논문 2017-4조

wav to midi converting algorithm을 이용한 midifile 생성 시스템

(Midifile creating system using wav to midi converting algorithm)

최윤정, 김동영, 강제원 교수님

(Yoonjung Choi, Dongyoung Kim, Jaewon Kang)

요 약

본 설계는 시중에 사용되어지고 있는 MIDI converter 보다 더 정확한 midi 파일의 형식을 구현하는 것을 목적으로 한다. 이를 위해 우리는 MATLAB을 이용하여 Audio 파일을 입력받고 기존 Audio to MIDI converter 에서 제공하는 MIDI file보다 개선된 알고리즘을 이용하여 출력된 MIDI 파일의 효율성과 성능을 높였다. 우리는 Harmonics를 거의 그대로 출력하는 기존 Audio to MIDI converter와는 달리 불필요한 Harmonics들을 제거하는 'Dynamic window algorithm' 라는 독창적인 방법을 개발하여 보다 명확한 형태의 MIDI file을 얻을 수 있도록 시스템을 고안하였다. 이 프로젝트를 통해 기존의 Audio to MIDI converter보다 더 정확한 MIDI file을 얻을 수 있을 것으로 기대된다.

Abstract

In this project, We purpose to implement an MIDI file creating system using wav to MIDI converting algorithm which applied harmonic eliminating method. For this, We use the Dynamic window algorithm for getting the small energy of fundamental frequency and upgrade the existing converting system using previous method of MIDI making algorithm in audio to MIDI converting system. So we are looking forward to achieving a high accuracy in the result of MIDI file in that eradicating useless harmonics. We implement audio to MIDI converting system in a way that STFT in MATLAB checked many times to extract meaningful sound. Through this, it is expected that the proposed system can support the previous audio to MIDI converting system.

Keywords : Audio to MIDI converting , MATLAB algorithm , Audio signal processing

Ⅰ. 서 론

'audio to MIDI converting' 은 MIDI note number와 주파수 값 사이의 일련의 matching을 통하여 파형으로 구성된 wav file을 MIDI file로 변환하는 방법이다. MIDI signal에서 보여 지는 - 기존의 audio signal에서 들리지 않던 필요 없는 harmonics들을 제거함으로써 음악의 편집, 혹은 악보를 제작하는 부분에서 아주 유용하게 쓰일 수 있는 기술이다. 따라서 [1]악보제작, [2]음악파일을 기반으로 한 게임 소프트웨어제작 등 활용분야가 무궁무진 하다. audio to midi converting 기술은 이용자들에게 거부감 없이 자연스럽고 정확한 midi file을 얻을 수 있게 하는 방법이며 절대음감을 가진 전공자의 청음과 사보가 필요치 않아 비용 측면에서도 큰 이득을 가져다줄 수 있다.

audio processing system은 현재까지 계속 사용되어왔던 분야로 audio to MIDI converting system은 <그림1>과 같이 세 단계로 이루어진다.

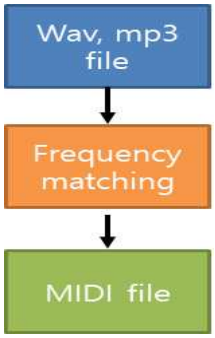


그림 1 : audio file을 MIDI file로 변환하는 과정

1) Audio file read : wav file을 읽고 음정들의 주파수 값과 그 외 MIDI file을 작성하기 위해 필요한 값들을 저장한다.
2) Frequency matching : 주파수 값에 따라 적절한 MIDI file note number를 Mapping한다.
3) MIDI file write : 저장한 값들을 이용해 MIDI file을 작성한다.

위의 순서를 토대로 많은 audio to midi converting system이 존재한다. 기존의 converting system을 이용하면 쉽고 간단하게 midi file을 추출 해낼 수 있다는 장점이 있어 현재까지 널리 사용되고 있다. 그러나 기존의 Converter들은 Frequency matching 과정에서 원음이 아닌 Harmonics까지 원음으로 인식하여 정확한 data를 얻지 못한다는 한계를 지닌다.

기존의 Converting system에서 문제가 되는 Harmonics란 음의 높이가 아닌 음색을 나타내기 위해 존재하는 주파수 성분이다. Frequency matching 과정에서 사용되는 Fundamental frequency의 정수 배의 주파수를 가지며, 주파수 도메인에서 Fundamental frequency보다 크기가 작다는 특징을 가지고 있다.

