



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

碩士學位論文

백색잡음 제거와 DTW를 이용한
목 넘김 소리의 인식

濟州大學校 大學院

컴퓨터工學科

金 佑 燦

2020 年 2 月

백색잡음 제거와 DTW를 이용한 목 넘김 소리의 인식

指導教授 郭鎬榮

金佑燦

이 論文을 컴퓨터工學 碩士學位 論文으로 提出함

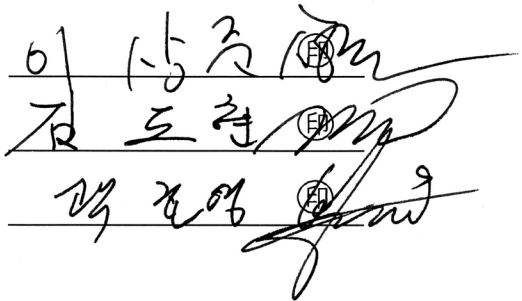
2019 年 12月

金佑燦의 工學 碩士學位 論文으로 認准함

審査委員長

委員

委員



濟州大學校 大學院

2019 年 12月



A Recognition of Gulping Sound using White noise Reduction
and DTW

Woo-Chan Kim

(Supervised by professor Ho-Young Kwak)

A thesis submitted in partial fulfillment of the requirement for the degree of
Master of Science in Computer Engineering

2019 12.

This Thesis has been examined and approved.

Thesis director, Gang Joon Lee

Thesis director, Do Hyeon Kim

Thesis director, Ho young Kwak

December 2019

Department of Computer Engineering
GRADUATE SCHOOL
JEJU NATIONAL UNIVERSITY

목 차

목 차	i
그림목차	iii
표 목 차	iv
국문초록	i
Abstract	ii
I. 서 론	1
II. 관련 연구	3
1. 기존 소리 인식 시스템	3
2. 노이즈 제거 알고리즘	3
3. MFCC(Mel Frequency Cepstral Coefficients)	6
4. Dynamic Time Warping	7
5. Audio time scale modification	10
III. 제안 시스템	12
1. 핵심 시스템 구조	12
2. 표준 목 넘김 사운드	15
3. 개별 프레임 사운드 MFCC	19
4. DTW 계산 및 Threshold를 이용한 목 넘김 횟수 세기	22
IV. 구현 및 평가	26
1. 구현 및 실험 환경	26
2. 평가	29

V. 결론	36
참고문헌	38

그림 목 차

그림 1. 화이트 노이즈에 대한 푸리에 변환	4
그림 2. 마스크의 민감도와 Threshold 계산 그래프	5
그림 3. 부드럽게 만들어주는 마스킹 필터	5
그림 4. MFCC 계산 과정	6
그림 5. 핵심 시스템 다이어그램	12
그림 6. 표준 목 넘김 사운드 MFCC 특징 구하는 절차	15
그림 7. 정규화 전 음성파일 그래프	17
그림 8. 정규화 후 음성파일 그래프	17
그림 9. 개별 프레임 MFCC 특징 구하기	20
그림 10. DTW 계산된 거리 수치 및 평균과 Threshold 값 비교	23
그림 11. Threshold 값을 이용하여 구한 구간	25
그림 12. Idam K10-PRO 녹음기	26
그림 13. 웨어러블 디바이스 전면	27
그림 14. 웨어러블 디바이스 후면	27
그림 15. 목 넘김 사운드 믹스	28
그림 16. mixing-gulp-fan-metal-buzz 사운드 파일의 노이즈 제거 후 그래프	31

표 목 차

표 1. DTW 비교 신호 값	7
표 2. 두 신호의 거리(Cost/Distance)표	8
표 3. 구현 시 사용한 라이브러리 목록	29
표 4. 웨어러블 디바이스 녹음 사운드를 이용하여 생성한 목 넘김 인식 대상 사운드	30
표 5. 웨어러블 디바이스 녹음 사운드를 이용하여 생성한 목 넘김 인식 결과	30
표 6. 웨어러블 디바이스 녹음 사운드를 이용하여 생성한 목 넘김 인식 대상 사운드	32
표 7. 웨어러블 디바이스 녹음 사운드를 이용하여 생성한 목 넘김 인식 결과	32
표 8. 웨어러블 디바이스 녹음 사운드 및 곰 녹음기를 통해 얻은 팬 돌아가는 사운드를 이용하여 생성한 목 넘김 인식 대상 사운드	33
표 9. 웨어러블 디바이스 녹음 사운드 및 곰 녹음기를 통해 얻은 팬 돌아가는 사운드를 이용하여 생성한 목 넘김 인식 결과	33
표 10. 녹음기를 통해 얻은 목 넘김 소리 및 곰 녹음기를 통해 얻은 팬 돌아가는 사운드를 이용하여 생성한 목 넘김 인식 대상 사운드	34
표 11. 웨어러블 디바이스 녹음 사운드 및 곰 녹음기를 통해 얻은 팬 돌아가는 사운드를 이용하여 생성한 목 넘김 인식 결과	35
표 12. 최종 목 넘김 인식 결과	35

백색잡음 제거와 DTW를 이용한 목 넘김 소리의 인식

컴퓨터공학과 김 우 찬
지도 교수 곽 호 영

축사에서 사육하고 있는 소의 사료 섭취량을 추정하기 위해서 목 넘김 횟수를 카운트하여 이를 통해 사료 섭취량을 예측할 수 있다. 목 넘김 횟수를 이용한 사료 섭취량을 추정하기 위해서는 소의 목 넘김 소리를 취득해야 하는데, 축사에서는 축사의 온도를 적정하게 유지하기 위해서 지속적으로 환풍기를 작동시키고 있다. 수집된 목 넘김 사운드 파일로부터 목 넘김 소리를 인식할 때, 환풍기 및 자연에서 발생하는 백색잡음의 영향이 매우 크다.

본 논문에서는 목 넘김 횟수를 카운트하기 위해서 녹음 수집된 사운드 파일을 대상으로 Spectral noise gate 기술을 이용하여 백색잡음을 제거하고, 개별 목 넘김 소리를 평균하여 표준 목 넘김 사운드 특징을 추출하였다. 표준 목 넘김 사운드의 MFCC 특징과 목 넘김 인식하고자 하는 사운드의 개별 프레임 MFCC 특징을 Dynamic Time Warping 알고리즘을 통해 거리 값을 계산하였고, 계산한 결과를 통해 얻을 수 있는 거리 값들에 대해 평균과 표준편차를 이용한 Threshold를 이용하여 목 넘김 소리를 인식하였다. 잡음을 제거한 후, 인식된 목 넘김 소리는 사료 섭취량을 추정하는데 매우 유용하게 사용될 수 있을 것으로 기대한다. 본 논문에서 제안한 소의 목 넘김 횟수 인식 방법을 이용한 개별 소의 사료 섭취량 유추와 활동량이나 체온 등과 같은 다른 요소들과 결합시켜 발정이나 질병을 조기에 인지할 수 있는 기반이 될 수 있을 것으로 기대한다.

주제어 : 목 넘김 소리, 소리 인식, 잡음 제거, DTW(Dynamic Time Warping)

Abstract

A Recognition of Gulping Sound using white noise Reduction and DTW

Kim, Woo-Chan
Department of Computer Engineering
Graduate School

Supervised by Professor Kwak, Ho-Young

In order to estimate the feed intake of the cattle raised in the cattle shed, it is possible to estimate the feed intake by counting the number of gulp down. To estimate feed intake using number of gulp down, cow's gulping sound should be obtained. However the fan is continuously operated to maintain proper temperature. Because of these fan noise and noise from nature, it is hard to recognize gulping sound from collected gulping sound file.

In this paper, white noise at collected sound file was reduced using Spectral noise gate and extract standard gulping sound feature value by take an average of each gulping sound. We calculate distance between MFCC feature of standard gulping sound and MFCC feature of each frame from target sound using Dynamic Time Warping algorithm. We recognize gulping sound from distance using Threshold calculated from mean and standard deviation of distance. it is expected that the recognized gulping sound after removing the noise can be very useful for estimating feed intake. We expect Feed intake using proposed method in this paper can be base of early recognition estrous and disease by combining physical activity level and body temperature

Keyword : Gulping Sound, Sound Recognition, Noise Reduction, Dynamic Time Warping

I. 서 론

오늘날 축산업은 지속적으로 발전하고 있다. 한우/낙농의 경우 기술발전으로 생산 자동화나 ICT 기술을 접목하여 다양한 형태의 자동관리 기술을 적용하고 있다. 또한, 이를 통해서 생산 효율성을 증대시키고 있다. 예를 들면 발정탐지기와 같은 기기를 적용하여 발정기를 놓치지 번식을 하고 있으며, 로봇 착유기 및 로봇 포유기를 통해 좀 더 효율적인 우유생산 및 송아지 사육을 하고 있다.

그러나 이들은 단순히 정보의 수집하거나 관찰할 수 있는 수단을 만들어 주는 기능을 제공하는 경우가 많아서 실질적인 개별 개체에 대한 자동화된 관리와는 거리가 멀다. 축산 현장에서는 발정기의 예측이나 개체의 건강관리, 사료 관리 등 전반적인 관리를 위해서는 개체별로 특정 정보를 수집하고 가공하여 저렴한 방법을 통해 체계적인 관리 방법이 필요하다. 개체(한우 또는 젖소) 관리를 위한 중요한 요소 중 하나는 개별 개체들이 먹는 사료섭취량을 파악하는 것이다.

이러한 관점에서 그동안 사용되어 오던 전자저울을 이용한 기존의 사료섭취량 측정 방식은 비용이 많이 들뿐 아니라 시설을 새로 구축해야 하는 불편함이 있다. 또한 일반적으로 사료통 밑에 설치하는 전자저울은 개별 개체의 사료 섭취량을 파악하기 힘들다는 한계점을 가지고 있다.

개별 개체의 목에 웨어러블 디바이스를 장착하고 해당 디바이스를 통해 수집되는 각종 신호를 분석한다면, 개별 개체별 관리가 가능하다. 이를 통해 전체 개체 개체들에 대한 다양한 정보를 토대로 건강 및 발정기, 활동량을 관찰

할 수 있다.

IoT(Internet of Things)기술이 발전하면서 음성인식(Sound Recognition)기술도 발전하고 있으나, 대동물에 대해 IoT 기술을 적용한 사례는 국내외에서 극히 한정적이다.

본 논문에서는 축사에서 사육되고 있는 대동물(육우/젖소)이 사료를 저작(咀嚼)하고, 식도를 통해서 저작된 사료를 넘길 때 발생하는 목 넘김 소리를 수집한 후 소리 신호의 추적을 통해 목 넘김 횟수를 세는 방법 및 백색잡음을 제거하는 방법을 제안한다.

