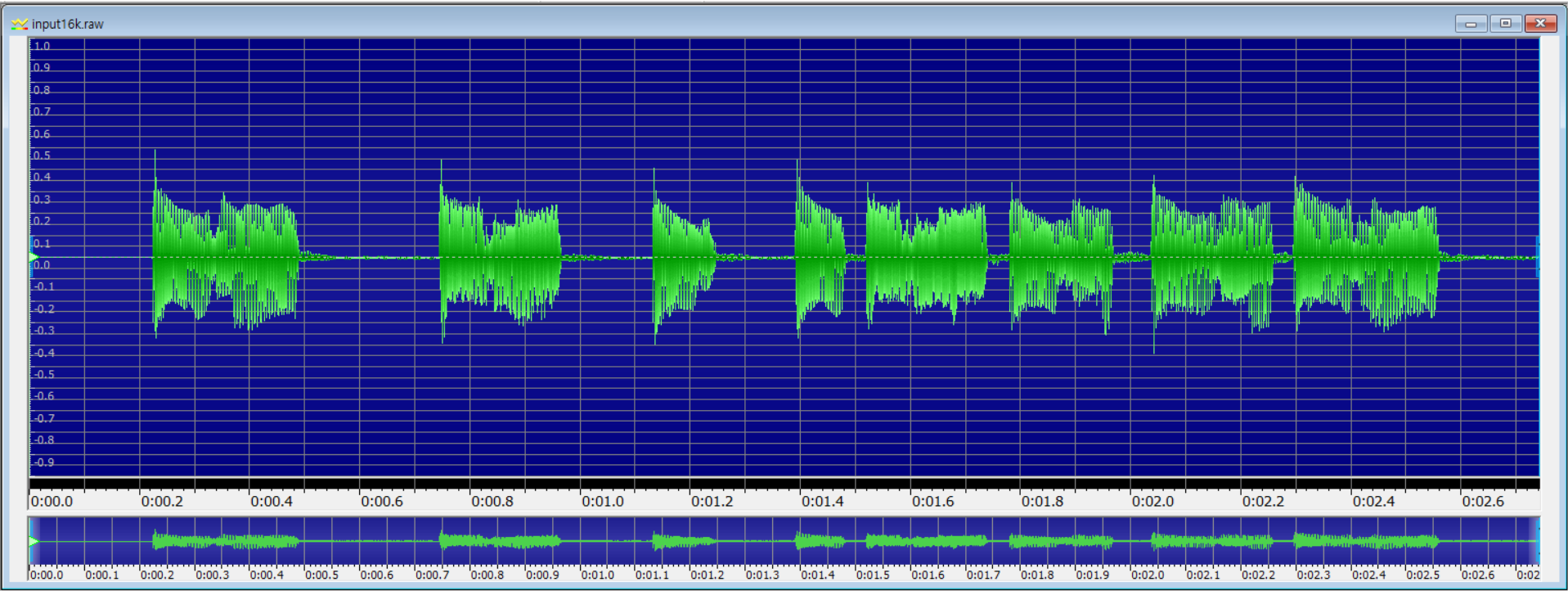
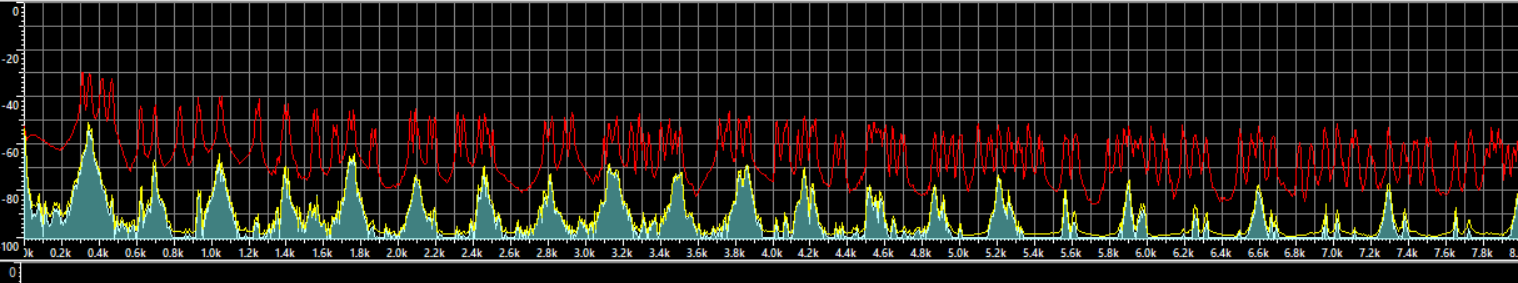
**인공지능과 음성신호처리 HW1**