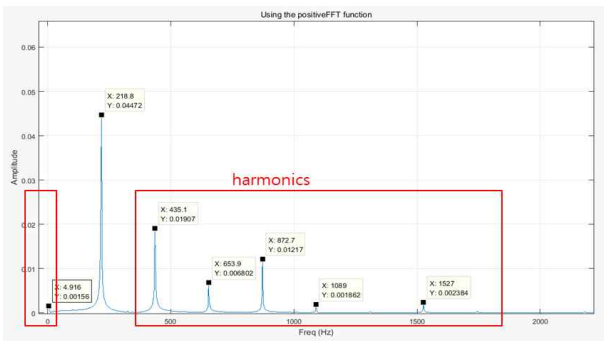


그림2 : A3(220Hz)의 FFT plot

<그림 2>는 Fundamental frequency가 220Hz인 '3옥타브 라' 를 FFT한 결과 영상이다. 한 개의 음정이지만, 여러 개의 주파수 성분들을 가지고 있다. Fundamental frequency인 220Hz 성분은 가장 크기가 크고 나머지 주파수 성분들(Harmonics)은 220Hz 성분보다는 크기가 작으며, 정수배의 주파수임을 알 수 있다.

하나의 노래 속에는 수많은 음정이 존재한다. 음의 세기는 일정하지 않으며, harmonics들의 크기와 fundamental frequency 크기도 제각각이기 때문에 노래 속에서 단순히 크기만으로 Harmonics와 fundamental frequency를 구별하기는 다소 어렵다.

사용자가 별도의 Harmonics 제거 알고리즘을 따로 적용하지 않고 Audio file을 MIDI file로 변환할 시 제대로 된 MIDI file을 얻을 수 없게 되며 비 전공자의 경우 Harmonics까지 필요한 음들이라고 잘못 인식하게 된다. 우리 조는 이와 같은 단점을 개선하는 Harmonics 제거 알고리즘을 고안하여 사용자들이 정확한 MIDI file을 얻을 수 있게 도와주고자 한다.

II. 본 론

1. 환경 구축

가. MATLAB을 통한 Audio 신호 처리

MATLAB으로 Audio file Data를 얻는다. Audio 신호의 시간에 따른 주파수 변화를 분석해야 하므로 STFT(short time fourier transform)처리한다. STFT란 MATLAB 인터페이스에서 시간에 따라 변화하는 주파수를 나타내고자 할 때 많이 사용되는 함수이다.

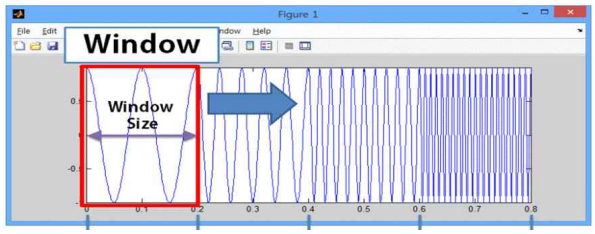


그림3 : STFT 수행과정

<그림 3>과 같이 전체 신호에 대하여 작은 크기의 window를 적용하고 Window를 시간에 따라 움직이면

서 여러 번 FFT를 수행한다. STFT 결과로 신호의 시간에 따른 주파수 변화를 알 수 있다. 이때, time resolution과 frequency는 trade-off 관계이므로 Parameter를 적절하게 조정하여 최적의 STFT결과를 얻는 것이 중요하다.

아래 <그림4>는 STFT 결과를 contour 함수를 이용해 등고선 모양으로 나타낸 Plot이다. 파란 점 하나는 음정 하나를 나타낸다.

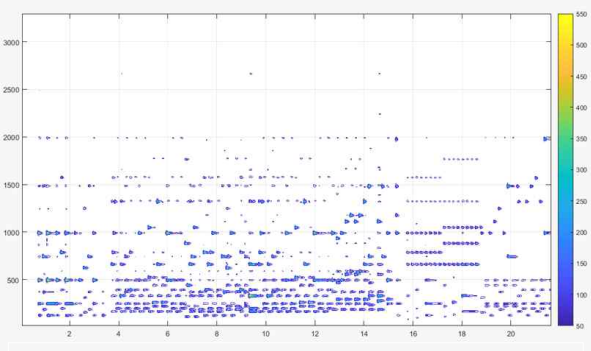


그림4 : STFT 결과 plot

나. 엑셀을 이용한 Frequency & MIDI note numer 정보 저장

<그림 5>는 Frequency와 MIDI note number 사이 관계를 나타낸다. Frequency - MIDI note number가 서로 매칭되어있는 DATA table을 엑셀에 작성한다. STFT 결과 값과 DATA table을 비교하여 적절한 MIDI note number 값 알아낸다. 결과값은 MIDI file의 한 구성 요소로 사용된다.

