MiraeCity
: Rail Robot AI Interface for Improving User Usability

MILab Undergraduate student, Kim Taehyeon

2023. 11. 14

목차

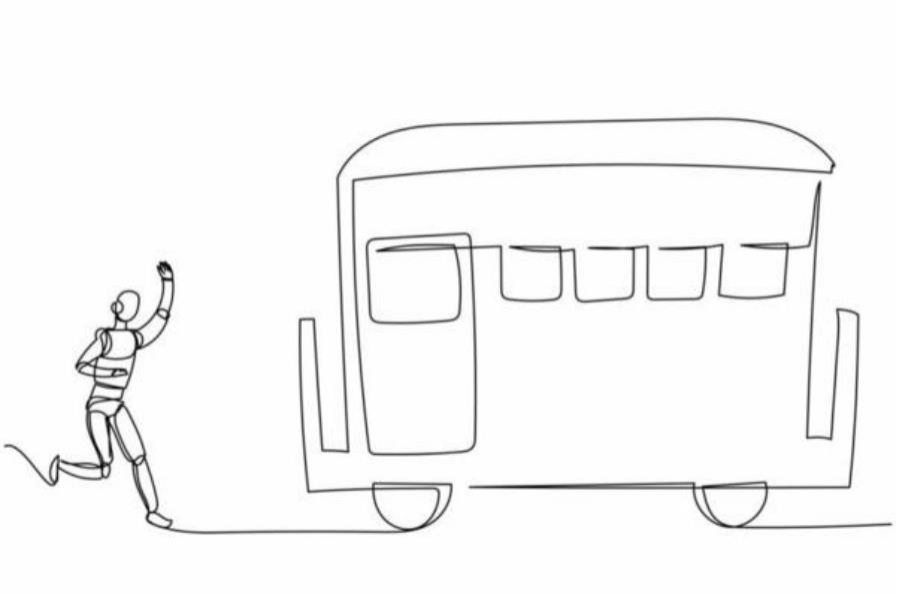
1. 서론

1. 미래시티는 무슨 회사인가?

사업 아이템으로는, 360도 VR/AR 영상진단용 개발, 교량계측관리 센서분야, BIG DATA & SOLUTION, 스마트오피스를 위한 IoT 플랫폼이 있음.

1. 서론

2. 무엇을 만들고자 하는가?

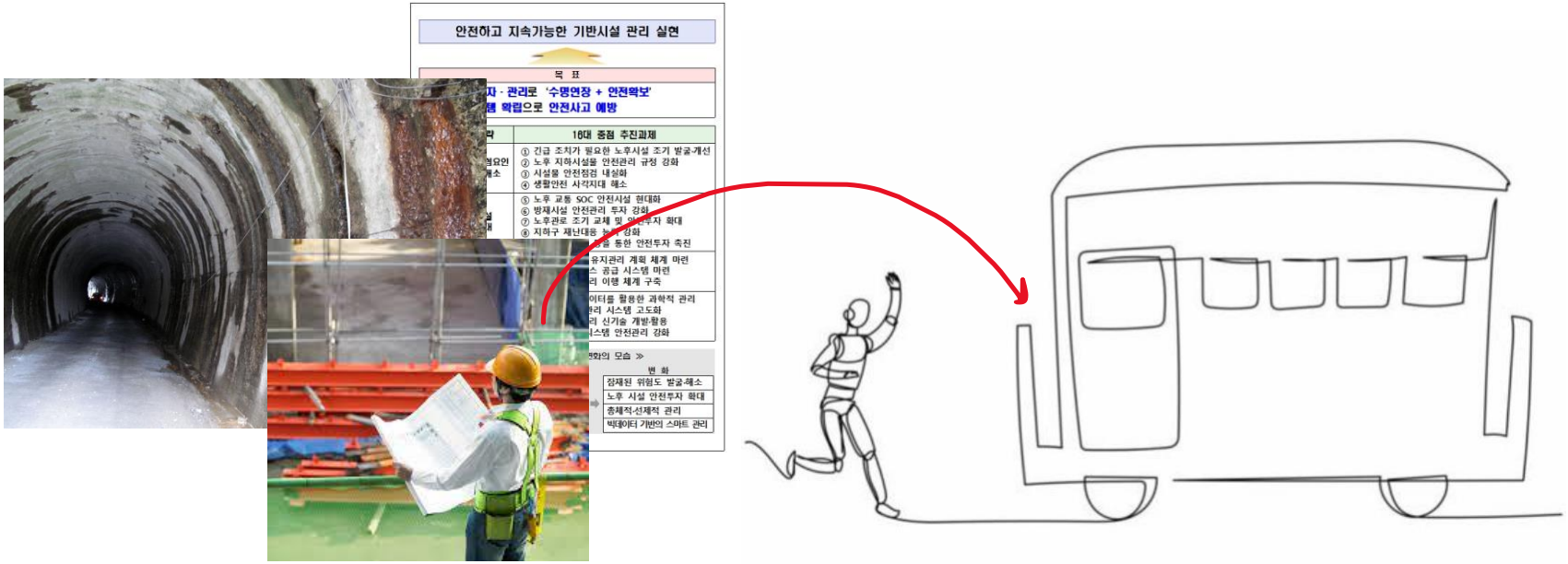


새로운 접근으로 터널의 점검을 위한 레일로봇의 개발로 이어지게 됨.

3인팀으로 레일로봇의 AI 명령어 인식 시스템의 개발을 시작하게 되었음.

1. 서론

3. 문제 상황 인식



터널에도 시간에 따른 부식 등 손상이 발생한다. 이러한 문제로 인하여 주기적인 점검이 필요하다. 하지만, 인부가 투입되어야 하는 단점이 있었음.

이러한 점을 바탕으로 **로봇을 도입**하게 되었음.

1. 서론

3. 문제 상황 인식

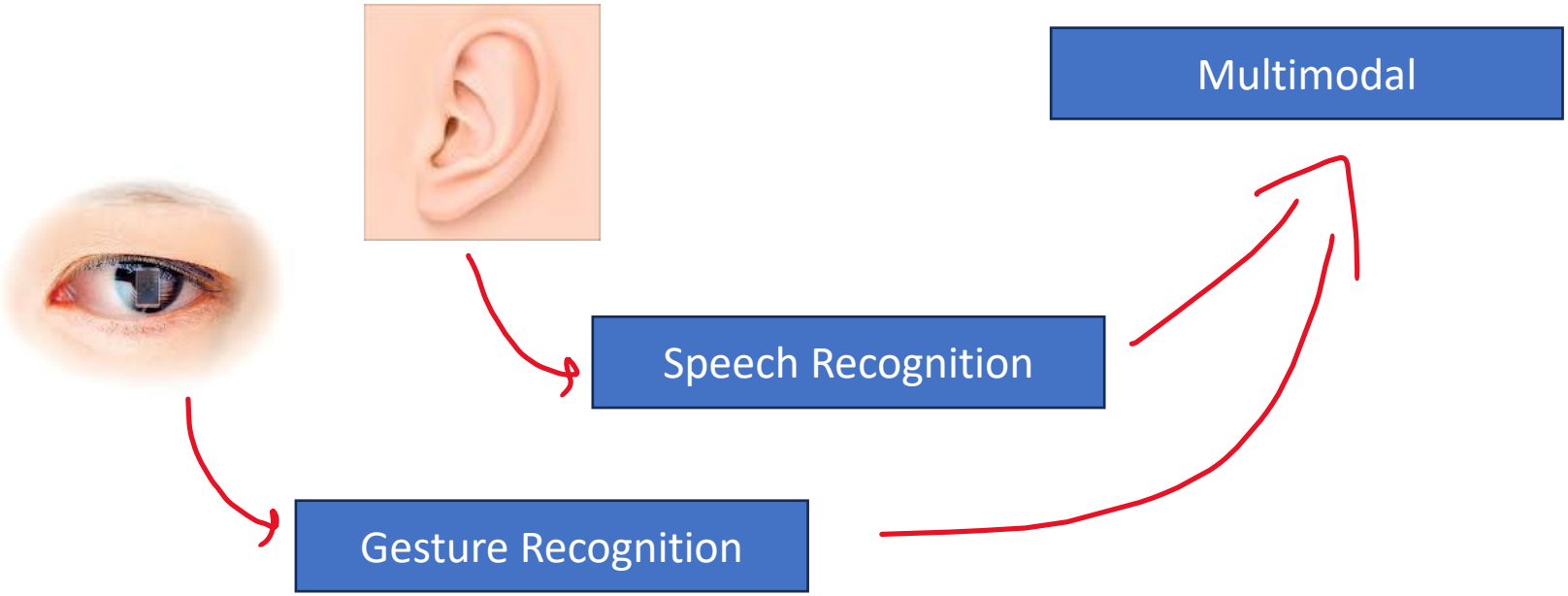
하지만, 로봇을 적용하더라도 소음과 저조도 환경에서의 인식이 어려웠음.