현재 농장에서 운영비용의 70%는 사료구매에 사용되는 비용이다. 따라서 성공적인 농장 경영을 위해서는 사료투입량 조절이 필요하다. 또한 소의 번식을 위해서 발정기를 모니터링 하는데 사료섭취량에 대한 정보가 매우 중요한 요소 중 하나이다.

또한, 제안된 시스템을 적용하여 수집한 정보를 이용하여 개별 개체별 평균 사료 섭취량을 구할 수 있고 이를 이용하여 평균보다 현저히 덜 먹거나 더 먹는 경우에 발정기나 질병으로 의심할 수 있다.

본 논문에서는 기존에 사용된 소리 인식 연구 사례와 목 넘김 소리 인식 시스템을 구성하는 관련 연구 및 기술에 대해서 서술한다. 이후 시스템의 구조에 대해서 제안하고, 이 시스템을 이용한 목 넘김 횟수 인식 평가 결과에 대해 제시하였다.

II. 관련 연구

1. 기존 소리 인식 시스템

기존 소리 인식 시스템의 사례로 “DTW를 이용한 윈도우 기반 명령어의 음성인식 구현 방법”에서는 LPC(Linear Predictive Coding)을 이용한 음성 Feature와 DTW(Dynamic Time Warping) 알고리즘을 활용하여 아래아 한글 환경에서 사용되는 명령어를 인식하는 시스템을 제시하였다[2].

다른 사례로 “Heart Sound Diagnosis Based on DTW and MFCC”에서는 심장이 정상적으로 뛰는 소리와 정상적이지 않는 케이스를 나누어 표준 사운드를 만든 후 심장 사운드가 어떤 패턴인지 구분하는 시스템을 제시하였다[3].

“Speech Recognition using MFCC and DTW”에서는 MFCC특징과 DTW 알고리즘을 이용하여 “Forward”, “Left”, “Right”, “Reverse”, “Control” 5개 단어를 인식하는 시스템을 제시하였다[4].

다수의 연구에서 DTW와 MFCC를 이용하여 소리 인식 시스템을 제시하였다.

2. 노이즈 제거 알고리즘[8,9,10]

노이즈 제거 알고리즘으로 Audacity라는 프로그램이 사용하는 Spectral Noise Gate라는 알고리즘을 사용하였다.

Spectral Noise Gate 알고리즘은 음성 데이터를 스펙트로그램으로 변환하여

개별 주파수 단위로 각각 Threshold를 결정한 후, 개별 주파수에 대하여 Threshold 미만의 에너지를 줄이는 방식을 통해 잡음을 제거하는 기술이다.

이 알고리즘에 대해서 Tim Sainburg가 자세히 분석해두었다[10].

알고리즘의 처리 순서의 첫 번째는 노이즈 클립의 모든 영역에 대해서 FFT(Fast Fourier Transform)연산을 적용한다. 여기에서 사용되는 FFT 연산 방법은 일반적으로 STFT(Short Time Fourier Transform)이라는 이름으로 불리는 기술로, 신호 데이터를 주파수-시간 스펙트로그램으로 변환할 수 있는 알고리즘이다.

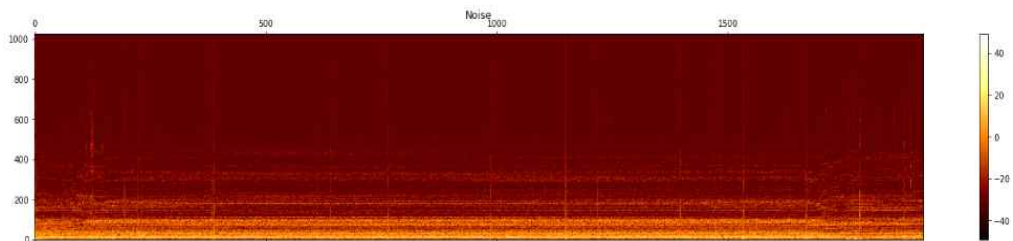


그림 1. 화이트 노이즈에 대한 푸리에 변환

두 번째 단계는 앞에서 계산한 스펙트로그램으로부터 통계적인 데이터를 얻는 것이다. 이 단계에서 얻은 통계 데이터를 통해 Threshold 값을 결정하게 된다.

세 번째 단계는 앞에서 얻은 통계데이터를 통해 주파수 별로 Threshold를 계산하는 과정이다. 여기에서는 Threshold를 결정하기 위해서 아래의 수식을 이용하였다.

$$Threshold_f = mean_f + std_f \times sensitivity$$

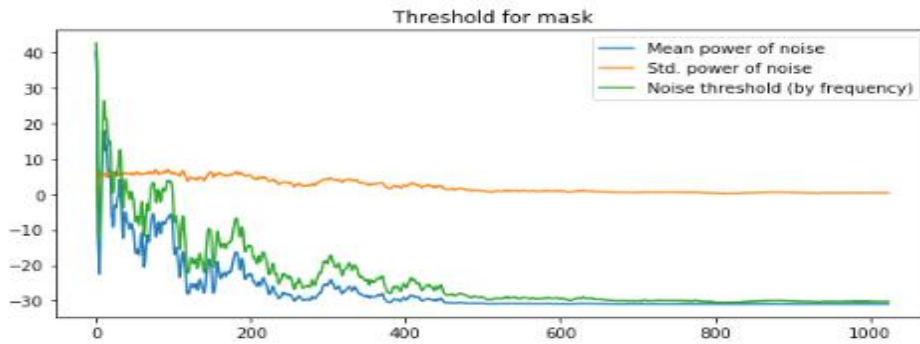


그림 2. 마스크의 민감도와 Threshold 계산 그래프

네 번째 단계는 노이즈를 제거하고자 하는 사운드에 대해서 첫 번째 단계와 같은 방법으로 FFT를 적용한다.

다섯 번째 단계는 세 번째 단계에서 얻은 주파수별 Threshold를 가지고 Mask를 생성하는 것이다. 이 마스크를 통해 차후에 노이즈에 해당하는 에너지를 줄이게 된다.

여섯 번째 단계에서는 다섯 번째에서 구한 마스크에 대해서 스무딩 필터를 적용하여 좀 더 완만한 마스크로 만들게 된다. 이 과정을 통해 자연스럽게 노이즈를 줄일 수 있다.

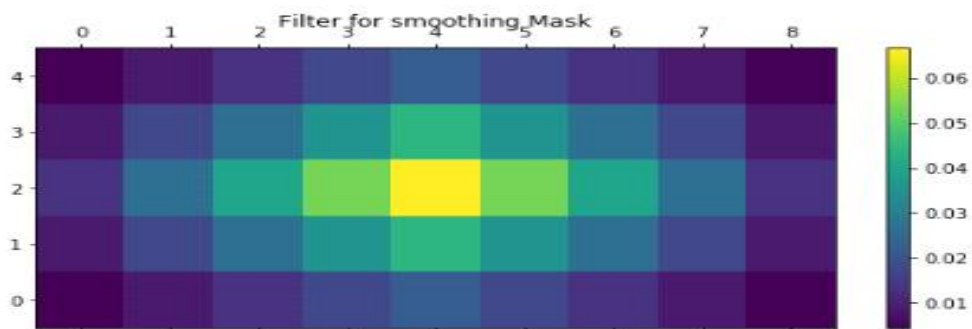


그림 3. 부드럽게 만들어주는 마스크 필터

일곱 번째 단계에서는 여섯 번째 단계에서 만든 마스크를 네 번째 단계에서 구한 스펙트로그램에 적용하여 노이즈의 에너지를 줄이고, 해당 스펙트로그램을 다시 음성신호로 변환한다. 이 과정에서 사용된 방법을 ISTFT(Inverse

Short Time Fourier Transform)이라고 부른다.

3. MFCC(Mel Frequency Cepstral Coefficients)[5,11,12]

MFCC는 음성인식에서 많이 사용하는 특징 중 하나이다. 또한 많은 응용에서 효과적이고, 견고하다고 알려져 있다.

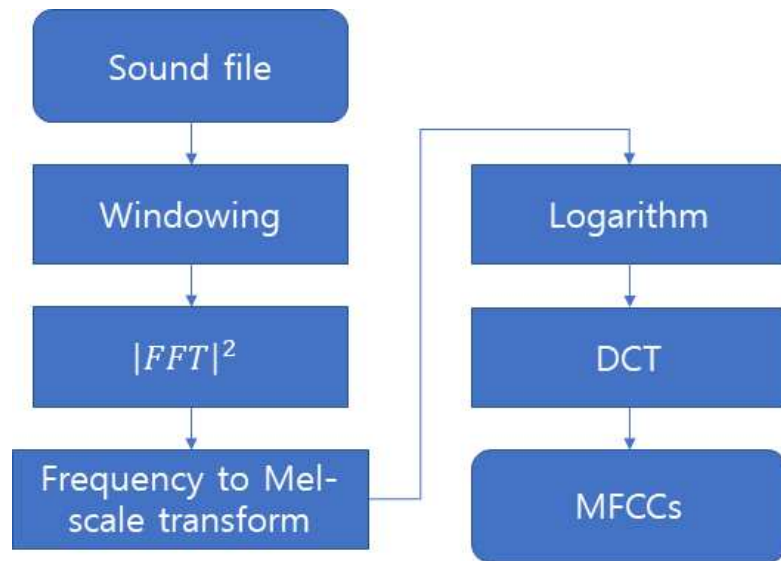


그림 4. MFCC 계산 과정

MFCC 특징은 기본적으로 STFT(Short Time Fourier Transform)을 취한 후 얻을 수 있는 magnitude 스펙트로그램에 추가적인 변환을 거쳐 만들어진다.

그림 4는 MFCC 특징 계산 과정에 대해서 표현을 하고 있다. 사운드 파일에 대해서 일정 구간과 간격으로 윈도우를 옮겨가면서 푸리에 변환을 진행하게 된다. 이후 제곱 합을 통해 magnitude 값을 구하게 되는데 주파수 별 에너지 값을 구한다고 볼 수 있다. 이 과정을 STFT가 진행하는 과정이다.

이후 Mel-scale이라는 단위로 주파수 단위를 변환하게 되는데, Mel-scale은 소리의 주파수적 단위를 사람의 지각적 특성에 맞춘 Mel이라는 단위로 표현

한 것으로 사람의 지각적인 특징을 더욱 잘 나타내어 준다[12]. 또 에너지 측면으로 로그를 취하게 되는데, 사람이 듣는 소리는 로그 스케일 단위로 느끼기 때문이다. 이후 이산 코사인 변환을 거치면 MFCC 특징을 얻을 수 있다.

이렇게 얻은 MFCC 특징은 사람이 듣는 것과 유사한 특징을 얻을 수 있게 된다.