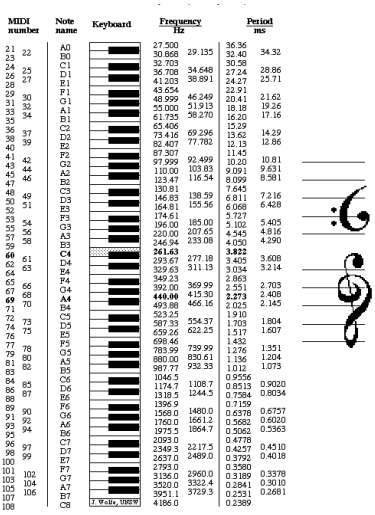


그림 5 : Frequecy - MIDI note number matching 정보

2. midi file 생성 방법

가. MIDI file의 구성

Audio file을 MIDI file로 만들기 위해서는 Audio file에서 적절한 Data를 가져오고 이를 토대로 MIDI file format에 맞춰 MIDI file을 작성해야 한다.

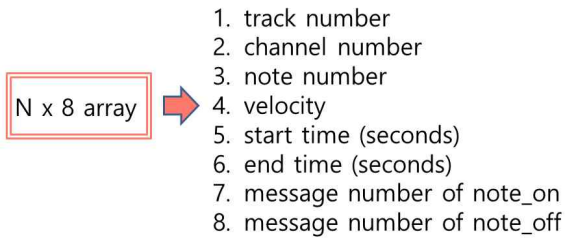


그림 6 : MIDI file 구성 요소

MIDI file은 (Note 수)X8 차원의 행렬로 이루어져 있다. Note 수란, Audio 신호 속 하나의 음정으로 인식되는 소리의 개수이다. 만약 Audio 신호가 피아노 곡이라면 눌러진 건반 개수를 의미한다. 각각의 음은 8개의 정보를 가진다. 이때, 실질적으로 MIDI file에 관여하는 1) ~ 6) 번이며, 7), 8) 번은 생략되어도 괜찮다.

1) Track number : 해당 음정이 들어있는 트랙의 인덱스
2) Channel number : 해당 음정이 들어있는 채널의 인텍스
3) Note number : 해당 음정에 해당하는 전자피아노 건반 번호
4) Velocity : 해당 음의 세기
5) Start time : 해당 음이 눌러지기 시작한 시간
6) End time : 해당 음이 끝난 시간

이때, 1), 2) 번은 1개로 통일하여 음정을 동일한 트랙과 채널에 넣었고, 4) Velocity를 일정하게 처리하여 MIDI file 결과를 simplify 하였다.

나. Harmonics제거 - Dynamic window

우리 조는 처음에 Window의 세분화 없이 일괄적인 Threshold 조건을 적용하여 Harmonics 제거를 꾀하였다. 이 방법은 큰 음의 Harmonics를 제거하고 작은 음

의 fundamental frequency를 인식하기엔 오류가 많았다. 대략적인 Harmonics는 제거했으나, 작은 음의 Fundamental frequency까지 모두 삭제해버렸기 때문이다. 우리 조는 일괄적인 threshold를 적용하여 harmonics를 제거하는 방법은 적절하지 못하다고 판단하였다.

우리는 이러한 문제점을 해결하기 위해 Dynamic window algorithm을 고안하였다. Dynamic window 방법은 크기가 비슷한 Harmonics와 detect 해야 하는 신호를 구분하기 위해 Window를 여러 개 나누고 Window마다 다른 Threshold를 적용한다.

큰소리의 Fundamental frequency와 Harmonics는 같은 시간에 발생하며 이 둘은 크기 차이로 구분할 수 있다. 그러나 다른 시간에 발생한 작은 소리의 Fundamental frequency가 큰소리의 Harmonics와 크기 차이가 없으므로 우리는 큰 음의 Harmonics와 작은 음의 Fundamental frequency를 독립적으로 처리할 수 있도록 분리하고자 노력했다. 이 둘은 서로 다른 시간에 존재하므로 시간에 따라 Window를 나눠주고 Window들을 독립적으로 처리하여 큰 신호가 작은 신호에 영향을 미치지 못하도록 한다.