이러한 상황에서 어떻게 해결하고자 노력했는지를 다뤄볼 것임.

목차

2. 로봇 명령어 인공지능 인식 시스템

1. 구조 설명

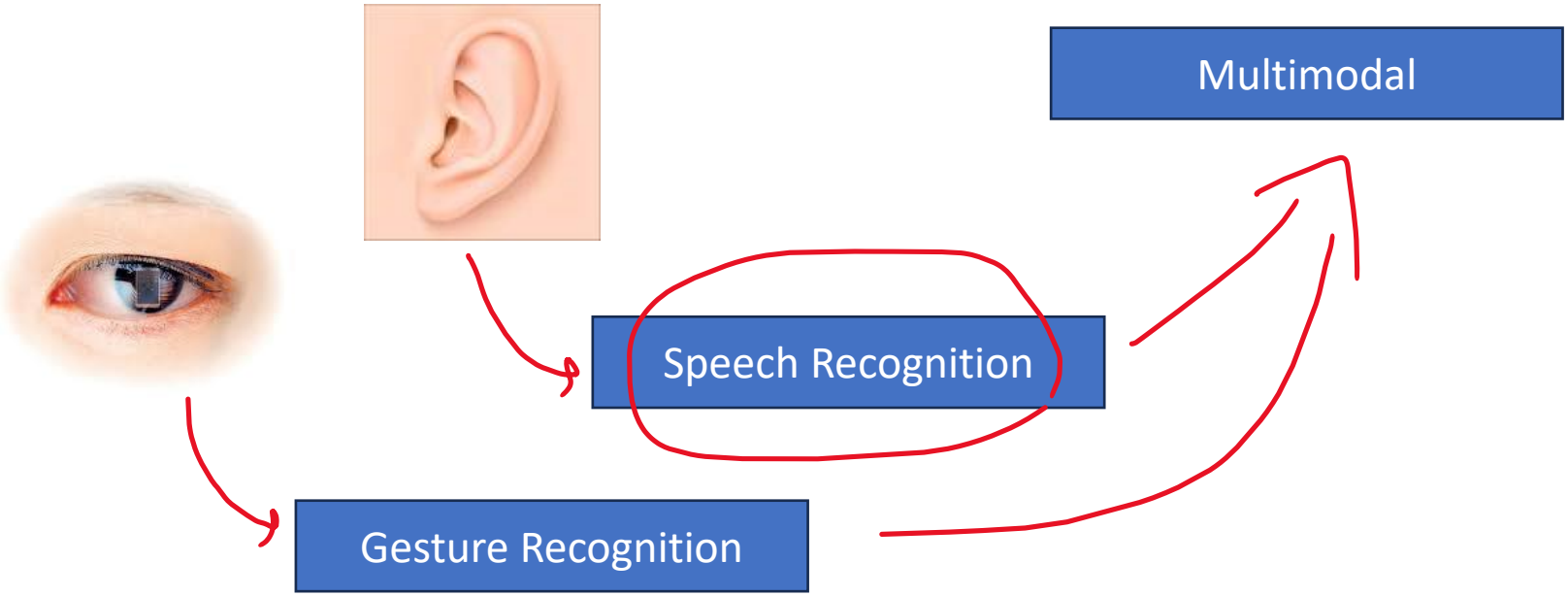


로봇에 '귀'인 음성인식과 '눈'인 제스처인식을 진행.

취합된 데이터를 바탕으로, 멀티모달에서 최종적인 결정을 내리게 됨.

2. 로봇 명령어 인공지능 인식 시스템

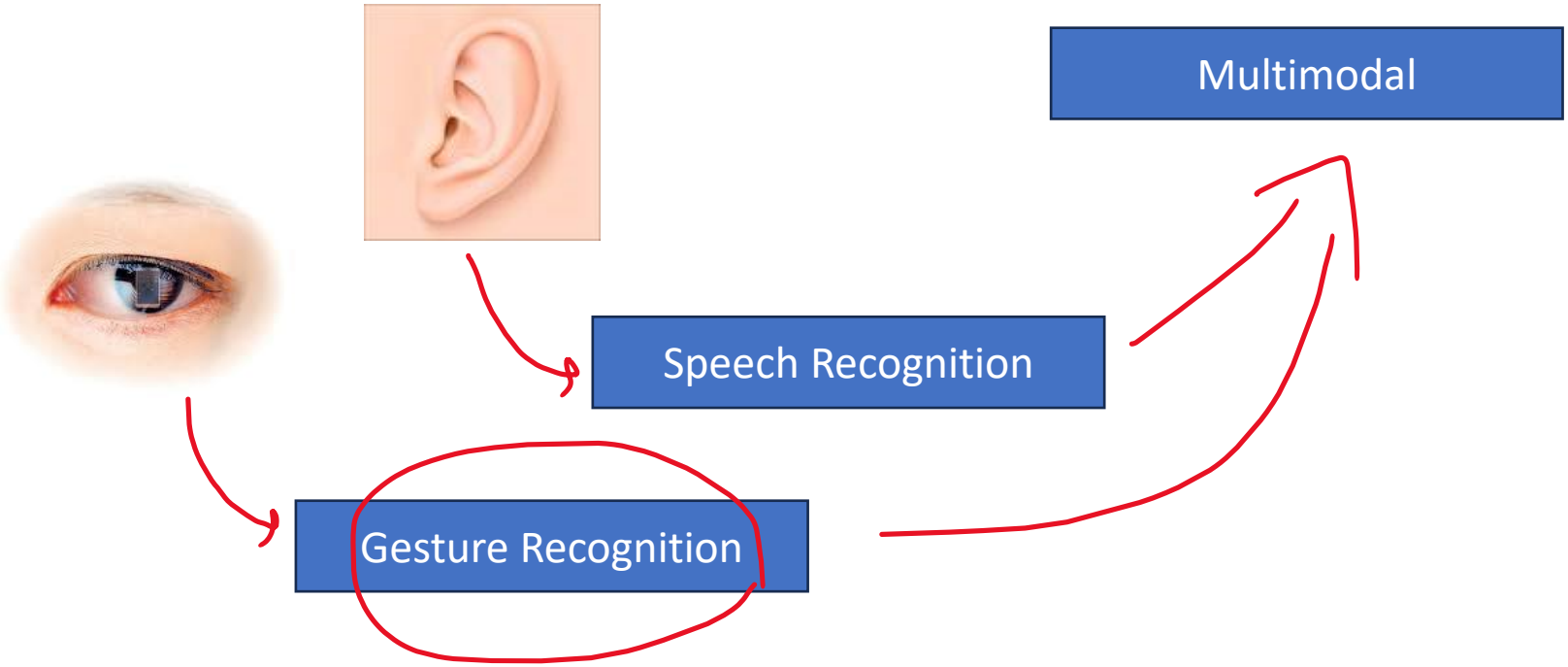
1. 구조 설명



첫 번째로, Speech Recognition, 즉 음성인식에 대한 내용을 먼저 볼 것임.