4. Dynamic Time Warping [6, 13]

DTW(Dynamic Time Warping) 알고리즘은 시간 관련 분석에서 가변속도의 두 신호의 유사성을 측정하는 알고리즘이다. DTW는 비디오, 오디오, 그래픽 데이터에 대해서 적용되었다. 잘 알려진 응용으로 자동 음성 인식(Automatic Speech Recognition)으로, 말하는 속도가 다른 것에 대응하기 위해 사용되었다.

본 논문에서도 인식하고자 하는 소리 패턴과 실제 패턴간의 유사성을 측정하기 위해서 사용하고 있다.

John Coleman의 “Introducing Speech and Language Processing”의 번역서인 음성처리와 자연언어 처리 개론에서도 패턴 인식 접근법으로 소개하고 있다. 저자는 시간적으로 동기화되어 있지 않지만 유사한 두 신호에 대해서 DTW를 이용한 비교를 통해서 알고리즘에 대해서 설명하였다.

두 신호를 테스트 신호와 참조 신호로 우선 구분하고 있다. 두 신호의 값을 다음의 표 1과 같이 제시하고 있다.

표 1 DTW 비교 신호 값

신호 설명	t=1	t=2	t=3	t=4	t=5	t=6	t=7
(a) 테스트 신호, x(t)	1	1	2	3	2	0	
(b) 참조 신호, y(t)	0	1	1	2	3	2	1
표본의 차이	1	0	1	1	-1	-2	미정의

테스트 신호의 어떤 지점이 참조 신호의 어떤 지점과 가장 밀접하게 대응하는지 찾아내자. 이를 위해서 두 신호간의 거리를 표 2와 같이 나타내었다.

표 2 두 신호의 거리(Cost/Distance)표

y[t]	1	0	0	1	2	1	1	
	2	1	1	1	1	0	2	
	3	2	2	1	0	1	3	
	2	1	1	0	1	0	2	
	1	0	0	1	2	1	1	
	1	0	0	1	2	1	1	
	0	1	1	2	3	2	0	
		1	1	2	3	2	0	x[t]

이 행렬은 테스트 신호의 각 프레임과 참조 신호의 각 프레임 사이의 거리를 계산한 것으로 아직까지는 두 신호가 어떻게 정렬되었는지 설명하지는 않지만, 어떻게 정렬하는지 계산하는 기초를 제공한다.

이제 두 신호의 정렬을 계산하는 단계이다. 두 신호의 시작과 끝은 언제나 서로 대응되어야하기 때문에 시작은 언제나 (1, 1)부터 시작한다. 경로는 ‘위쪽’, ‘오른쪽’, ‘대각선 우측상단’, ‘좌측하단에서 우측상단’등과 같은 일련의 이동이다.

시간은 언제나 앞으로 이동하기 때문에 왼쪽으로의 이동이나, 아래로의 이동은 금지되어 있다.

최적의 경로는 일반적으로 대각선에 가까이 있기 때문에 대각선을 따라서 우측 상단으로 이동하는 것이 선호되어야 한다.

테스트 신호와 참조 신호의 가장 가까운 대응을 발견하는 방법은 행렬을 따라서 가장 비용이 적게 드는 경로를 알아내는 것이다.

정리해서 우리가 사용하게 될 DTW 알고리즘의 두 번째 단계는 다음과 같이 설명될 수 있다.

행렬의 각 원소에 대해서 (a) 아래쪽, (b) 왼쪽, (c) 좌측 하단에서 출발해서 도착하는 비용을 가장 낮은 가격에 거리 비용을 더해서 계산하라.

DTW를 계산할 때 쓰이는 두 행렬을 *accdist*, *move* 행렬이다. *accdist* 행렬을 각 원소에서 도달하는 가장 저렴한 비용(누적 거리), *move* 행렬이 가장 저렴한 경로를 거쳐서 그 원소에 도착하는 이동을 저장하기 위해 사용된다.

$$move(x, y) = \begin{cases} 1, \text{위로 이동} \\ 2, \text{대각선 이동} \\ 3, \text{오른쪽 이동} \end{cases}$$

move 행렬은 가장 저렴한 이동 경로가 위 수식을 따라 결정된다. 예를 들어 가장 저렴한 이동경로가 대각선 이동이라면 2의 값을 가진다.

$$\begin{aligned} accdist(1, y) &= accdist(1, y-1) + d(1, y) \\ accdist(x, 1) &= accdist(x-1, 1) + d(x, 1) \end{aligned}$$

accdist 행렬은 초기에 위 두 수식에 따라서 가장 좌측의 열, 가장 하단 행의 값을 결정한다. 여기서 $d(x, y)$ 는 거리 행렬의 값을 의미한다. $accdist(1, 1)$ 의 값은 $d(1,1)$ 로 생각하자.

$$accdist(x, y) = \min \left(\begin{array}{l} accdist(x, y-1) + d(x, y) \\ accdist(x-1, y) + d(x, y) \\ accdist(x-1, y-1) + d(x, y) \end{array} \right)$$

이 행렬의 나머지 부분은 위 수식을 재귀적으로 적용하여 구할 수 있다. 또한 *move* 행렬은 *accdist* 행렬의 값을 구할 때, 적절한 이동을 기록하게 된다.

이렇게 필요한 두 행렬을 만들 수 있다. 프로세스를 끝내기 위해서 *move*

행렬에서 최적의 경로를 선택하여 상단 우측 모서리로부터 (1,1)까지 이르는 가장 저렴한 경로를 가리키는 이동을 선택하는 과정을 거꾸로 경로를 추적해 나간다. 이 과정을 통해 행렬에서 가장 테스트 신호와 참조 신호가 서로 유사한 경로를 알아내고 가장 저 비용이 드는 거리를 알아낼 수 있다.

5. Audio time scale modification[7,14]

Audio time scale modification 기술은 원래의 소리 신호의 음높이를 유지하면서 소리의 속도를 늘이고 줄이는 기술이다. 일반적으로 재생속도를 늘이거나 줄이는 방법을 사용하게 되면 소리의 음높이가 달라지는 부작용이 생긴다. 이 기술의 근간이 되는 절차는 다음과 같다.

1. 원래 소리를 여러 프레임으로 분리한다. 이 결과물을 Analysis frame이라고 한다. 이때 프레임의 길이는 일반적으로 50ms에서 100ms 크기의 프레임 사이즈를 갖는다. 프레임의 길이를 N 으로 표기한다. 프레임을 생성하는 간격을 H_a 로 표기한다. 이렇게 생성된 프레임을 x_m 으로 표기한다.

$$x_m(r) = \begin{cases} x(r + mH_a), & \text{if } r \in [-N/2 : N/2 - 1] \\ 0, & \text{otherwise} \end{cases}$$

2. 생성된 프레임들을 synthesis hopsize라고 부르는 크기의 간격으로 재배치한다. 해당 간격을 H_s 로 표기한다. 이렇게 재배치하는 과정이 실제 소리 신호의 속도를 늘리고 줄이는 역할을 한다. 이후 이렇게 재배치된 프레임들을 synthesis frame이라고 부르는 형태로 변환한다. 이 프레임을 y_m 으로 표기한다. 이때 $\alpha = H_s/H_a$ 라는 수식으로 정의되는 값은 stretching factor라고 부르고 속도를 늘이고 줄이는데 영향을 준다.

3. 위 단계에서 오버랩된 프레임들을 겹치고 결과 신호로 재합성하게 된다.

$$y(r) = \sum_{m \in Z} y_m(r - mH_s)$$

Audio time scale modification의 기술은 OLA(OverLap Add), WSOLA(Waveform Similarity OverLap Add), Phave Vocoder가 있다.

III. 제안 시스템

목넘김 소리를 인식하기 위해서 다양한 방식의 분류기를 이용하여 목넘김 패턴을 분석할 수 있다. 우리는 목 넘김 소리의 개수가 부족하고 DTW 알고리즘의 이해하기 쉽고 단순하지만 강력한 비교가 가능하다는 점 때문에 DTW 알고리즘을 선택했다. 또한 MFCC 특징은 음성 인식 시스템에 널리 사용되어 검증되었다. 따라서 본 논문에서는 DTW를 이용하여 인식 시스템을 구현하였다.

1. 핵심 시스템 구조

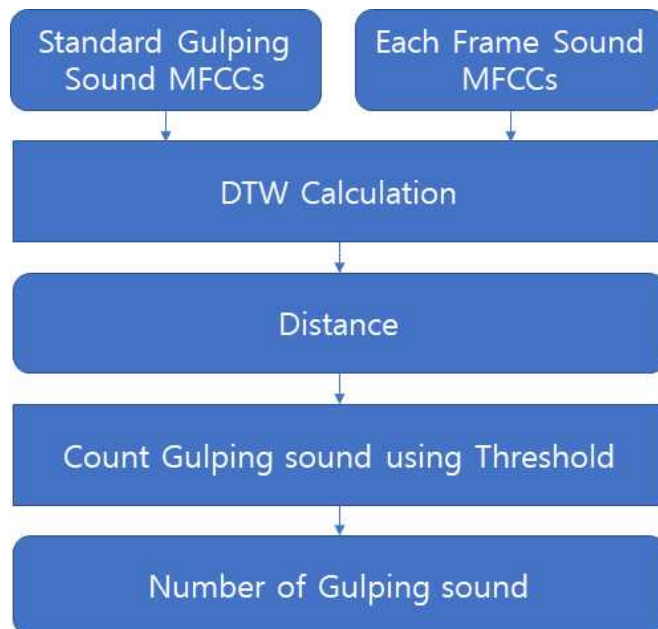


그림 5. 핵심 시스템 다이어그램

처음으로 큰 틀에서 핵심적인 목 넘김 횟수를 세는 방법에 대해서 서술한다. 핵심적인 목 넘김 횟수를 세는 시스템의 구조는 그림 5와 같다.

목 넘김 횟수를 세기 위해서는 목 넘김을 인식하고자 하는 소리에서 목 넘김 소리가 발생한 시점을 찾아내어 그 개수를 세야 한다. 목 넘김을 인식하고자 하는 소리에서 목 넘김이 발생한 시점을 찾기 위해서는 목 넘김을 인식하고자 하는 소리에 대해 전처리 과정을 거친 뒤 작은 구간(프레임)으로 잘라서 목 넘김 소리인지 비교해야 한다. 이 방법을 통해 어떤 시점에서 목 넘김이 발생 했는지 인식할 수 있다.

이 과정을 통해서 많은 수의 프레임을 얻을 수 있다. 이렇게 얻은 많은 프레임에 대해서 MFCC 특징을 구하여 개별 프레임에 대한 MFCC 특징을 얻을 수 있다.

개별 프레임에 대해서 목 넘김 소리인지 비교하기 위해서는 표준 목 넘김 MFCC 특징과 비교하게 된다. 여러 개의 목 넘김 사운드 클립에 대해 전처리 과정을 거친 뒤, MFCC 특징을 구하여 평균을 취하는 방법으로 표준 목 넘김 MFCC 특징을 구한다. 이 방법을 통해 다양한 형태의 목 넘김 소리에 대해 평균적인 특징을 구해 한 개의 목 넘김 소리에 치우치지 않은 목 넘김 특징을 찾을 수 있다.