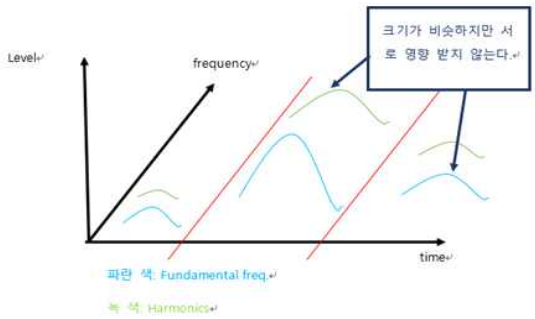


그림 7 : Dynamic Window 적용 모습

<그림 7>은 Dynamic window의 원리를 간단하게 나타낸 것이다. 파란색은 Fundamental frequency, 녹색은 Harmonics라 할 때, 같은 시간에 발생한 파란색 음은 항상 녹색음 보다 높은 level을 가진다. 두 번째 Window 속 큰 음의 Harmonics는 level이 비슷한 세 번째 Window 속 Fundamental frequency에 영향을 미칠 수 없고, 각각의 Window (빨간색 선으로 구분되어있음) 속에서만 처리되므로 작은 level의 Fundamental frequency는 살아남고, 높은 level의 Harmonics는 더 높은 level의 Fundamental frequency 때문에 제거된다.

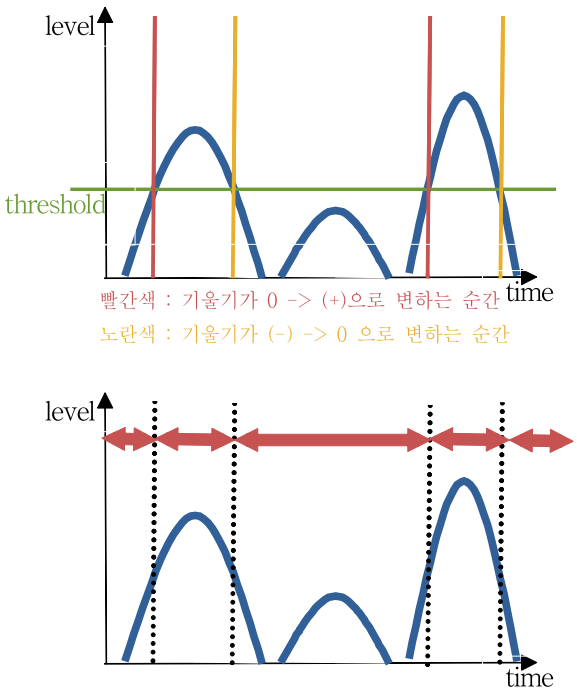


그림 8 : Dynamic Window의 경계선

<그림 8>에서 다양한 크기의 Window를 결정하는 과정을 나타내고 있다.

1) 신호 전체에 큰 Threshold를 적용하여 매우 큰 신호만 남기고 다 삭제한다.
2)Threshold가 적용된 상태에서 기울기가 zero ->(+)로 바뀌는 순간을 기준으로 Window를 나눈다.
3) Threshold가 적용된 상태에서 기울기가 (-) -> zero 로 바뀌는 순간을 기준으로 Window를 한번 더 나눈다.
4)원 신호를 Window들 간 서로 독립적으로 처리한다.

<그림 8>의 경우 빨간 선으로 2번, 노란 선으로 2번 분할되었으므로 총 5개의 Window를 가지게 된다.

이 알고리즘을 이용하면 Harmonics를 효과적으로 제거할 수 있어 각 음을 보다 정확하게 분류해낼 수 있다. 동시에 여러 개의 음을 치더라도 소리의 크기가 비슷하다면 detect 할 수 있다. 그러나 큰 소리의 음과

작은 소리 음이 동시에 쳐진다면, 이 둘을 시간에 따라 나눌 수 없기 때문에 각각 처리할 수 없고, 해당 Window에 Threshold 적용 시 작은 음이 아예 삭제된다. 그러나 Audio file에서도 큰 소리와 작은 소리가 동시에 발생한다면 작은 소리가 아예 삭제되진 않지만, 묻혀서 잘 안 들린다. 즉, MIDI file에서 큰 소리와 동시에 발생한 작은 소리가 삭제되더라도 Audio file과 큰 차이가 발생하지 않는다.

피아노의 damper가 첨가된 상태에서 얻게 된 Audio file을 처리 시 damper 때문에 발생한 연음 때문에 Window를 제대로 나눌 수 없게 되고 음의 정확한 인식이 불가능해진다는 단점이 있다.

다. 추출 된 data로 peak값을 통한 대푯값구하기

이상적으로 Harmonics를 모두 제거했다면, 남은 Data들은 모두 Fundamental Frequency 이다. 정확한 음정과 박자를 알기 위해서는 소리의 Frequency와 Time duration이 필요하다.