2. 로봇 명령어 인공지능 인식 시스템

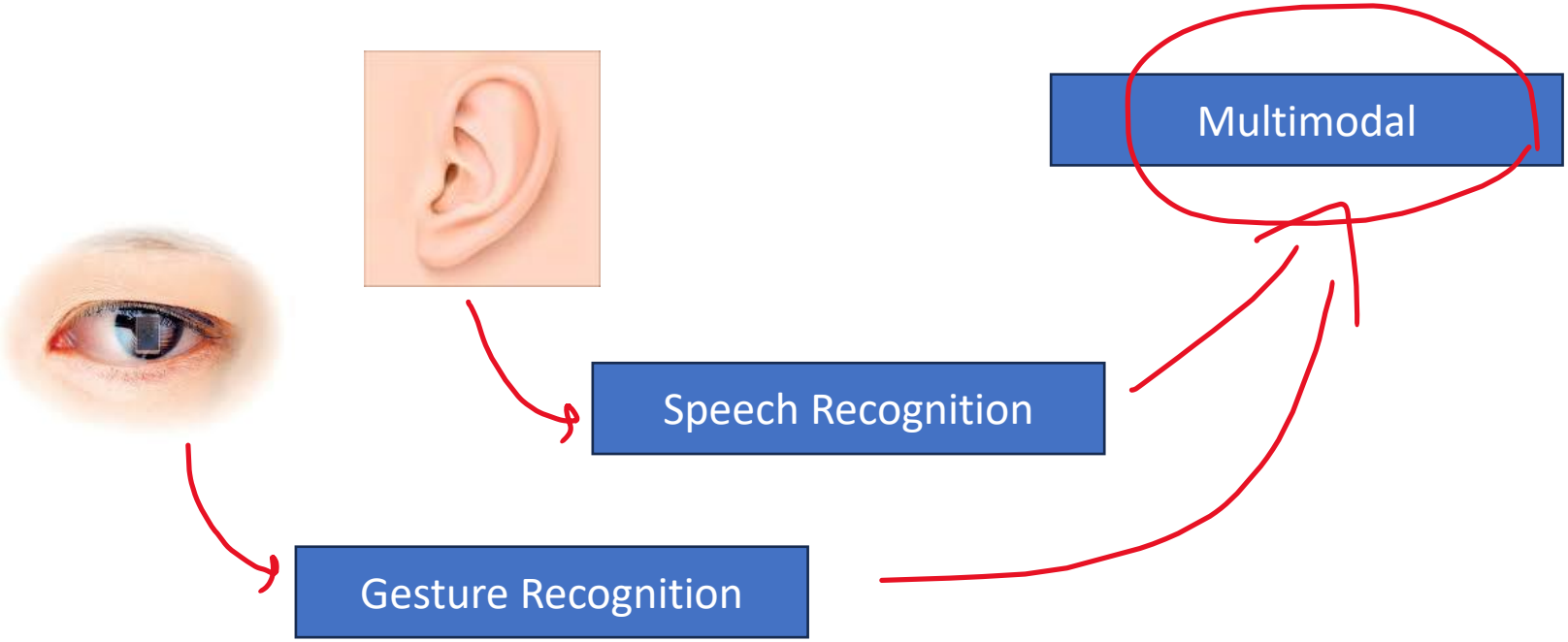
1. 구조 설명



그 다음으로 Gesture Recognition, 제스처인식에 대한 내용을 볼 것임.

2. 로봇 명령어 인공지능 인식 시스템

1. 구조 설명

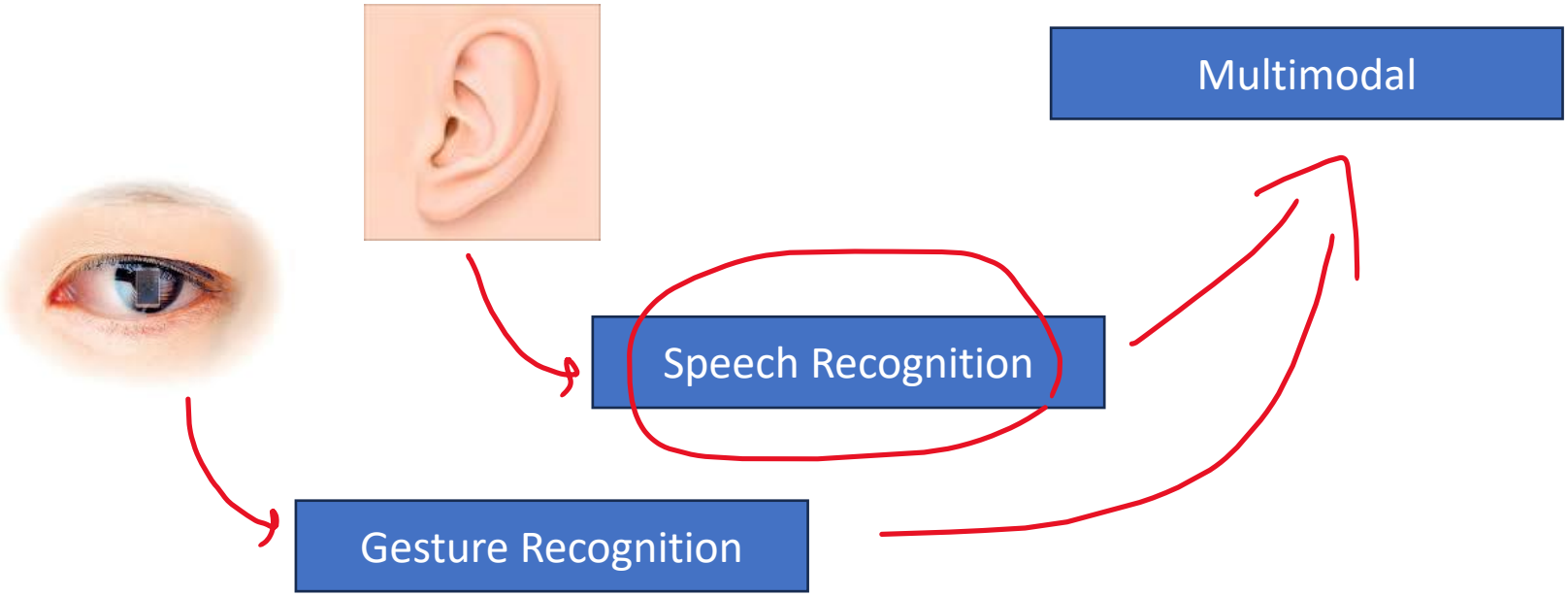


마지막으로, Multimodal, 멀티모달에 대한 내용을 볼 것임.

목차

3. 음성인식 시스템

1. 구조 설명



첫 번째로, Speech Recognition, 즉 음성인식에 대한 내용을 먼저 볼 것임.

