            DATA_XPIXELS,
            DATA_YPIXELS,
            spect,
            bits_x,
            bits_y,
            percenton,
            randarr)) < 0)
    perror ("ERROR!!");

/* ピクセルデータをウインドゥに表示する */
  if((SigPutSpectrogram  (spect,
                bits_x,
                bits_y,
                DATA_SIZE,
                TOPLEFT_X,
                TOPLEFT_Y,
                TOPRIGH          SAMPLINGRATE,
                ST,
                SF,
                D,
                DATA_XPIXELS,
                DATA_YPIXELS)) < 0)
    perror ("ERROR!!");

/* ピクセル配列の領域を解放する */
  SigFree  (spect);

}
```