이렇게 구한 개별 프레임의 MFCC 특징을 우리가 인식하고자 하는 표준 목 넘김 사운드에 대한 MFCC 특징과 DTW 알고리즘을 이용하여 비교한다.

MFCC 특징은 시간 축과 주파수 축을 기준으로 에너지 값을 나타낸 2차원 행렬 구조를 가진다. 또한 앞에서 설명한 DTW 알고리즘을 보면 DTW 알고리즘은 두 신호에 대해서 각 시점간의 거리를 구해서 거리 값이 최소가 되는 시점끼리 맞춰서 거리를 계산하는 알고리즘이다.

목 넘김 소리는 소리가 발생할 때 마다, 소리의 길이와 속도가 일관적이지 않은 특성으로 표준 목 넘김 MFCC 특징과 시점이 일치하지 않는 경우가 때

우 많다. 따라서 DTW 알고리즘을 이용한 비교가 이 문제를 해결하기에 적절하다.

DTW 알고리즘을 통해서 개별 프레임의 MFCC 와 우리가 인식하고자 하는 표준 목 넘김 사운드 간의 거리를 계산할 때, 개별 프레임의 MFCC 특징과 표준 목 넘김 MFCC 신호의 가장 밀접하게 대응하는 지점을 찾기 위해서 시간축을 기준으로 선택된 두 시점의 주파수 대역별 에너지 수치를 비교한다. 선택된 두 시점에서 주파수 대역별 에너지 차이를 맨해튼 노름(manhattan norm)을 이용하여 크기를 구하였다.

$$l^1 = \|x\|_1 = \sum_{i=1}^n |x_i|$$

위 수식은 맨해튼 노름의 수식을 나타낸다. DTW 알고리즘을 통해 선택된 각 두 지점 간의 거리를 구한다고 할 때, i 는 주파수를 나타내고 n 은 주파수의 개수를 나타낸다. 이렇게 구한 거리 값을 이용하여 DTW 알고리즘을 통해 가장 밀접하게 대응하는 지점을 찾고 그 거리를 구할 수 있다.

DTW 알고리즘을 통해 얻은 값은 거리 값으로 낮은 수치를 보일수록 유사하다고 할 수 있다. 따라서 어떤 프레임의 MFCC 특징이 표준 목 넘김 MFCC 특징과 유사하다면 낮은 수치를 보일 것이다.

DTW 알고리즘으로부터 얻은 거리 값으로 어느 시점의 구간에서 목 넘김이 발생했는지 결정을 해야 한다. 앞에서 계산된 모든 프레임의 거리 값들을 시간 및 거리 그래프로 나타내었을 때, 목 넘김 소리가 발생한 구간은 유난히 낮은 수치의 거리를 보여주었다. 이를 이용하여 평균과 편차를 이용하여 유난히 낮은 수치의 거리를 가를 수 있는 Threshold 값을 구할 수 있다.

모든 개별 프레임과 표준 목 넘김 사운드 간의 거리 값들을 가지고 Threshold 미만의 구간과 이상의 구간으로 나눌 수 있다. 이렇게 나뉜 구간 중 Threshold 미만의 구간의 개수를 세어 목 넘김 횟수를 파악할 수 있다. 이

때, Threshold 경계 부근에서 떨리는 현상이 발생할 수 있는데 다음 목 넘김이 발생하기 전까지 걸리는 시간 중 최소의 시간동안 무시를 하는 방법으로 해결할 수 있다.

다음으로 표준 목 넘김 사운드 특징을 어떻게 생성하는지에 대해서 더 자세히 설명하겠다.

2. 표준 목 넘김 사운드 MFCC 특징

본 논문에서 제안한 목 넘김 소리 인식 시스템의 경우 표준 목 넘김 사운드에 대한 MFCC 특징을 기준으로 목 넘김을 인식하고자 하는 사운드로부터 목 넘김 소리를 인식하는 것이다. 따라서 우리가 인식하고자 하는 표준 목 넘김 사운드 MFCC 특징을 올바르게 구하는 것이 매우 중요하다.

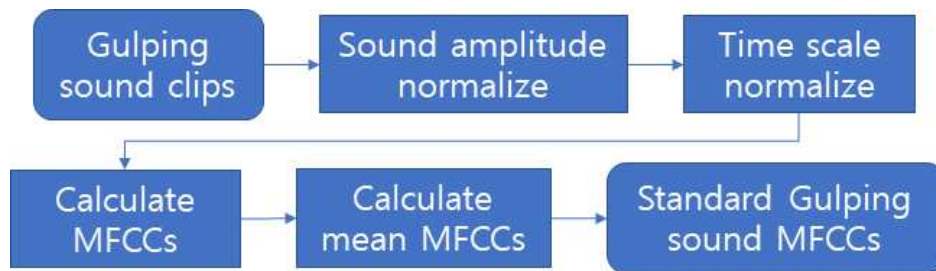


그림 6 표준 목 넘김 사운드 MFCC 특징 구하는 절차

표준 목 넘김 사운드는 그림 6과 같은 절차를 통해 구할 수 있다.

전체 목 넘김 사운드에 대해서 진폭 수준 정규화(normalize)를 적용한 이후, 시간 축 정규화를 다시 진행하여 사이즈를 일정하게 맞춘다. 이렇게 얻어진 일정한 사이즈의 사운드 파일들에 대해 MFCC 특징을 구한 후 전부 더하고

사운드 파일들의 개수만큼 나눔으로서 평균을 구할 수 있다. 이렇게 얻은 MFCC 특징을 표준 목 넘김 사운드 MFCC 특징으로 하였다.

1) 사운드 진폭 수준 정규화[15]

표준 목 넘김 MFCC 특징을 구하기 위해 처음 하는 단계는 진폭 수준에서 정규화를 하는 것이다.

이 과정은 다양한 환경에서 녹음될 수 있는 목 넘김 소리에 대해서 일관성 있는 에너지 크기로 맞추어, 환경에 따른 변화를 최소화하기 위해 필요하다.

이 과정에서 사용된 정규화 기법은 음성 편집 프로그램인 Audacity에서 사용하는 정규화 기법이다. 전체 사운드에서 사운드의 평균을 빼는 방법을 통해 사운드를 0을 기준으로 맞추고, 사운드의 진폭을 $[-1, 1]$ 구간에 맞추므로서 정규화를 진행한다. 이 과정을 아래의 수식으로 표현이 가능하다.

$$y[n] = (x[n] - mean_x) \times ratio$$

사운드 신호 $x[n]$ 에 평균을 빼서 0을 기준으로 표현하도록 맞추게 된다. 사운드의 진폭을 $[-1, 1]$ 구간으로 맞추기 위하여 절댓값이 가장 큰 값을 기준으로 배율 $ratio$ 을 결정한다. 해당 수식은 아래와 같다.

$$ratio = \frac{1}{\max(|x[n]|)}$$

앞에서 구한 배율을 전체 사운드 시그널에 곱함으로써 목 넘김 사운드를 $[-1, 1]$ 구간으로 정규화 할 수 있다.

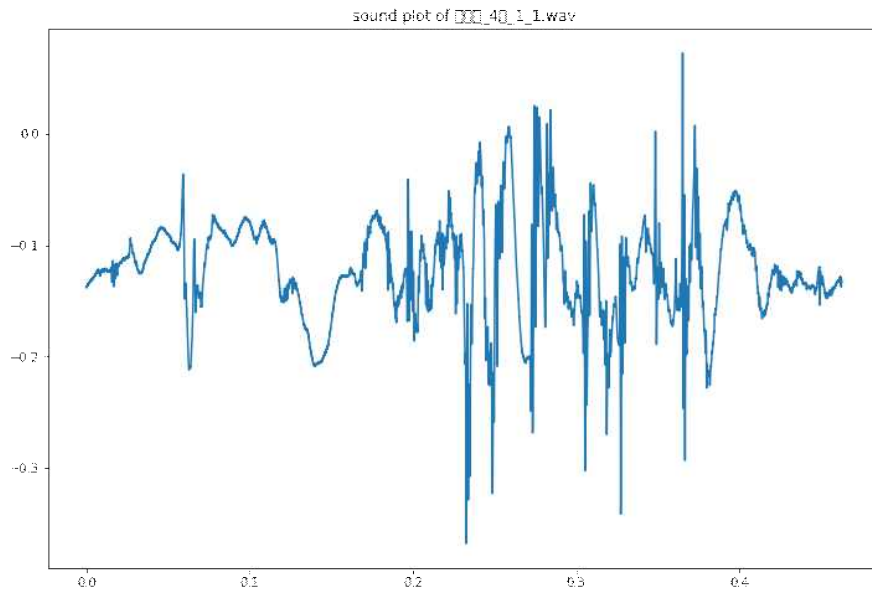


그림 7 정규화 전 음성파일 그래프

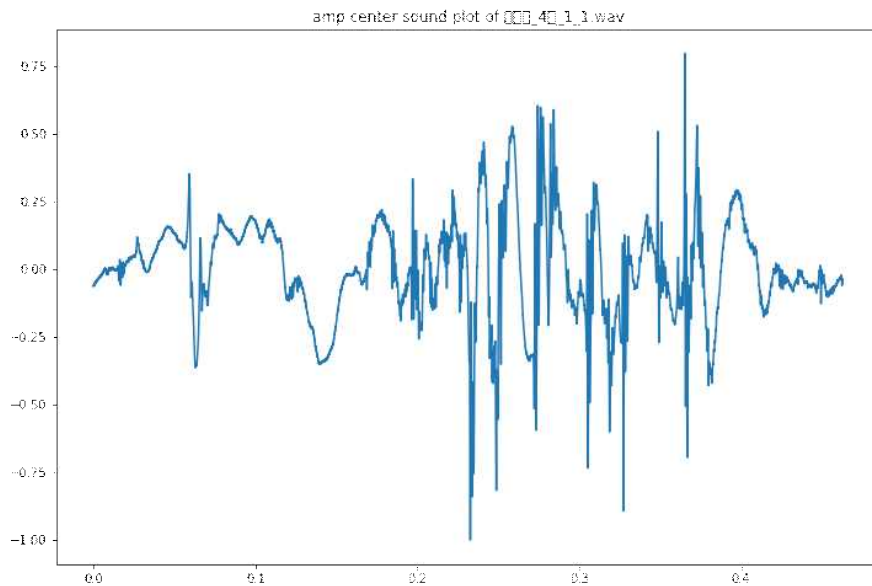


그림 8 정규화 후 음성파일 그래프

그림 7번과 그림 8번의 가로 축은 시간을 의미하며, 세로축은 PCM값이다.