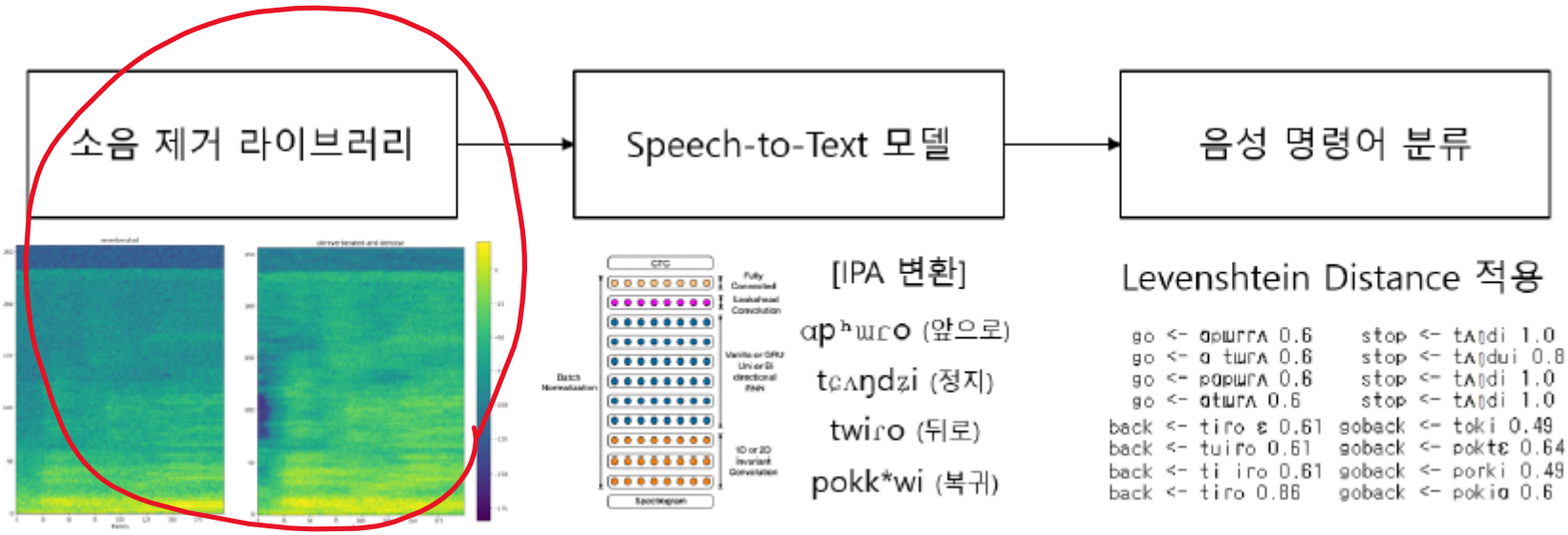
3. 음성인식 시스템

1. 구조 설명

전체적으로, 소음 제거 라이브러리 적용, STT 모델 적용, 마지막으로, 음성 명령어 분류의 순서로 진행됨.

3. 음성인식 시스템

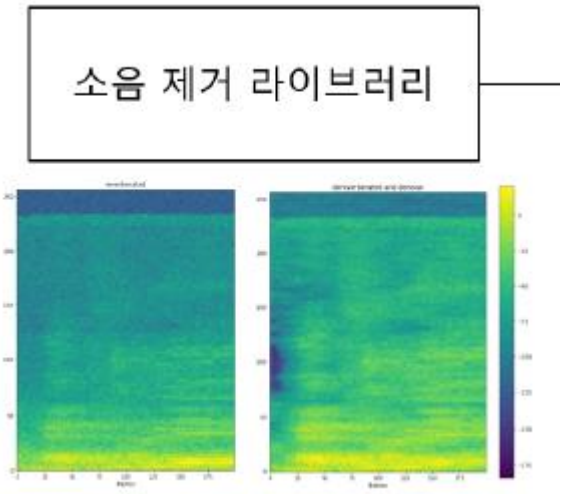
2. 소음 제거 라이브러리 적용



전체적으로, 소음 제거 라이브러리 적용, STT 모델 적용, 마지막으로, 음성 명령어 분류의 순서로 진행됨.

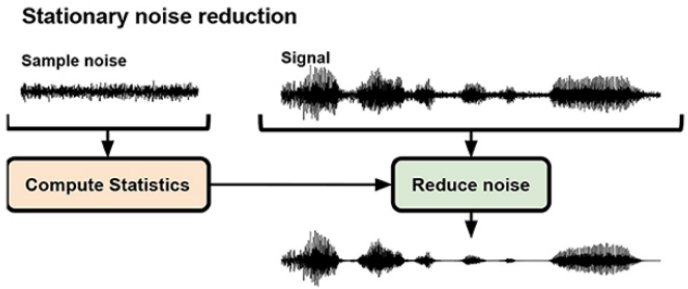
3. 음성인식 시스템

2. 소음 제거 라이브러리 적용



전체적으로 소음에 강건한 모델을 학습하기 위하여 소음 제거 라이브러리를 적용하고 소음이 동반된 학습 데이터를 사용하는 방식으로 접근하였다.

특히, 소음이 동반된 데이터에 대하여 denoiser 라이브러리를 적용하였다. (stationary-noise)

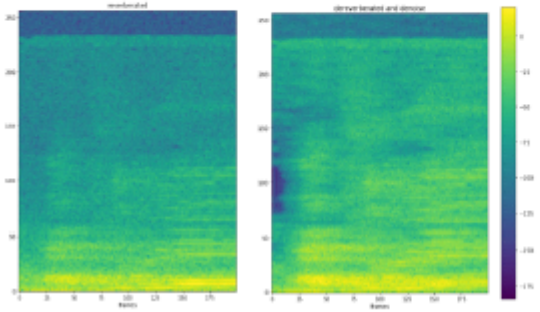


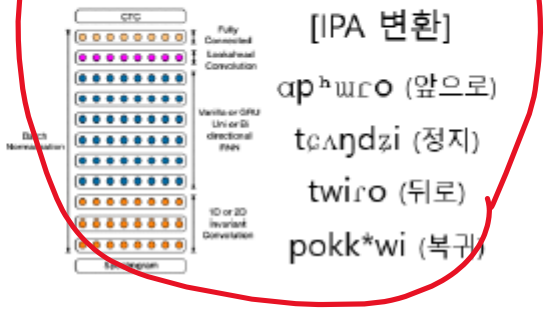
라이브러리 적용 결과, 정적으로 발생하는 소음을 감소할 수 있었다.

3. 음성인식 시스템

3. STT(Speech-to-Text) 모델 적용

소음 제거 라이브러리 → Speech-to-Text 모델 → 음성 명령어 분류





[IPA 변환]

ɑpʰɯɾo (앞으로)

tɕʌŋdʑi (정지)

twiɾo (뒤로)

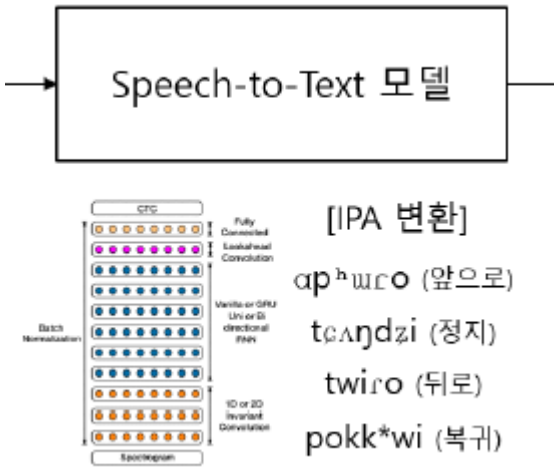
pokk*wi (복귀)

Levenshtein Distance 적용

```
go <- ɑpɯrrʌ 0.6        stop <- tʌdi 1.0
go <- ɑ twrʌ 0.6        stop <- tʌdui 0.8
go <- pɑpɯrʌ 0.6        stop <- tʌdi 1.0
go <- ɑtɯrʌ 0.6         stop <- tʌdi 1.0
back <- tiro ɛ 0.61  goback <- toki 0.49
back <- tuiro 0.61   goback <- poktɛ 0.64
back <- ti iro 0.61  goback <- porki 0.49
back <- tiro 0.86    goback <- pokiɑ 0.6
```

3. 음성인식 시스템

3. STT(Speech-to-Text) 모델 적용 – Deep Speech 2 모델 적용



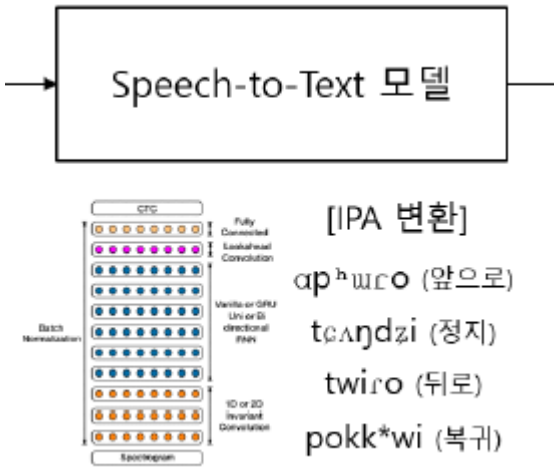
음성 인식을 위해 Deep Speech 2 모델을 적용하였음.