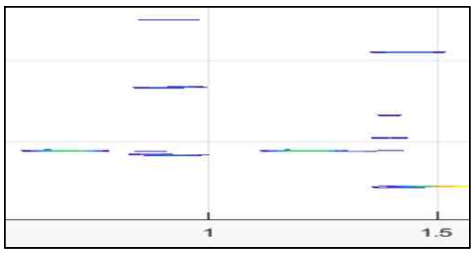


그림 9 : Harmonics가 제거된 Contour plot

<그림 9>는 Dynamic window 알고리즘을 이용하여 Harmonics를 제거한 결과 영상이다. 우리 귀에는 등고선 덩어리 하나당 하나의 음정으로 인식된다. 그러나 실제 Data값을 살펴보면, Contour 덩어리 1개가 수많은 Data로 이루어져 있다.

MIDI file을 작성하기 위해 서 음정 하나당 1개의 Frequency, 시작시간, 끝 시간만 필요하므로 Contour 덩어리 하나당 1개의 Frequency를 정해야한다. 또한 Contour의 중심 시간을 정하고 이를 중심으로 시작시간과 끝 시간을 추정해야 한다. 우리 조는 level이 제일 높은 점을 대표 값이라 생각하고 해당 지점의 Frequency와 Time을 함께 저장했다.

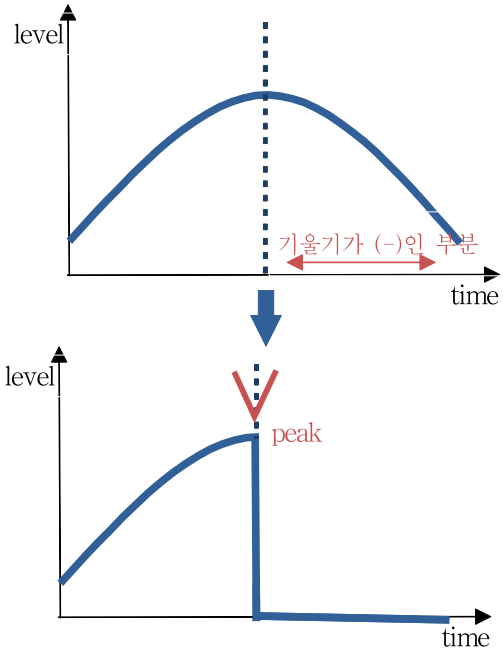


그림 10 : level이 peak인 time을 얻기

<그림 10>는 peak값을 가지는 time을 얻는 과정이다. level에는 (-)값이 없으므로, 기울기가 0 ->(-) 혹은 (+) -> (-)가 되는 순간이 peak점이 된다.
기울기가 (-)인 지점의 level을 모두 0으로 만들면 Contour 덩어리 하나당 기울기가 (-)인 부분이 1개로 줄어든다. 이후 기울기가 (-)인 지점은 모두 peak지점이라고 생각하고 해당 지점의 Time과 Frequency를 저장한다.

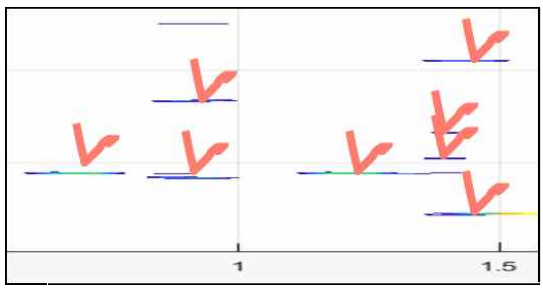


그림 11 : peak값들이 추출된 모습

<그림 11>은 <그림 9>의 Contour 덩어리 중 level이 높은 지점(색깔이 노랗거나 연두색) 임의로 표시한 것이다.

저장한 대푯값들은 이처럼 1개 Contour 덩어리 당 1개의 대푯값만 가질 때 가장 이상적이다. 1개 Contour덩어리 당 여러 개의 Frequency들이

오차범위인 2Hz 이하로 차이 날 경우 1개의 음정으로 처리하기 위해 Frequency 들의 평균값을 저장한다. 저장된 Frequency의 시간이 대표Time 값이 된다.

이때, 오차범위를 2Hz로 정한 이유는 자주 사용되는 음정 중 가장 낮고 주파수 차이가 적게 나는 (음정이 높아질수록 사이 주파수 간격이 넓어진다.) '2옥타브 도' 와 '2옥타브 레' 가 4Hz 차이가 나기 때문이다. 2Hz까지는 Frequency 오차가 나도 제대로 된 음정을 알 수 있으므로 오차범위를 2Hz로 정하였다. 만약 Frequency가 2Hz이상 차이가 난다면 이는 다른 음정으로 취급된다. 오차범위가 작을수록 낮은음까지 섬세하게 잡아낼 수 있지만, 오차범위가 너무 작으면 높은 음정에서 같은 음정을 다른 음으로 취급하게 된다. 이는 부정확한 MIDI file을 초래하므로, 자주 사용되는 음역 대를 기준으로 오차범위를 제한했다.