그림 7번과 그림 8번을 보면 명시한 정규화 기법을 사용하여 전체적인 파형을 유지하면서 0을 기준으로 [-1, 1] 구간으로 맞춰서 정규화 하는 것을 잘 보여주고 있다.

이 과정을 통해서 서로 다른 환경에서 녹음된 목 넘김 소리에 대해 에너지

수준을 유사하게 맞춰줄 수 있다. 이 과정을 거치지 않으면, 환경에 따라서 인식이 잘 되지 않았다.

2) 사운드 시간 축 정규화

진폭 수준에서 정규화를 끝낸 목 넘김 소리들은 각자 녹음된 길이가 모두 다른 사운드이다.

대표성 있는 목 넘김 소리는 목 넘김 소리를 평균을 취했을 때 얻을 수 있다. 그러나, 이 과정에서 우리가 가진 목 넘김 사운드들은 길이가 일정하지 않아 평균을 취할 수가 없다.

이 문제를 해결하기 위해서 Audio time scale modification 기술을 적용하여 시간 축에 따라 늘이고 줄이는 작업을 진행하였다. Audio time scale modification 기술의 많은 알고리즘 중, phase vocoder를 이용한 Audio time scale modification 기술을 사용하여 프레임의 크기에 맞추어 늘였다. 이 기술을 사용했을 때, 스펙트럼에서 음성 피치에 대한 변형이 적게 일어나면서 일정한 길이로 늘일 수 있었다. 그러나 우리가 원하는 크기로 딱 맞추어 늘일 수는 없다는 한계가 있었다.

이 과정을 통해 전체적인 길이를 최대한 균일하게 유지할 수 있었다.

3) MFCC 특징 구하기

다음으로는 MFCC(Mel Frequency Cepstral Coefficients) 특징을 구하는 단계이다.

앞에서 설명한 것과 같이, MFCC 특징은 음성 인식을 위해서 가장 많이 사용되는 특징이며, 단순히 시간-주파수 스펙트럼으로 변환하는 기존의 푸리에 변환과는 다르게 주파수 대역을 Mel scale이라는 사람의 지각 특성에 맞춘 단위를 사용하여 좀 더 사람이 듣는 것과 유사하게 나타낸다. 이 과정을 통해서 각 목 넘김 사운드의 길이 차이가 많이 줄어들게 된다.

이 과정을 통해 PCM 으로 표현된 소리를 시간-주파수 축의 2차원 행렬로

표현할 수 있다.

4) MFCC 평균 구하기

시간 축 정규화를 통해 늘인 사운드의 길이가 일정한 길이로 늘어나는 것이 아닌, 유사한 길이로 늘어나서 곧바로 평균을 구하기 어렵다. MFCC 특징을 계산하면, 시간에 따른 길이의 차이가 많이 줄어든다. 이 성질을 이용하여 미리 MFCC 특징을 계산하고 평균을 구하는 것이 유리하다.

또한, 일반적으로 우리가 느끼는 소리는 시간-주파수 축을 기준으로 표현된 MFCC 가 더 일관성 있고 유사하게 표현할 수 있어서 유리하다.

계산된 MFCC 특징에서도 시간 축으로 길이의 차이가 발생하는 데, 여기에서 발생하는 길이의 편차가 매우 작다. 따라서 우리는 이런 성질을 이용하여 가장 짧은 MFCC 특징에 맞추어 모든 MFCC 특징들의 길이를 잘랐다. 특징을 가장 짧은 MFCC 특징에 맞추어 정보의 손실이 발생했지만, 큰 차이나 나타나지 않았기 때문에 큰 문제가 되지는 않았다.

이렇게 자른 MFCC 특징들을 모두 더한 후, MFCC 특징들의 개수로 나누어 평균적인 MFCC 특징을 구할 수 있었다.

앞에서 설명한 방법으로 구한 MFCC 특징을 표준 목 넘김 사운드 MFCC 특징으로 삼았다.

3. 개별 프레임 사운드 MFCC

본 절에서는 목 넘김 횟수를 판별하고자 하는 사운드에 대해서 어떤 방식을 통해 개별 프레임에 대한 MFCC 특징을 구하는지 설명한다.

앞에서 설명한 것처럼, 목 넘김을 인식하고자 하는 사운드에 대해서 인식한 횟수를 세기 위해서는 어떤 시점에서 목 넘김 소리가 발생했는지 인식할 수 있어야 한다. 이를 위해서는 목 넘김을 인식하고자 하는 사운드를 여러 프레

임으로 쪼개서 표준 목 넘김 소리와 비교를 해야 한다.

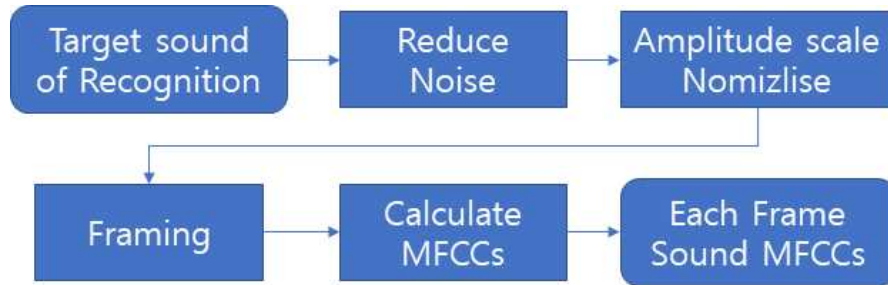


그림 9 개별 프레임 MFCC 특징 구하기

그림 9에서는 목 넘김을 인식하고자 하는 사운드로부터 개별 프레임에 대한 MFCC 특징을 구하는 단계에 대해 나타내고 있다. 그림 9과 같이 목 넘김 인식 대상 사운드 파일에 대해서 노이즈를 제거하고 진폭 수준의 정규화를 거친 이후 개별 프레임으로 쪼개어 MFCC 특징을 계산하는 방법으로 개별 프레임에 대한 MFCC 특징을 계산한다.

1) 노이즈 제거

본 논문에서 가장 문제로 생각하고 있는 노이즈는 축사 안에서 지속적으로 돌아가는 Fan sound로, 규칙적이면서 지속적으로 발생하는 백색 잡음(white noise)의 성격을 띤다. 또한 해당 노이즈의 크기가 목 넘김 소리를 구분하기 어려운 정도로 크기 때문에 해당 노이즈를 제거하는 것이 필수적이다.

노이즈 제거 단계에서는 오디오 편집 프로그램인 Audacity에서 사용하는 노이즈 제거 알고리즘을 채택하여 사용하였다. 해당 알고리즘은 일반적으로 Spectral Noise Gate라고 불리는 기술이다[8,10].

Spectral Noise Gate는 노이즈의 평균적인 에너지와 편차를 이용하여 Threshold 보다 작은, 잡음에 해당하는 부분의 에너지를 줄이기 때문에 Fan sound와 같이 지속적이고 규칙적으로 나타나는 잡음을 제거하기에 알맞다.

fan sound를 노이즈의 표본으로 설정하고 민감도를 1.5 정도로 설정하여 노이즈를 제거할 수 있다.

2) 진폭 수준 정규화

개별 Frame을 생성하기 전에 전체적인 에너지 수준을 맞춰주기 위해서 앞에서 사용하였던 정규화 기법을 사용하였다. 이 과정을 통해서 에너지 수준을 대등하게 맞춰줌으로서 비교할 수 있는 조건을 만들 수 있다.

이 과정을 적용하지 않았을 때, DTW 비교의 원리 상 두 시점의 주파수별 에너지 값의 차이를 이용하여 거리 값을 구하기 때문에 올바른 비교를 하지 못하고 한쪽의 에너지 추세를 따라가는 현상을 보인다.

이 정규화 수식은 앞에서 설명한 표준 목 넘김 사운드 MFCC 계산 시 사용한 정규화 기법과 동일하다.

3) Framing (개별 프레임 생성)

다음 순서로 목 넘김 소리가 어디에서 발생하였는지 인식하기 위해서 프레임링 기술을 사용하였다. 전체 판별하고자 하는 사운드에서 평균적으로 목 넘김 소리가 나타나는 구간을 설정하여 일반적으로 단 구간으로 사용되는 10ms 단위로 윈도우를 옮기면서 프레임을 생성하였다.

평균적으로 목 넘김 소리가 나타나는 시간 구간을 프레임(윈도우)의 사이즈로 설정하였고, 이를 계산하기 위한 수식은 다음과 같다.

$$FrameSize = Mean(GS\ size) + \alpha \times Std(GS\ size)$$

GS size는 목 넘김 소리의 사이즈 이다. α 는 표준분포의 신뢰도를 결정하는 상수이다. 본 논문에서는 α 값을 2.6 으로 설정하였다. 프레임을 자를 때 목 넘김 소리 구간이 프레임에 전부 들어갈 수 있는 수준으로 설정하는 것이 중요하다. 이 값은, 목 넘김 소리의 사이즈 값을 정규분포를 따른다고 가정을 했을 때, 신뢰도 99%의 구간 안에 목 넘김 소리가 들어갈 수 있도록 설정하

였다. 이를 통해 실제 목 넘김 사운드 길이의 99% 신뢰도 구간으로 목 넘김 사운드를 포함 할 수 있는 프레임 크기를 설정 할 수 있었다.

4) MFCC 특징 계산

위 단계를 끝내면 많은 수의 프레임들을 얻을 수 있다. 본 단계에서는 개별 프레임에 각각 MFCC 특징을 계산한다. 대상만 개별 프레임으로 바뀌었을 뿐, 위에서 표준 목 넘김 MFCC 특징을 계산할 때와 동일하게 계산한다.

이 과정을 통해 시간-주파수 축으로 이루어진 2차원 행렬로 된 MFCC 특징을 구할 수 있었다.

위 과정들을 통해 핵심 시스템에서 DTW 알고리즘을 이용한 비교를 하기 위한 개별 프레임의 MFCC 특징을 구할 수 있다.

4. DTW 계산 및 Threshold를 이용한 목 넘김 횟수 세기

위 단계까지 끝난 개별 프레임의 MFCC 특징들은 표준 목 넘김 MFCC 특징과 DTW 계산을 통해 전체 프레임에 대한 유사도를 판별할 수 있는 지표를 계산할 수 있다.

1) DTW 계산 및 Threshold 결정[16]

DTW에 사용한 거리(Cost, Distance) 함수는 MFCC 특징의 시간 축을 따라 각 시점을 결정하고 각 시점에서의 주파수 대역별로 에너지를 빼서 이를 맨해튼 노름을 이용하여 거리를 구하였다. 이를 통해서 비교대상인 개별 프레임 MFCC 특징과 표준 목 넘김 MFCC 특징에 대해서 DTW 거리표를 만들었다. 각 시점의 주파수별 에너지 값들을 하나의 벡터로 표현하여 벡터 뺄셈 후, 맨해튼 노름(manhattan norm)을 구했다고 표현할 수도 있다.