살짝 변형하여, CNN 2 + RNN 3 의 구조로 구성하였음.

데이터의 경우 AIHub 의 '극한 소음 음성인식 데이터'를 활용하여 학습을 진행하였다.

3. 음성인식 시스템

3. STT(Speech-to-Text) 모델 적용 – 국제표준발음기호(IPA) 적용



사람이 말하는 것을 문자로 변환하는 과정을 거친 후에 정해진 명령어를 바탕으로 분류를 진행

"앞으로" -> go / "뒤로" -> back

디루 -> 뒤로 (0점)

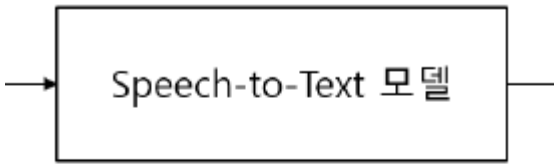
소음 동반된 상황에서 정확한 발음을 캐치하기 어렵기 때문에 국제표준발음 기호를 적용하여 유사도를 계산하기로 함.

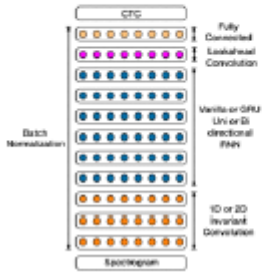
diru -> twiɾo (50점)

위 사례를 바탕으로 비교 난이도가 낮아진 것을 확인.

3. 음성인식 시스템

3. STT(Speech-to-Text) 모델 적용 – 발음 유사도 함수 적용





[IPA 변환]

ɑpʰɯɾo (앞으로)

tɕʌŋdʑi (정지)

twiɾo (뒤로)

pokk*wi (복귀)

발음 유사도 함수의 적용을 진행

<자음의 속성>

자음 조음위치, 자음 조음방법, 자음 조음강도,

자음 유성음여부

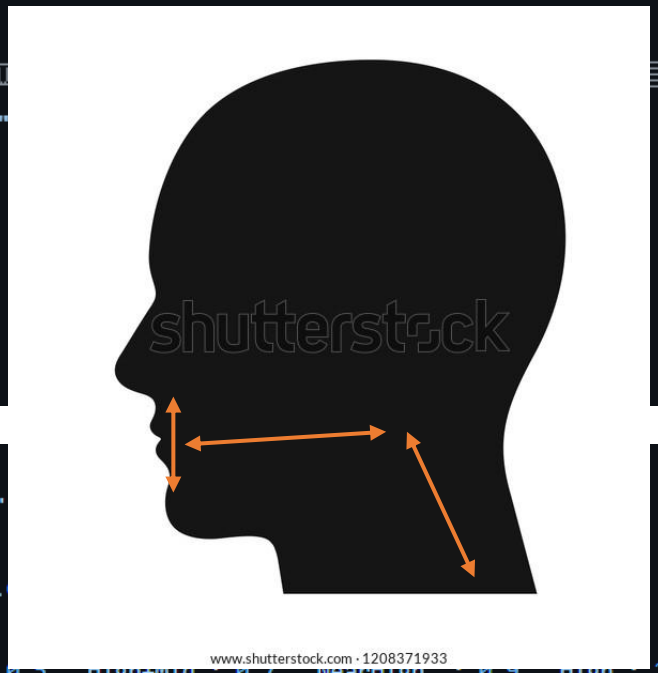
<모음의 속성>

모음 조음상하위치, 모음 조음좌우위치, 모음 입술모양

3. 음성인식 시스템

3. STT(Speech-to-Text) 모델 적용 – 발음 유사도 함수 적용

```
# Position (Bilabial) 0 --- 1 (Glottal)
conso_pos = { "Bilabial": 0, "Alveolar": 0.25, "Alveo-Palatal": 0.5, "Velar": 0.75, "Glottal": 1 }
# HowToPronunce (Plosive) 1 --- 0 (Lateral)
# 파열음(ㅂ, ㅍ, ㅃ, ㄷ, ㅌ, ㄸ, ㄱ, ㅋ, ㄲ), 마찰음(ㅅ, ㅆ, ㅎ), 파
conso_how = { "Plosive": 1, "Fricative": 0.5, "Affricate": 0.75, "
# Strength (Lenis) 0 --- 1 (Fortis)
conso_str = { "Lenis": 0, "Aspirated": 0.5, "Fortis": 1 }
# Voice or not (Yes) 0.5 --- 0 (No)
# 유성음인지 아닌지 구분
conso_voi = { "Yes": 0.5, "No": 0 }
```

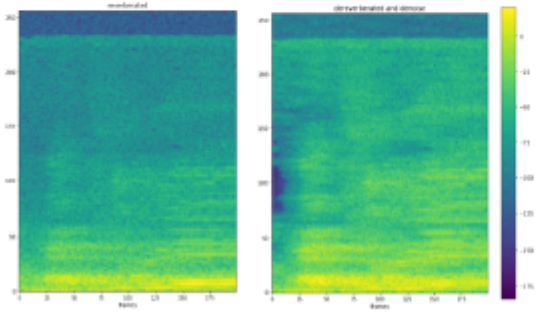


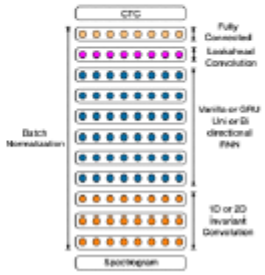

```
# shape (Unrounded) 0 --- 0.5 (Rounded)
vowel_shp = { "Unrounded": -0.5, "Rounded+Unrounded": -0.17, "Unrounded+Rounded"
# width position (Front) 0 --- 1 (Back)
vowel_wps = { "Front": 0, "NearFront": 0.1, "Back+Front": 0.33, "Front+Back": 0.
# height position (Low) 0 --- 1 (High)
vowel_hps = { "Low": 0, "NearLow": 0.1, "Mid+Low": 0.2, "High+Low": 0.4, "Mid": 0.5, "High+Mid": 0.7, "NearHigh": 0.9, "High": 1 }

vowels = pd.read_csv(os.path.join(path, "csv", "vowels.csv"))
```

3. 음성인식 시스템

4. 음성 명령어 분류

소음 제거 라이브러리 → Speech-to-Text 모델 → 음성 명령어 분류





[IPA 변환]

ɑpʰɯɾo (앞으로)

tɕʌŋdʑi (정지)

twiɾo (뒤로)

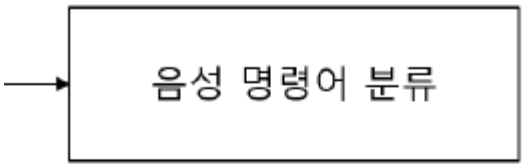
pokk*wi (복귀)