라. Time duration구하기

Contour 한 덩어리를 대표하는 대푯값을 이용하여 시작시간과 끝 시간을 추정해야한다.

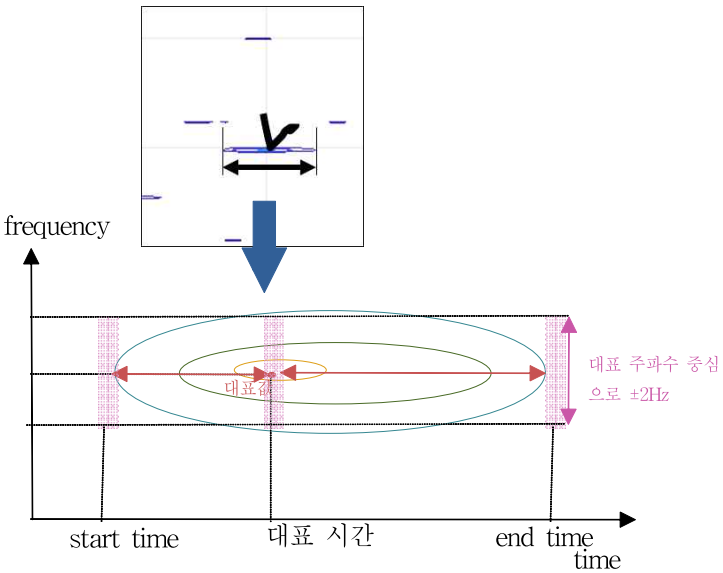


그림 12 : Time duration계산

<그림 12>sms Time duration 계산하는 과정을 간단하게 나타내고 있다. 대표 주파수 중심으로 (±오차범위) 크기의 스캔용 Window를 만든다. 대표 시간을 중심으로 앞뒤로 스캔용 Window속 값이 0이 나올 때까지 스캔한다. Window속 값이 모두 0이 되는 순간시간을 Start time/End time이라 추정하고 이를 대푯값과 함께 저장한다.

마. Excel을 이용하여 frequency matching 하기

엑셀의 Frequency - MIDI note number가 서로 매칭 되어있는 data table을 이용하여 다)에서 저장한 대표 주파수를 MIDI note number로 바꾼다.

1		hz	pitch	오차범위	
2	0C(도)	16	12	0.00	16.49
3	C#	17	13	16.50	17.49
4	D(레)	18	14	17.50	18.59
5	D#	20	15	18.60	20.49
6	E(미)	21	16	20.50	21.49
7	F(파)	22	17	21.50	22.49
8	F#	23	18	22.50	23.59
9	G(솔)	25	19	24.60	25.49
10	G#	26	20	25.50	26.55
11	A(라)	28	21	26.56	28.55
12	A#	29	22	28.56	29.55
13	B(시)	31	23	29.56	32.55

그림 13 : Frequency - MIDI note number matching table

바. Audio2midi를 이용하여 MIDI file생성하기

MIDI file format에 맞춰 저장한 Data를 집어넣은 array를 작성하고, MATLAB의 matrix2midi와 writemidi 함수를 이용하여 MIDI file을 만든다.

3. 개선 내용

가. Eradicated harmonics

기존 Audio to MIDI converting system은 Harmonics들 까지 모두 포함된 결과의 MIDI file을 생성했다. 우리 조가 만든 Audio to MIDI converting system은 Harmonics를 제거하기 위하여 Dynamic한 크기를 가지는 Window 들로 time축을 나눈다. 각각의 Window안에는 비슷한 level의 Window만 존재하게 처리한다. 작은 level의 음만 존재하는 Window에서는 작은 Threshold값을 적용하여 작은 Fundamental frequency는 살리고, 더 작은 Harmonics만 한다. 높은 level의 음만 존재하는 Window에서는 높은 Threshold 값을 적용하여, 큰 Harmonics를 제거하고 더 큰 Fundamental frequency만 살린다.

나. STFT parameter 조정

time to frequency로 Audio file을 나타내기 위해서 STFT를 수행해야한다. STFT는 parameter값에 따라

time resolution과 frequency resolution이 달라지는데 이 둘은 trade-off 관계이므로 최적화된 parameter를 찾는 것이 중요하다.