이렇게 구해진 거리 값을 DTW의 두 시점에 따른 거리로 취급하여 최소의 거리 값을 가지는 Warping path 라고 부르는 가장 올바른 대응 시점들을 찾을 수 있고, 이때의 거리를 구할 수 있다.

이 과정을 개별 프레임 MFCC 특징 계산 과정에서 계산된 모든 프레임의 MFCC 특징들에 대해서 계산함으로써 우리가 목 넘김 소리를 인식하고자 하는 소리에 대해서 시간에 따른 거리 값들을 얻을 수 있다.

이렇게 얻은 거리 값들에 대해서 Threshold를 이용하여 목 넘김을 인식 하기 위해서 일정한 범위의 거리 값으로 정규화를 진행해야 일관성 있는 Threshold를 결정할 수 있고, 이를 통해서 시스템을 구성할 수 있다. 이에 MinMax Normalize를 적용하여 [0, 1] 범위에서 표현되도록 하였다.

$$y(n) = \frac{x(n) - \min(x(n))}{\max(x(n)) - \min(x(n))}$$

위 수식은 MinMax Normalize에 대한 수식이다. DTW 알고리즘을 통해 얻은 거리 값들에 대해 이 수식을 적용하여 [0, 1] 범위에서 표현하여 일관성 있는 Threshold를 결정하는데 도움이 되었다.

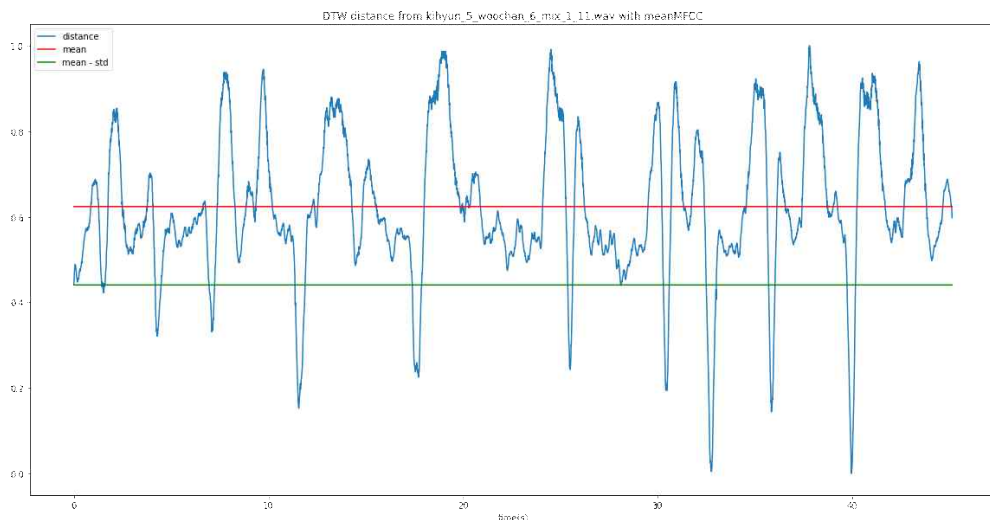


그림 10 DTW 계산된 거리 수치 및 평균과 Threshold 값 비교

그림 10은 모든 개별 프레임의 MFCC와 표준 목넘김 MFCC를 비교하여 시간에 따른 거리 값을 MinMax Normalize 하여 그래프로 그린 그림이다. 그림 10을 보면 목 넘김 소리에 해당하는 부분은 거리가 상대적으로 낮게 나오는 것을 알 수 있다.

앞에서 설명한 것처럼 목 넘김이 발생한 구간을 결정하기 위해서는 Threshold를 적절하게 선택하여 결정해야 한다. 그림 10에서 볼 수 있는 것처럼 목 넘김 소리가 발생한 시점에서 거리 값이 유난히 내려가는 특징을 볼 수 있다. 이에 Threshold를 평균과 표준편차를 이용하여 구하였다.

$$Threshold = mean_{distance} - std_{distance} \times \alpha$$

그림 10의 빨간 선은 거리의 평균값, 녹색 선은 위 수식을 적용하여 계산한 Threshold 값을 나타내는 수평선이다. 휴리스틱 방법론적으로 α 값을 조절하면서 적절한 Threshold 값을 구할 수 있다. 본 논문에서는 α 값을 1.1로 설정하였다.

2) Threshold 및 목 넘김 횟수 세기(count)

앞에서 결정한 Threshold 값을 기준으로 Threshold 값 미만의 구간 개수를 세는 방법을 통해 목 넘김 구간을 구할 수 있다.

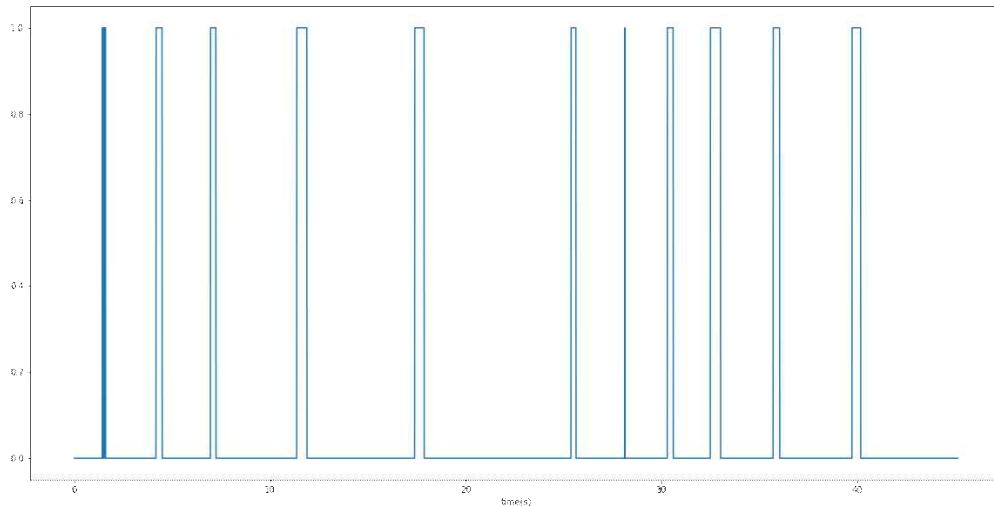


그림 11 Threshold 값을 이용하여 구한 구간

그림 11과 같이 Threshold 미만의 값들을 1로 나머지 값들을 0으로 처리함으로써 목 넘김이 발생한 구간을 찾아낼 수 있다.

기본적으로 목 넘김이 발생한 구간을 셀 때에는 0에서 1로 변하는 지점을 인식하여 세고 있다. 그림 11의 첫 번째 구간을 보면, 구간 사이에 파랗게 채워져 보이는 부분이 존재하는데, 이는 Threshold 값 기준으로 진동이 발생하는 구간이다. 이 경우, 1번의 목 넘김만 발생했지만, 여러 번의 목 넘김이 발생했다고 할 수 있어서 일정 시간동안 무시하는 시간이 필요하다.

목 넘김 횟수를 셀 때 무시하는 시간은 다음 목 넘김이 발생하기까지 걸리는 시간이다. 본 논문에서는 목 넘김 소리의 길이로 판단되는 0.451s 동안은 다음 목 넘김이 발생하지 않을 것으로 판단하였다.

1에서 0으로 떨어지는 지점으로부터 앞에서 결정한 시간만큼 무시하는 방법을 통해 중복으로 목 넘김 횟수를 세는 문제를 해결할 수 있다.

IV. 구현 및 평가

1. 구현 및 실험 환경

제안된 목 넘김 인식 시스템을 구현하고 평가하기 위해서 녹음기 및 웨어러블 디바이스를 통해 물을 마시면서 목 넘김 소리를 녹음하고, 목 넘김 소리가 있는 구간만 잘라 목 넘김 사운드를 만들었다.

녹음기는 idam 의 K10 Pro 16G 라는 명칭의 녹음기로, 48KHz PCM WAV 를 지원하고, MP3 녹음을 지원하는 고성능의 녹음기이다.

녹음은 44.1KHz 16bit Mono 녹음으로 세팅하여 녹음하였다.



그림 12 Idam K10-PRO 녹음기

웨어러블 디바이스는 Espressif사의 ESP32 마이크로컨트롤러 칩을 사용한 디바이스이다.

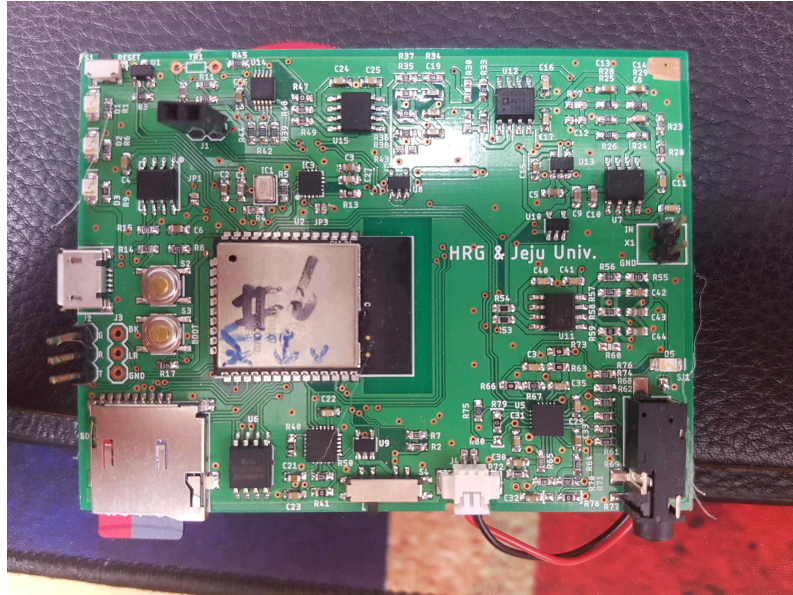


그림 13 웨어러블 디바이스 전면

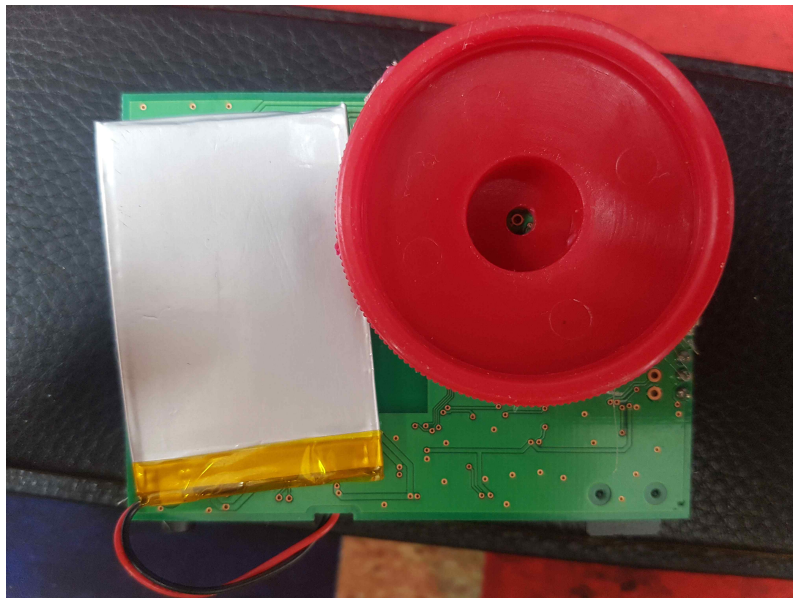


그림 14 웨어러블 디바이스 후면

웨어러블 디바이스는 블루투스 및 WiFi를 이용한 무선 통신이 가능하고, MicroSD 카드를 통한 정보 저장이 가능하다. Mems Mic가 탑재되어 I2S프로토콜을 통한 소리 녹음이 가능하다.