전자통신공학과 2014707073 김수환

1. 음성 파일 확인 (Gold Wave)



:: 음성 파형



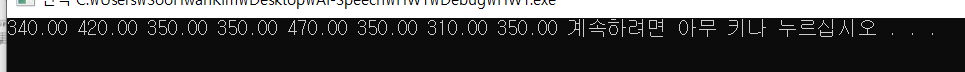
:: 스펙트럼

2. 주파수 (계이름)

- 주파수 & 계이름

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **계이름** | 파 | 솔# | 파 | 파 | 라# | 파 | 레# | 파 |
| **주파수** | 340 | 420 | 350 | 350 | 470 | 350 | 310 | 350 |

* 실행 결과



- 실행 환경

Window 10, Visual Studio 2019 (c언어)

2. 소스코드

- main.c

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | #include "signal.h"    #define N\_POINT           1600  #define N\_EXG             8  #define N\_READ            1  #define SAMPLE\_RATE      16000    FILE\* fin;  float comp\_freq(int index);  int skip[N\_EXG] = {      SAMPLE\_RATE \* 0.28,      SAMPLE\_RATE \* 0.4,      SAMPLE\_RATE \* 0.22,      SAMPLE\_RATE \* 0.16,      SAMPLE\_RATE \* 0.1,      SAMPLE\_RATE \* 0.07,      SAMPLE\_RATE \* 0.17,      SAMPLE\_RATE \* 0.18  };      int main() {      float note\_freq[N\_EXG] = { 0.0, };        fopen\_s(&fin, "input16k.raw", "rb");        for (int i = 0; i < N\_EXG; i++){          note\_freq[i] = comp\_freq(i);          printf("%.2f ", note\_freq[i]);      }        system("pause");      fclose(fin);      return 0;  }      float comp\_freq(int index) {      float signal[N\_POINT];      float spec\_real[N\_POINT], spec\_imag[N\_POINT], spec\_magn[N\_POINT];      float freq = 0.0;      short data = 0;      int k = 0;        /\* Skip to the start \*/      for (int n = 0; n < skip[index]; n++)          fread(&data, SHORT\_SIZE, N\_READ, fin);        for(int n = 0; n < N\_POINT; n++){          fread(&data, SHORT\_SIZE, N\_READ, fin);          signal[n] = (float)data;      }        /\* N point Discrete Fourier Transform \*/      dft(signal, spec\_real, spec\_imag, spec\_magn, N\_POINT, "hamming");      k = argmax\_magnitude(spec\_magn, N\_POINT);      freq = convert\_hertz(k, SAMPLE\_RATE, N\_POINT);        return freq;  } | [cs](http://colorscripter.com/info#e) | |