Levenshtein Distance 적용

```
go <- ɑpɯrrʌ 0.6      stop <- tʌ▯di 1.0
go <- ɑ tɯrʌ 0.6      stop <- tʌ▯dui 0.8
go <- pɑpɯrʌ 0.6      stop <- tʌ▯di 1.0
go <- ɑtɯrʌ 0.6       stop <- tʌ▯di 1.0
back <- tiro ɛ 0.61   goback <- toki 0.49
back <- tuiro 0.61    goback <- poktɛ 0.64
back <- ti iro 0.61   goback <- porki 0.49
back <- tiro 0.86     goback <- pokiɑ 0.6
```

3. 음성인식 시스템

4. 음성 명령어 분류

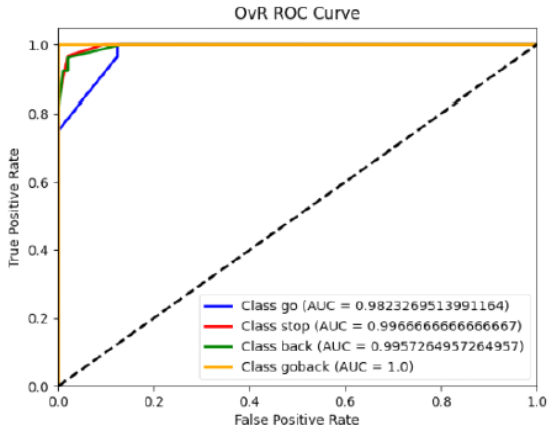


Levenshtein Distance 적용

go <- gowrra 0.6
go <- g twra 0.6
go <- gogwra 0.6
go <- gtwra 0.6
back <- tiro ɛ 0.61
back <- tuiro 0.61
back <- ti iro 0.61
back <- tiro 0.86
stop <- tʌŋdi 1.0
stop <- tʌŋdui 0.8
stop <- tʌŋdi 1.0
stop <- tʌŋdi 1.0
goback <- toki 0.49
goback <- poktɛ 0.64
goback <- porki 0.49
goback <- pokiɑ 0.6

마지막으로, 모든 명령어에 대하여 유사도 함수를 적용한 뒤에 가장 유사도가 높은 명령어를 고르는 방식으로 마무리 짓는다.

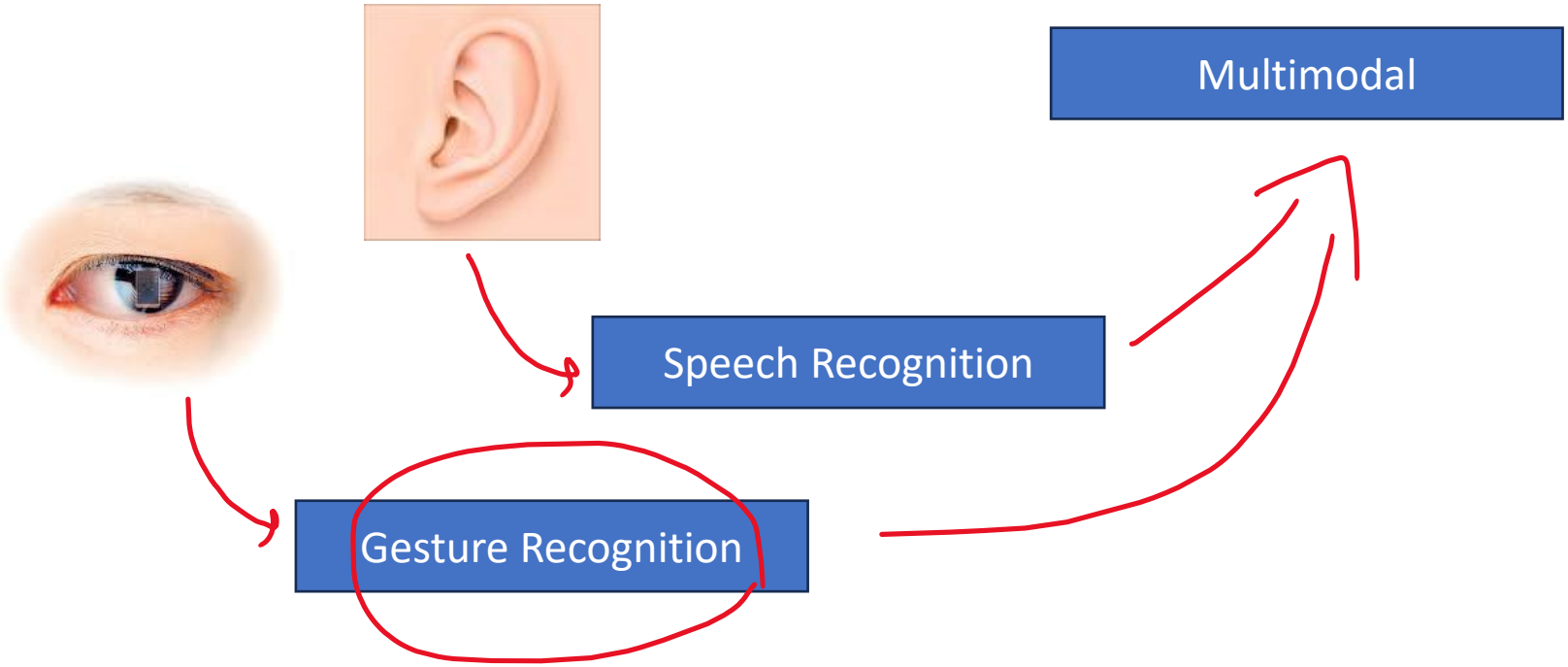
검증 결과는 Accuracy 및 F1-Score 는 93% 가 나오고 아래와 같은 ROC-Curve 그래프가 나왔다.



목차

4. 제스처 인식 시스템

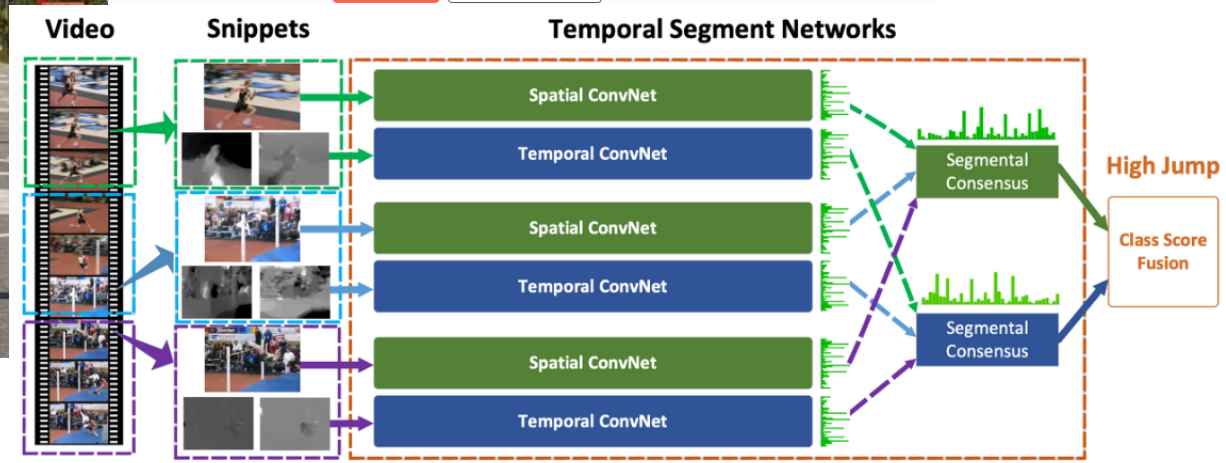
1. 구조 설명



그 다음으로 Gesture Recognition, 제스처인식에 대한 내용을 볼 것임.