본 논문에서 웨어러블 디바이스를 이용하여 44.1Khz 16bit Wav 형태로 소리를 녹음하였다.

측사 환경에서 주로 나타나는 환풍기 돌아가는 소리 및 새 지저귀는 소리를 유튜브에서 ‘Relaxing White Noise’의 “REALLY AWESOME FAN SOUND FOR SLEEP | White Noise For Superb Slumber, Studying & Relaxation”이라는 동영상에 대해 녹음하여 Fan 소리를 녹음하였고, ‘1HarryH’의 “4 HOURS Natural sounds: Morning Birds singing - NO MUSIC”이라는 동영상에 대해 녹음하여 새가 지저귀는 소리를 녹음할 수 있었다.

이후 그림 15와 같이 Audacity라는 음성 편집 프로그램을 이용하여 목 넘김 사운드 클립들과 쇠 부딪히는 소리, 새 지저귀는 소리를 적절히 배치한 후 white noise 로 팬이 돌아가는 소리를 배치하였다.

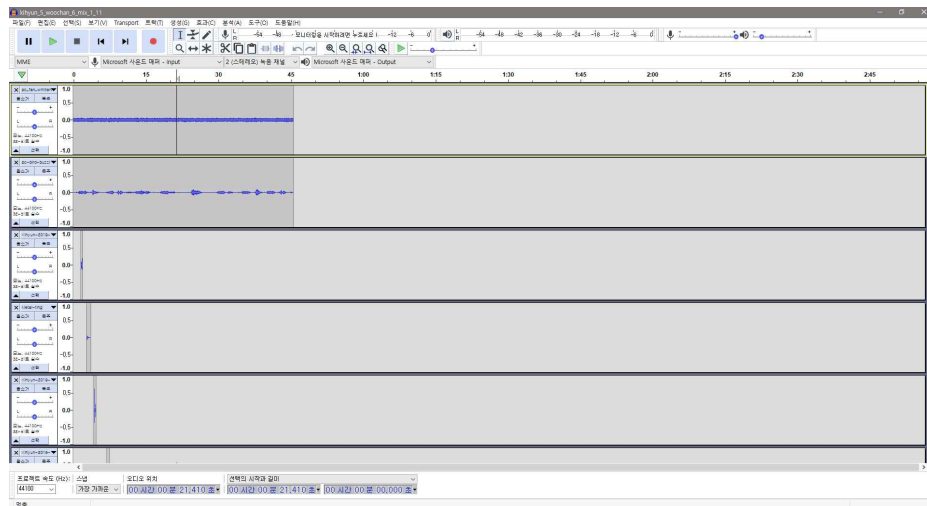


그림 15 목 넘김 사운드 믹스

이렇게 생성된 사운드가 목 넘김을 인식하기 위한 사운드 파일이다.

제안 시스템의 경우 Python을 이용한 데이터 분석 툴로서 많이 사용되고 있는 Jupyter Notebook 이라는 환경에서 구현 및 실험을 진행하였다.

시스템을 구현하기 위해 사용한 라이브러리는 다음의 표 3에서 설명하였다.

표 3 구현 시 사용한 라이브러리 목록

이름	목적
librosa	사운드 파일 로드 및 MFCC 특징 계산
pierre-rouanet/dtw	DTW 계산 알고리즘 구현체
timsainb/noisereduce	노이즈 제거 알고리즘 구현체
numpy	벡터 및 행렬 계산 단순화
Muges/audiotism	Audio time scale modification 구현체

2. 평가

본 논문에서 제안된 시스템에 대해 평가하기 위해서 실제 목 넘김이 일어난 횟수와 시스템이 인식한 목 넘김 횟수를 비교하는 방식으로 진행하였다.

1) 웨어러블 디바이스를 이용한 녹음

웨어러블 디바이스를 통해 녹음한 목 넘김 소리와 팬 돌아가는 소리, 쇠 부딪치는 소리, 새 지저귀는 소리를 녹음하여 믹스하였다.

녹음된 목 넘김 소리는 목 넘김이 발생한 구간만 잘라서 클립으로 만들었다. 이렇게 만들어진 클립의 개수는 3명의 연구원이 각각 12개씩 목 넘김 소리를 얻을 수 있었다.

표준 목 넘김 사운드는 앞에서 얻은 총 36개의 목 넘김 소리 클립을 가지고 생성하였다.

목 넘김 소리 클립과 각종 잡음을 이용하여 생성한 목 넘김 인식 대상 사운드 파일을 다음의 표 4와 같이 생성하였다.

표 4 웨어러블 디바이스 녹음 사운드를 이용하여 생성한 목 넘김 인식 대상 사운드

이름	목 넘김 횟수	설명
mixing-gulp-fan	5회	목 넘김 5회 및 팬 소음
mixing-gulp-fan-metal-buzz	5회	목 넘김 5회 및 팬 소음, 쇠 부딪히는 소음, 새 지저귀는 소음
jihoon_and_key_fan_metal_buzz_noise_mix_24	24회	두 연구원의 목 넘김 각각 12회 및 팬 소음, 쇠 부딪히는 소음, 새 지저귀는 소음

이렇게 생성된 목 넘김 인식 대상 사운드에 대해서 제안된 시스템을 이용하여 인식을 진행하였다.

표 5 웨어러블 디바이스 녹음 사운드를 이용하여 생성한 목 넘김 인식 결과

이름	목 넘김 횟수	인식 횟수	오차	정확도
mixing-gulp-fan	5회	5회	0회	100%
mixing-gulp-fan-metal-buzz	5회	6회	1회	80%
jihoon_and_key_fan_metal_buzz_noise_mix_24	24회	28회	4회	83%

웨어러블 디바이스를 통해 녹음한 목 넘김 소리와 쇠 부딪히는 소리, 팬 돌아가는 소리, 새 지저귀는 소리를 합쳐서 인식 실험을 진행하여 표와 같은 결과를 얻을 수 있었다. 이 실험 결과를 분석하면서 문제가 발생할 수 있을만한 부분이 있었다.

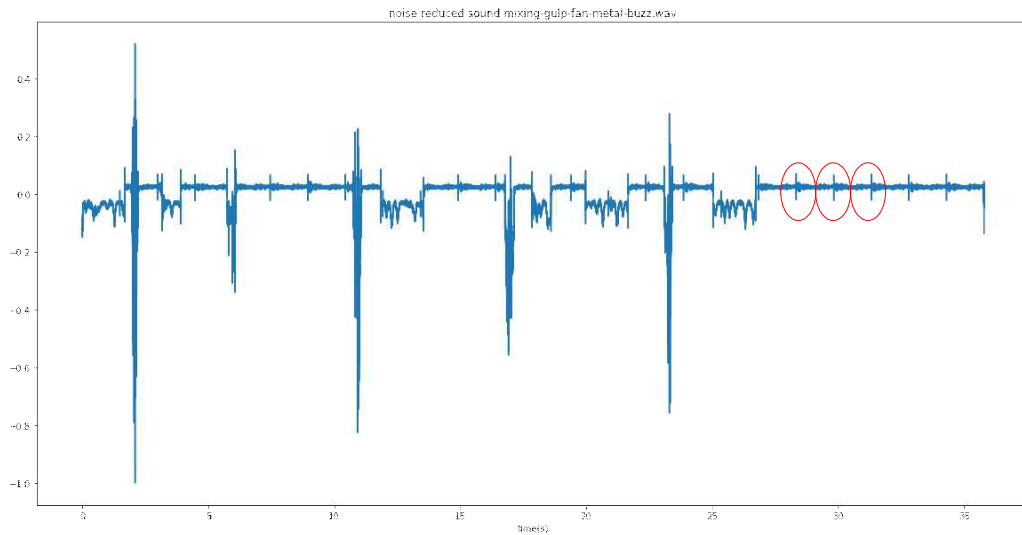


그림 16 mixing-gulp-fan-metal-buzz 사운드 파일의 노이즈 제거 후 그래프

그림 16를 보면 주기적으로 조그마하게 신호가 나타나는 것을 볼 수 있다. 빨간색 타원으로 감싸져 있는 패턴이 그것이다. 이 패턴이 나타나는 이유로 판단되는 원인은 팬 돌아가는 소리를 배경잡음으로 생성할 때, 일부 짧은 구간의 팬 돌아가는 소리를 가지고 복사 붙여넣기를 반복하여 생성하였기 때문에 Phase가 맞지 않는 부분이 잡음으로서 나타날 수 있다고 보았다.

이번에는 팬 돌아가는 소리를 길게 녹음하여 복사 및 붙여넣기 하는 과정에서 발생할 수 있는 불일치를 없앤 상태에서 실험을 다시 진행하였다.

표 6 웨어러블 디바이스 녹음 사운드를 이용하여 생성한 목 넘김 인식 대상 사운드

이름	목 넘김 횟수	설명
mixing-gulp-fan-metal-buzz_new	5회	목 넘김 5회 및 팬 소음, 쇠 부딪히는 소음, 새 지지귀는 소음
jihoon_and_key_fan_metal_buzz_noise_mix_new_24	24회	두 연구원의 목 넘김 각각 12회 및 팬 소음, 쇠 부딪히는 소음, 새 지지귀는 소음

표 6과 같이 다시 생성한 목 넘김 인식 대상 사운드 파일에 시스템을 이용하여 인식해보았다.

표 7 웨어러블 디바이스 녹음 사운드를 이용하여 생성한 목 넘김 인식 결과

이름	목 넘김 횟수	인식 횟수	오차	정확도
mixing-gulp-fan-metal-buzz_new	5회	6회	1회	80%
jihoon_and_key_fan_metal_buzz_noise_mix_new_24	24회	31회	7회	71%

제안 시스템을 이용하여 목 넘김 인식 대상 사운드에 대해서 인식을 진행한 결과, 표 7과 같은 결과를 얻을 수 있었다. 결과를 지켜보면 오히려 정확도가 떨어지는 결과를 얻을 수 있었다.