* signal.h

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | #include <stdio.h>  #include <stdlib.h>  #include <math.h>    #define SHORT\_SIZE     2  #define PI             3.14    float\* get\_window(char\* win\_type, int window\_size);  float\* windowing(float\* signal, float\* window, int window\_size);  float convert\_hertz(int k, int sr, int N\_POINT);  int dft(float signal[], float spec\_real[], float spec\_imag[], float spec\_magn[], int N\_POINT, char\* win\_type);  int argmax\_magnitude(float spec\_magn[], int N\_POINT);      float\* get\_window(char\* win\_type, int window\_size) {      float\* window = (float\*)malloc(sizeof(float) \* window\_size);        if (!strcmp(win\_type, "hamming")) {          for (int n = 0; n < window\_size; n++)              window[n] = 0.54 - 0.46 \* cos(2 \* PI \* n / (window\_size - 1));      }      else if (!strcmp(win\_type, "hanning")) {          for (int n = 0; n < window\_size; n++)              window[n] = 0.5 - 0.5 \* cos(2 \* PI \* n / (window\_size - 1));      }      else if (!strcmp(win\_type, "rectangular")) {          for (int n = 0; n < window\_size; n++)              window[n] = 1.0;      }      else if (!strcmp(win\_type, "blackman")) {          for (int n = 0; n < window\_size; n++)              window[n] = 0.42 - 0.5 \* cos(2 \* PI \* n / (window\_size - 1)) + 0.08 \* cos(4 \* PI \* n / (window\_size - 1));      }      else if (!strcmp(win\_type, "sine")) {          for (int n = 0; n < window\_size; n++)              window[n] = sin(PI \* (n + 0.5) / window\_size);      }      else {          printf("%s is not Supported\n", win\_type);      }        return window;  }      float\* windowing(float\* signal, float\* window, int window\_size) {      for (int n = 0; n < window\_size; n++)          signal[n] \*= window[n];        return signal;  }      int dft(float signal[], float spec\_real[], float spec\_imag[], float spec\_magn[], int N\_POINT, char\* win\_type) {      float\* window = NULL;      float\* windowed = NULL;        window = get\_window(win\_type, N\_POINT);      windowed = windowing(signal, window, N\_POINT);        for (int k = 0; k < N\_POINT; k++) {          spec\_real[k] = 0.0;          spec\_imag[k] = 0.0;            for (int n = 0; n < N\_POINT; n++) {              spec\_real[k] += signal[n] \* cos(2 \* PI \* k \* n / N\_POINT);              spec\_imag[k] -= signal[n] \* sin(2 \* PI \* k \* n / N\_POINT);          }          spec\_magn[k] = pow(spec\_real[k], 2) + pow(spec\_imag[k], 2);          spec\_magn[k] = sqrt(spec\_magn[k]);      }        free(window);      return 0;  }      int argmax\_magnitude(float spec\_magn[], int N\_POINT) {      float max\_magn = 0.0;      int argmax = 0;        for (int k = 0; k < N\_POINT; k++) {          if (max\_magn < spec\_magn[k]) {              max\_magn = spec\_magn[k];              argmax = k;          }      }        return argmax;  }      float convert\_hertz(int k, int sr, int N\_POINT) {      float spec\_resolution = (float)sr / (float)N\_POINT;        return (float)k \* spec\_resolution;  } | [cs](http://colorscripter.com/info#e) | |

3. 고찰

|  |
| --- |
| 저는 전자통신공학과 4학년 학생이며, 현재 서강대학교 청각지능연구실에서 학부연구생을 하고 있습니다 내년 봄 학기에 해당 연구실로 석사 진학 예정에 있으며 현재 캡스톤 팀원들과 ‘한국어 음성 인식’을 주제로 프로젝트 진행중에 있습니다. 공부와 프로젝트를 진행하며 음성 신호에 대해서 잘 알아야 된다고 많이 느꼈고, 지난 2-3학년까지 소프트웨어 과목들을 위주로 수강했기에 저 스스로 신호처리 분야 실력이 많이 부족함을 느꼈습니다. 그래서 부족한 실력을 채우기 위해 본 수업을 신청하여 수강하고 있습니다. ‘한국어 음성 인식’ 프로젝트를 진행하며 Mel-Spectrogram, MFCC 등에 대하여 공부하며 어떤 프로세스를 거치는지까지는 공부했지만, 왜 그러한 프로세스를 거쳐야 하는지에 대해서는 알지 못했습니다. 본 강의를 들으면서 조금씩 알아가며 신호 처리 분야에 재미를 붙이고 있습니다. 그래서 이번 과제가 나왔을 때 자신있게, 재미있겠다고 생각했습니다만, 실제로 적용하려니 꽤나 막막하고 제가 이해했다고 생각한 내용들이 머리 속에서 뒤엉켰습니다. 다시 개념을 재정립하고 나서야 숙제를 마칠 수 있었습니다. 이번 숙제를 하면서 신호 처리 개념을 실제로 구현해보는 것도 굉장히 재미있다고 느꼈습니다. 그래서 교수님의 수업을 듣고 여름방학 때 짬짬히 MFCC 피쳐 추출 과정을 c언어를 이용해서 직접 하나하나 구현해보는 목표가 생겼습니다. 이와 비슷한 숙제들이 나온다면 재밌게 해볼 수 있을 것 같습니다. |