4. 제스처 인식 시스템

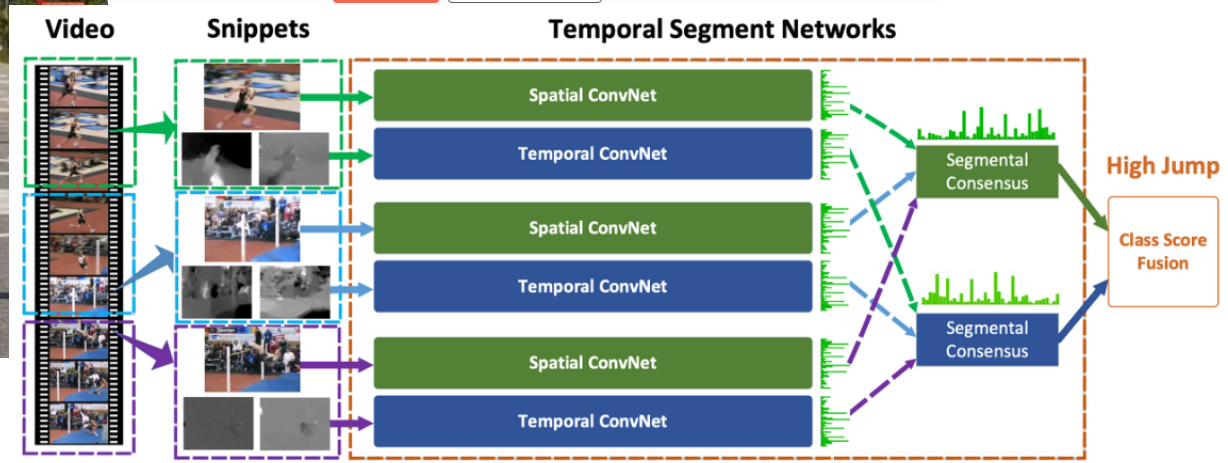
2. 제스처 인식 모델



자율주행차량의 실제 기상환경 및 시간대를 고려하여 다양한 촬영환경을 조성하고 촬영방법을 달리하여 구축한 교통 수신호 패턴 영상 데이터 학습

4. 제스처 인식 시스템

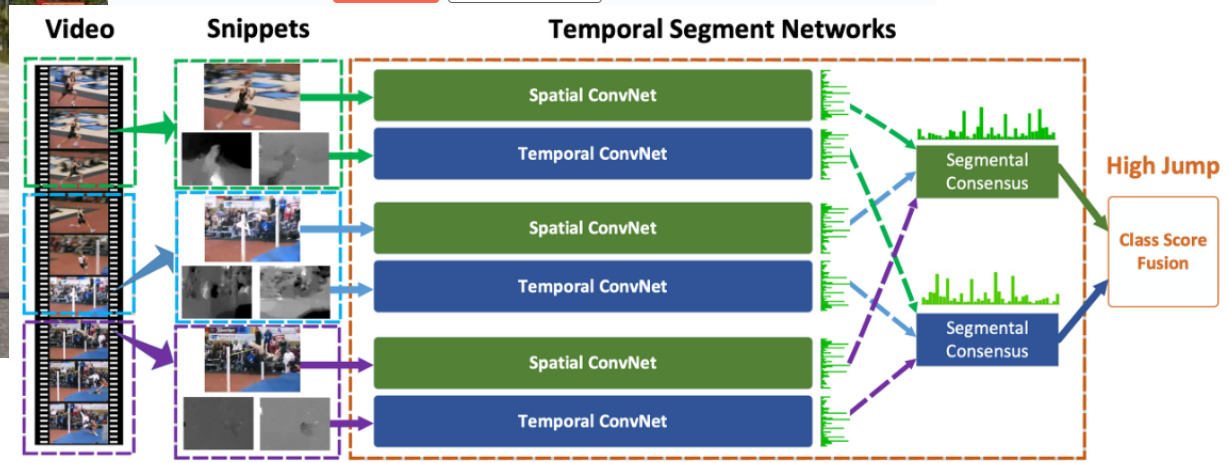
2. 제스처 인식 모델



총 6가지 교통 수신호 중에서 오른쪽 손들기는 go, 왼쪽 손들기는 back, 양손 들기는 goback, 전방 손들기는 stop 으로, 나머지는 none 으로 설정함.

4. 제스처 인식 시스템

2. 제스처 인식 모델



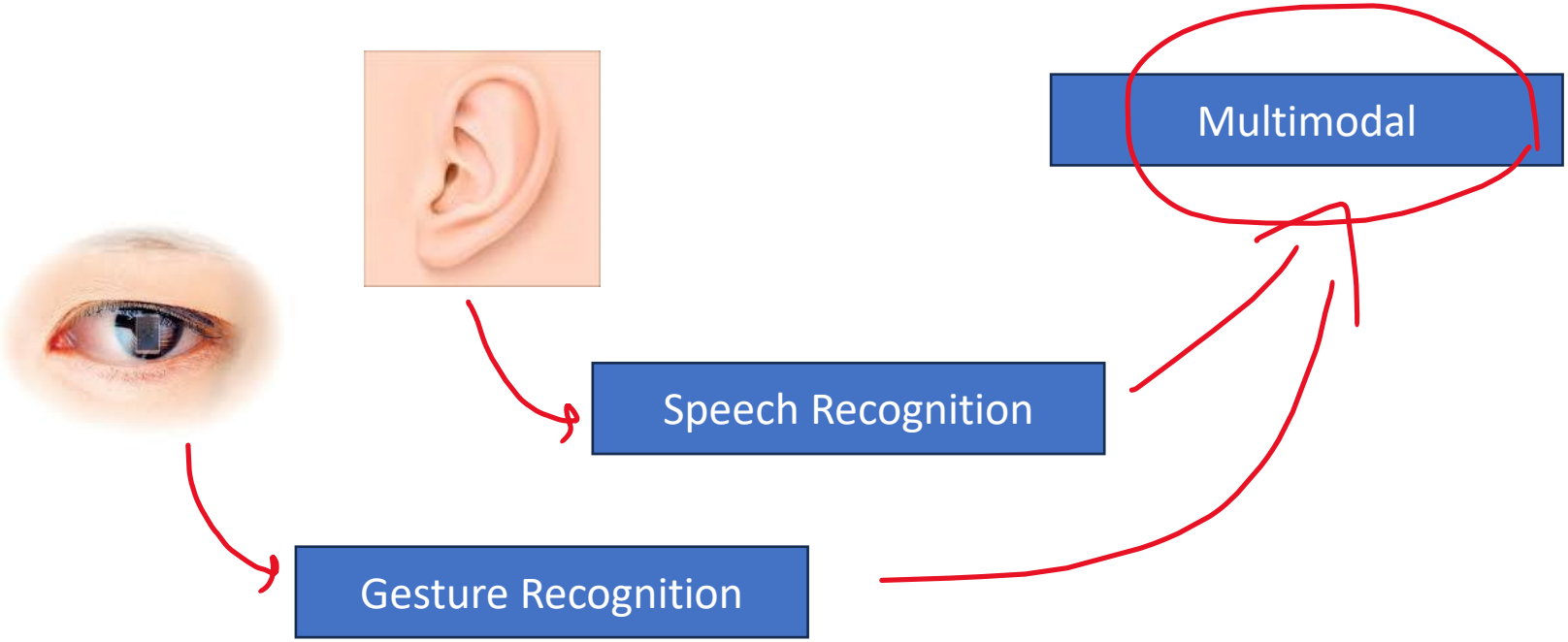
Pose Estimation Model 과 Gesture Recognition Model 을 적용하여

스켈레톤 포인트 기반 제스처 인식 모델 학습을 목표로 하였음.