새로 생성 시 사용한 충분히 길게 녹음한 팬 돌아가는 소리를 들어보았을 때, 팬이 돌아가는 소리에 추가적으로 스파크 소리와 비슷한 잡음이 섞여 들어 있었다. 이런 잡음의 영향에서 자유로운 실험을 위해서 곰 녹음기를 통해서 충분히 긴 팬 돌아가는 소리를 녹음하였다. 또한 곰 녹음기를 통해서 얻은 팬 돌아가는 소리를 이용하여 새로운 목 넘김 인식 대상 사운드를 믹스하였다.

표 8 웨어러블 디바이스 녹음 사운드 및 곰 녹음기를 통해 얻은 팬 돌아가는 사운드를 이용하여 생성한 목 넘김 인식 대상 사운드

이름	목 넘김 횟수	설명
mixing-gulp-fan-metal-buz z_new_pc	5회	목 넘김 5회 및 팬 소음, 쇠 부딪히는 소음, 새 지지귀는 소음
jihoon_and_key_fan_metal_b uzz_noise_mix_new_pc	24회	두 연구원의 목 넘김 각각 12회 및 팬 소음, 쇠 부딪히는 소음, 새 지지귀는 소음

표 8과 같이 생성된 목 넘김 인식 대상에 대해서 제안된 시스템을 이용하여 목 넘김 인식을 진행하였다.

표 9 웨어러블 디바이스 녹음 사운드 및 곰 녹음기를 통해 얻은 팬 돌아가는 사운드를 이용하여 생성한 목 넘김 인식 결과

이름	목 넘김 횟수	인식 횟수	오차	정확도
mixing-gulp-fan-metal-buz z_new_pc	5회	6회	1회	80%
jihoon_and_key_fan_metal_b uzz_noise_mix_new_pc	24회	27회	3회	87%

곰 녹음기를 통해 얻은 충분히 긴 팬 돌아가는 소리를 이용하여 생성한 목 넘김 인식 사운드에 대해서 목 넘김 인식을 진행했을 때, 조금 더 좋은 결과를 얻을 수 있었다.

2) 녹음기를 이용한 녹음

녹음기를 이용하여 조용한 환경에서 목 넘김 소리를 녹음하고, 쇠 부딪히는 소리, 새 지지귀는 소리, 팬 돌아가는 소리는 곰 녹음기를 통해 녹음하였다.

녹음된 목 넘김 사운드 개수는 43개로 한 연구원은 11번 한 연구원은 32번 녹음하였다. 이렇게 녹음된 목 넘김 사운드 43개를 이용하여 표준 목 넘김 사운드를 생성하였다.

이후 앞에서 수집한 목 넘김 사운드 및 팬 돌아가는 소리, 새 지저귀는 소리, 쇠 부딪히는 소리를 이용하여 목 넘김 인식 대상 사운드 파일을 생성하였다.

표 10 녹음기를 통해 얻은 목 넘김 소리 및 곰 녹음기를 통해 얻은 팬 돌아가는 사운드를 이용하여 생성한 목 넘김 인식 대상 사운드

이름	목 넘김 횟수	설명
woochan_mix_1_5	5회	목 넘김 5회 및 팬 소음, 쇠 부딪히는 소음, 새 지저귀는 소음
kihyun_mix_1_4	4회	목 넘김 4회 및 팬 소음, 쇠 부딪히는 소음, 새 지저귀는 소음
kihyun_woochan_mix_1_6	6회	두 연구원 목 넘김 각 3회, 팬 소음, 쇠 부딪히는 소음, 새 지저귀는 소음
kihyun_5_woochan_6_mix_1_11	11회	한 연구원 목 넘김 5회 다른 연구원 목 넘김 6회, 팬 소음, 쇠 부딪히는 소음, 새 지저귀는 소음

이후 생성된 목 넘김 인식 대상 사운드는 표 10에 잘 나와 있다. 목 넘김 인식 대상 사운드에 대해서 제안된 시스템을 이용하여 목 넘김 인식을 진행하였다.

표 11 웨어러블 디바이스 녹음 사운드 및 곰 녹음기를 통해 얻은 팬 돌아가는 사운드를 이용하여 생성한 목 넘김 인식 결과

이름	목 넘김 횟수	인식 횟수	오차	정확도
woochan_mix_1_5	5회	5회	0회	100%
kihyun_mix_1_4	4회	4회	0회	100%
kihyun_woochan_mix_1_6	6회	6회	0회	100%
kihyun_5_woochan_6_mix_1_11	11회	11회	0회	100%

인식한 횟수를 비교해 보았을 때 정확한 결과를 얻을 수 있었다.

총 실험 결과를 정리하면 다음의 표와 같다.

표 12 최종 목 넘김 인식 결과

이름	목 넘김 횟수	인식 횟수	오차	정확도
mixing-gulp-fan	5회	5회	0회	100%
mixing-gulp-fan-metal-buzz	5회	6회	1회	80%
jihoon_and_key_fan_metal_buzz_noise_mix_24	24회	28회	4회	83%
mixing-gulp-fan-metal-buzz_new	5회	6회	1회	80%
jihoon_and_key_fan_metal_buzz_noise_mix_new_24	24회	31회	7회	71%
mixing-gulp-fan-metal-buzz_new_pc	5회	6회	1회	80%
jihoon_and_key_fan_metal_buzz_noise_mix_new_pc	24회	27회	3회	87%
woochan_mix_1_5	5회	5회	0회	100%
kihyun_mix_1_4	4회	4회	0회	100%
kihyun_woochan_mix_1_6	6회	6회	0회	100%
kihyun_5_woochan_6_mix_1_11	11회	11회	0회	100%
총합	118회	135회	17회	86%

V. 결론

우리는 실험을 통해서 백색잡음 제거 후 DTW를 이용하여 목 넘김 소리와 팬 돌아가는 소리, 쇠 부딪히는 소리, 새 지저귀는 소리를 믹스하여 인식 실험을 진행하였다. 목 넘김 소리 인식 대상 사운드에 제안한 시스템을 이용하여 목 넘김 인식을 수행하였을 때, 총 118회 목 넘김 소리에 대해서 총 135회로 목 넘김이 발생하였다고 인식하였다. 실제 목 넘김 소리 횟수와 17회 정도의 차이를 보였고 대략 86%의 정확도를 얻을 수 있었다. 이 결과를 통해 제안된 시스템을 이용하여 목 넘김 횟수를 셀 수 있음을 보일 수 있었다.

기존에 사용되던 전자저울은 사료통에 설치되어 개별 소에 대한 사료 섭취량을 알기 어려운 점이 있다. 이에 비해서 웨어러블 디바이스를 통해서 녹음한 목 넘김 소리 인식 시스템을 적용한다면, 개별 소에 대해서 섭취량을 파악할 수 있는 기본 자료를 제공해 줄 수 있을 것으로 판단한다.

본 논문에서 소에 대해서 목 넘김을 인식하기 위한 적절한 데이터를 구하지 못하였다. 이에 사람의 목 넘김 소리에 대해서 축사 환경과 유사한 주변 잡음을 혼합한 인위적인 사운드에 대해서 실험을 진행하였다. 따라서 실제 축사 환경에서 소의 목 넘김에 대해 실험을 해야 한다.

실제 축사 환경에서는 소의 평균적인 목 넘김 길이, 평균적인 목 넘김 사이 간격 등 유용한 데이터를 얻을 수 있을 것으로 기대한다.

추가적으로 개별 소에 대해서 목 넘김 횟수를 통해 섭취량을 추정하는 실험이 더 필요하다. 섭취량은 소의 발정기 및 질병과 연관성이 있다. 따라서 섭취량과 소의 발정기에 대한 연구를 진행하여 발정기 및 질병을 진단할 수 있을

것이다.

웨어블 디바이스를 이용하여 자동화된 시스템을 통해 병이나 발정기를 조기에 진단이 가능하다면 축산업의 생산 효율성이 크게 증대될 것으로 기대된다. 또한 조기에 질병을 진단함으로써 질병의 확대를 방지할 수 있어 다수의 가축이 폐사하는 위험에 대해서 빠르게 대응할 수 있다는 점에서 중요하다.

참 고 문 헌

- [1] Ho-young Kwak, Woo-chan Kim and Jin-Wook Jang, "A Method of White Noise Reduction for Recognizing Cattle's Gulp Downing Sounds", Journal of The Korea Society of Computer and Information Vol. 24 No. 11, pp. 153-161, November 2019
- [2] 이창우, "DTW를 이용한 윈도우기반 명령어의 음성인식 구현", 광운대학교전산대학원 컴퓨터공학과 석사학위논문 1996.
- [3] Fu, Wenjie, Xinghai Yang, and Yutai Wang. "Heart sound diagnosis based on DTW and MFCC." 2010 3rd International Congress on Image and Signal Processing. Vol. 6. IEEE, 2010.
- [4] Mohan, Bhadrageiri Jagan. "Speech recognition using MFCC and DTW." 2014 International Conference on Advances in Electrical Engineering (ICAEE). IEEE, 2014.
- [5] Molau, Sirko, et al. "Computing mel-frequency cepstral coefficients on the power spectrum." 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221). Vol. 1. IEEE, 2001.
- [6] John Coleman, "Introducing Speech and Language Processing" Hankookmunhwasa Publishing Company, 2009
- [7] Driedger, Jonathan; Müller, Meinard. 2016. "A Review of Time-Scale Modification of Music Signals." Appl. Sci. 6, no. 2: 57.
- [8] Audacity, "How Audacity Noise Reduction Works", https://wiki.audacityteam.org/wiki/How_Audacity_Noise_Reduction_Works#

algorithm

- [9] Wikipedia, “Noise gate”, https://en.wikipedia.org/wiki/Noise_gate
- [10] Tim Sainburg, “Noise reduction using spectral gating in python”,
<https://timsainburg.com/noise-reduction-python.html>
- [11] Wikipedia, “Mel-frequency cepstrum”,
https://en.wikipedia.org/wiki/Mel-frequency_cepstrum
- [12] Wikipedia, “Mel scale”, https://en.wikipedia.org/wiki/Mel_scale
- [13] Wikipedia, “Dynamic time warping”,
https://en.wikipedia.org/wiki/Dynamic_time_warping
- [14] Muges, “A real-time audio time-scale modification library”,
<https://audiotsm.readthedocs.io/en/latest/>
- [15] Audacity, “Normalize”,
<https://manual.audacityteam.org/man/normalize.html>
- [16] pierre-rouanet, “simple speech recognition”,
<https://github.com/pierre-rouanet/dtw/blob/master/examples/speech-recognition.ipynb>