STFT 입력 parameter 중 Window length를 늘리면 time resolution이 줄어들면서 frequency resolution이 증가하게 된다. 낮은음은 음정 사이 주파수 차이가 작으므로 frequency resolution이 클 때 정확하게 detect가 능하다. 대신 time resolution이 줄어들면서 높은 음이 일부 소실된다.

반대로 Window length를 줄이면 Frequency resolution이 줄어들면서 time resolution이 증가하게 된다. 높은 음의 경우 음정사이 주파수 사이가 크므로 frequency resolution 보다 time resolution이 더 중요하다. 높은 time resolution으로 높은 음이 소실되지 않는다. 대신 Window length가 너무 낮을시 frequcy resolution이 떨어지므로 정확한 낮은음을 알 수 없게 된다.

그러나 노래 하나에는 낮은음과 높은음 모두 들어가 있다. 결과를 최적화하기 위하여 주파수에 따라 STFT parameter를 다르게 적용하였다.

4. 실험 결과

가. Eradicated harmonics

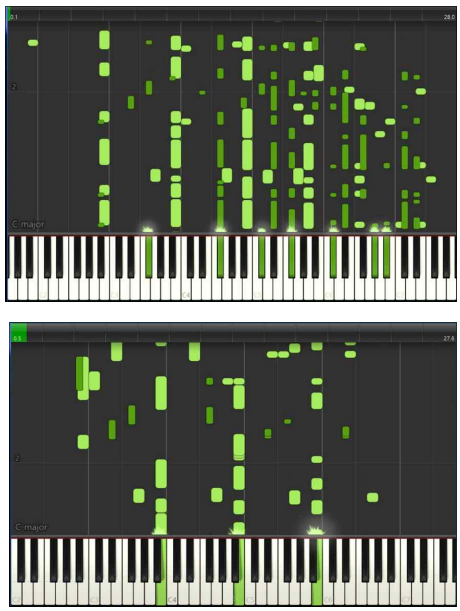


그림 14 : 기존의 Audio2MIDI converter(위)와 새롭게 고안한 Audio2MIDI(아래)

<그림 14>는 같은 곡을 기존 Audio to MIDI converter를 이용하여 생성한 MIDI file과 Dynamic window 알고리즘을 이용하여 Harmonics를 제거한 후 생성한 MIDI file을 비교한 그림이다. Harmonics가 제거되어 훨씬 깔끔한 결과가 나온 것을 확인 할 수 있다.

나. 주파수에 따른 STFT parameter 변화

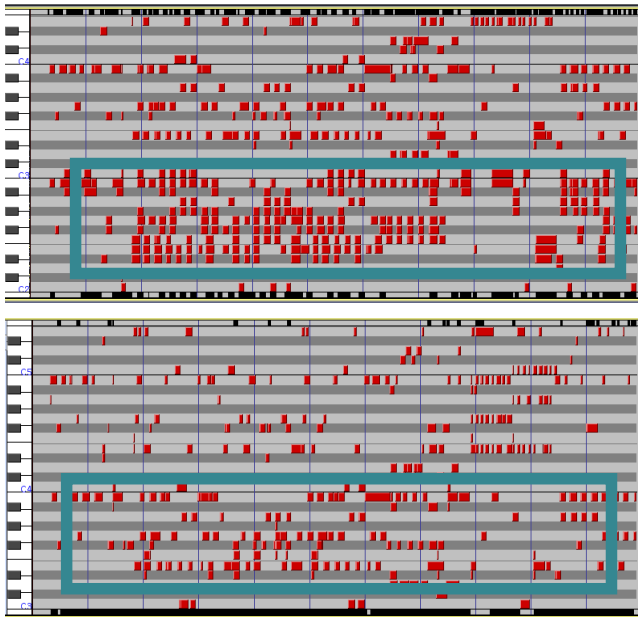
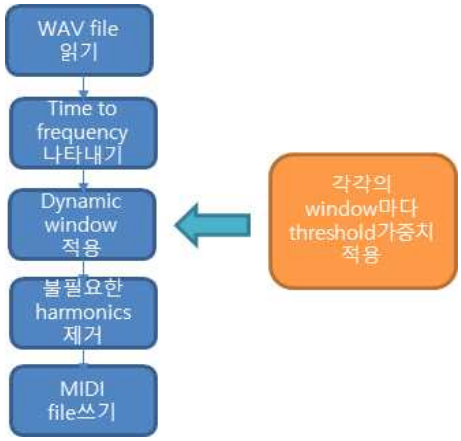


그림 15 : STFT parameter를 일괄적으로 적용(위)와 STFT parameter를 주파수에 따라 다르게 처리한 MIDI file 영상(아래)

<그림 15>에서 STFT parameter를 일괄적으로 적용한 결과와 STFT parameter를 주파수에 따라 다르게 처리한 결과를 비교했다. 같은 곡/같은 부분이지만 파란색 네모 속 신호가 좀 더 깔끔해 진 것을 알 수 있다. <그림 15> 아래 쪽 그림에서는 주파수가 작아질수록 Window length를 늘려주었다. frequency resolution이 커지면서 주파수 차이가 작은 낮은 음을 섬세하게 잡아냈다.

5. Flow Chart



6. 한계 및 보안할 점

가. Eradicated harmonics

피아노 Damper가 들어간 곡의 경우 연음이 형성되면서 이전 음정의 에너지가 계속 지속되므로 제대로 Window를 나눌 수 없었다. 결과로 생성된 MIDI file도 오류가 많았다.

나. 주파수에 따른 STFT parameter 변화

주파수에 따라 STFT input parameter를 다르게 설정 할 시 낮은 음은 높은 frequency resolution / 낮은 time resolution을 가지고, 높은 음은 낮은 frequency resolution / 높은 time resolution을 가진다. time resolution차이 때문에 낮은 음과 높은음의 synchro가 미묘하게 다르다. Audio file처럼 음악 감상이 목적이라면 다소 거슬리게 들린다. 그러나 음악 분석 및 사보를 위해 MIDI file을 사용한다면 음정의 시간차가 매우 작기 때문에 큰 문제를 일으키지 않는다.

다. Velocity의 통일

우리 조는 MIDI file을 만들 때 Velocity를 일정하게 만들었다. 결과로 나온 MIDI file의 음정세기를 일정하게 만든 것이다. 음의 세기가 계속 바뀌는 Audio file과는 다르게 MIDI file은 일정한 세기로 음이 출력될 것이다. 음악 감상용으로는 다소 부적합하나 사보를 할 경우 이는 크게 문제 되지 않는다.

7. 진행 일정 및 역할 분담

가. 진행 내용 및 일정

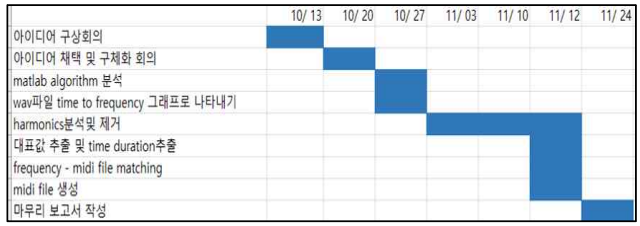


표2 : 진행내용 및 일정

나. 역할 분담

팀 원	역 할	기여도(%)
최윤정	Data extraction: Peak data time duration,Frequency-Pitch Matching, 발표, 보고서 제작, 포스터 제작, 시연영상 제작	50
김동영	Data extraction: Peak data time duration,Frequency-Pitch Matching, 발표, 보고서 제작, 포스터 제작, 시연영상 제작	50

표 3 : 역할 분담

III. 결 론

서론에서 언급한 바와 같이 현재 많이 사용되는 Audio to MIDI converting system은 Harmonics를 제대로 제거하기 못한다. 하지만 Dynamic window algorithm을 통하여 Harmonics를 제거한 후 MIDI file을 생성 하면, 정확한 정보의 midi file을 얻을 수 있어 오류 없는 악보를 얻을 수 있다. 시중의 인터넷 사이트에서 악보를 구하기 어려운 곡의 wav파일만 가지고 있다면 이 시스템을 통해 거의 정확한 악보를 얻을 수 있다.

최근 스트리밍 플랫폼 시장의 규모가 높아지면서, 스트리밍을 통해 접한 음악의 악보를 얻고자 하는수요도 증가하고 있다. 이 프로젝트는 작곡 혹은 음악 편집 등 정확한 정보의 MIDI file을 필요로 하는 곳에서 수요가 많이 발생할 것으로 예상된다. 또한 정확한 정보의

MIDI file을 바탕으로 리듬게임 혹은 여러 가지 다양한
전자 악기를 통한 곡 제작과 편집에 응용될 수 있다.

참 고 문 헌

[1] score making program
http://mosesoft.com/mose/home/page/home.php

[2] music interface - based game software
http://www.agame.com/games/music

[3] audio to midi file converting
https://www.ofoct.com/audio-converter/convert-wav-or-mp3-ogg-aac-wma-to-midi.html