목차

5. 멀티모달 시스템

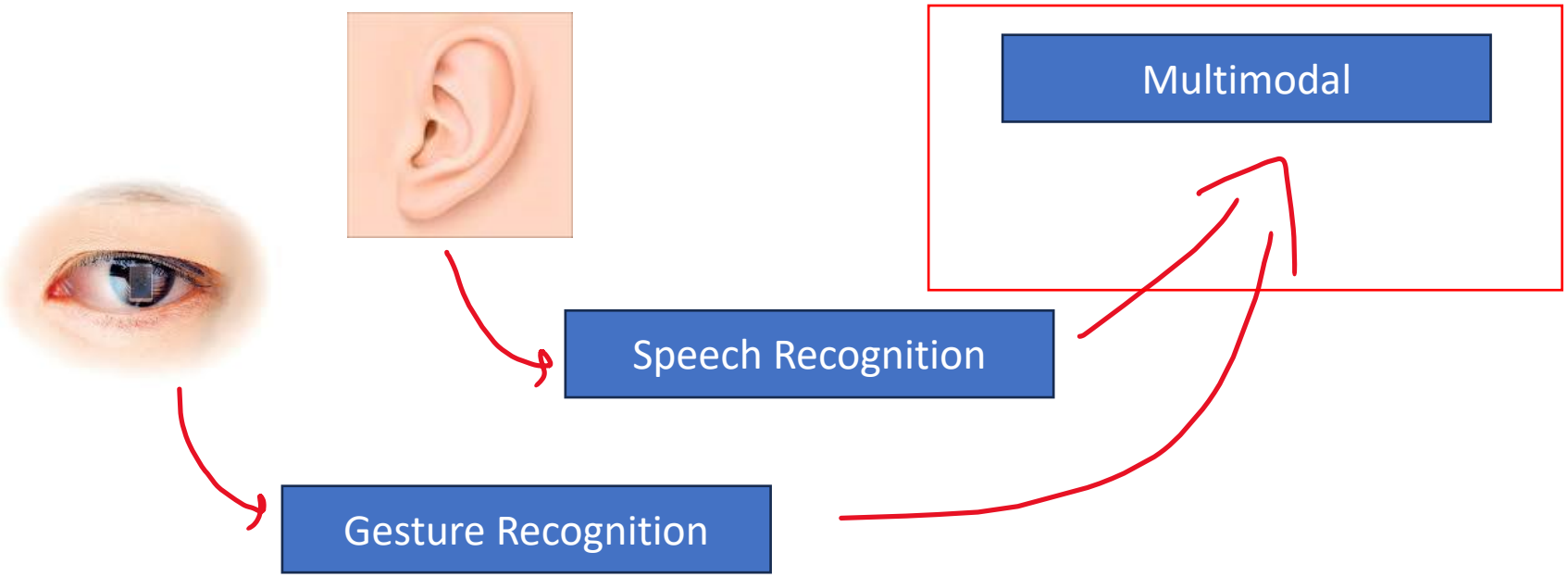
1. 구조 설명



마지막으로, Multimodal, 멀티모달에 대한 내용을 볼 것임.

5. 멀티모달 시스템

2. 알고리즘 설명



계산이 되어지는 방식은 다음과 같음.

1) (Speech + Gesture) / 2

2) 1) 의 결과값이 기준치를 상회하는지?

5. 멀티모달 시스템

2. 알고리즘 설명

Multimodal

(go, stop, back, goback) = (0.1, 0.45, 0.15, 0.2)

Speech Recognition

(go, stop, back, goback) = (0.1, 0.6, 0.2, 0.3)

Gesture Recognition

(go, stop, back, goback) = (0.1, 0.3, 0.1, 0.1)

계산이 되어지는 방식은 다음과 같음.

1) (Speech + Gesture) / 2

2) 1) 의 결과값이 기준치를 상회하는지?

5. 멀티모달 시스템

2. 알고리즘 설명

Multimodal

(go, stop, back, goback) = (0.1, 0.45, 0.15, 0.2)

0.5 넘는 것이 있나?

Speech Recognition

(go, stop, back, goback) = (0.1, 0.6, 0.2, 0.3)

Gesture Recognition

(go, stop, back, goback) = (0.1, 0.3, 0.1, 0.1)

계산이 되어지는 방식은 다음과 같음.

1) (Speech + Gesture) / 2

2) 1) 의 결과값이 기준치를 상회하는지? -> Threshold ???

5. 멀티모달 시스템

2. 알고리즘 설명 – Adaptive Threshold

$(speech) = (go, stop, back, goback)$

$(gesture) = (go, stop, back, goback)$

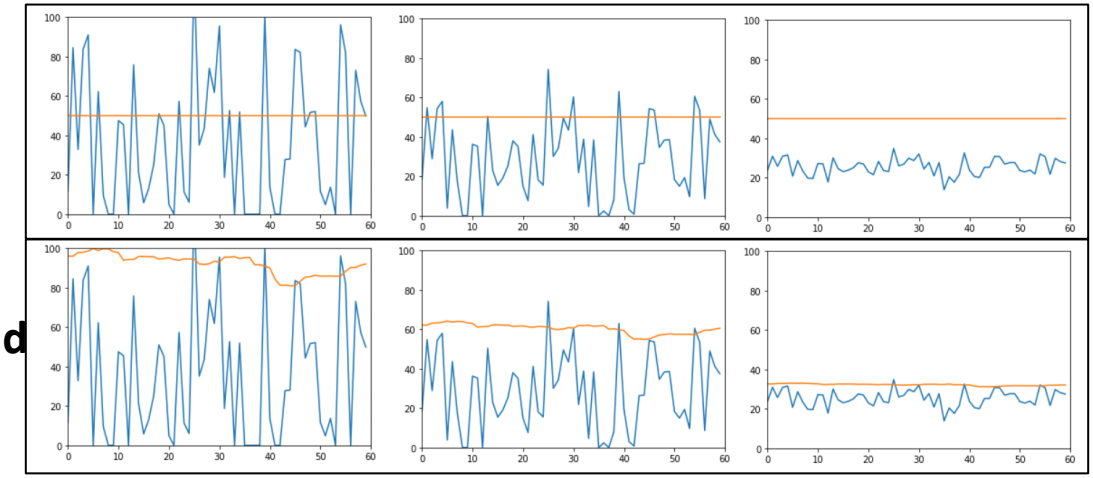
$(prob) = (speech * decay + gesture * decay) / 2$

$T = Mean(prob[-60:]) + Stdev(prob[-60:])$

$(Adaptive\ Threshold) = max(min(T, 0.8), 0.2)$

Fixed Threshold

Adaptive Threshold



-> 환경마다 다른 확률값이 분포

계산이 되어지는 방식은 다음과 같음.

1) (Speech + Gesture) / 2

2) 1) 의 결과값이 기준치를 상회하는지? **-> 환경에 따른 Threshold 제시**

5. 멀티모달 시스템

3. Real-Time Recognition 시스템 – 실전 적용을 위한 과정

평균 응답 시간도 중요함.

실험에 대한 자세한 내용은 다음 페이지에 기술됨.

연구실 101호 / RTX 3060 12GB + Intel i5 13400F 세팅으로 실험 진행

5. 멀티모달 시스템

3. Real-Time Recognition 시스템 – 요인 분석 (Factor Analysis)

응답 시간(Response Time)에 대한 요인 분석을 진행한 결과 다음과 같았음.

i) **컴퓨터의 사양** – HDD 보다는 SSD, CPU 및 GPU 성능 등

ii) **GPU 연산의 활용도** – 전처리 (gpu 연산 도입 후 매번 4초 절감) 등

iii) **모델의 복잡성** – 음성 및 비전 모델

iv) **Real-Time Recognition 시스템에서의 window 사이즈 및 Interval 설정**

5. 멀티모달 시스템

3. Real-Time Recognition 시스템 – 요인 분석 (Factor Analysis)

응답 시간(Response Time)에 대한 요인 분석을 진행한 결과 다음과 같았음.

i) **컴퓨터의 사양** – HDD 보다는 SSD, CPU 및 GPU 성능 등

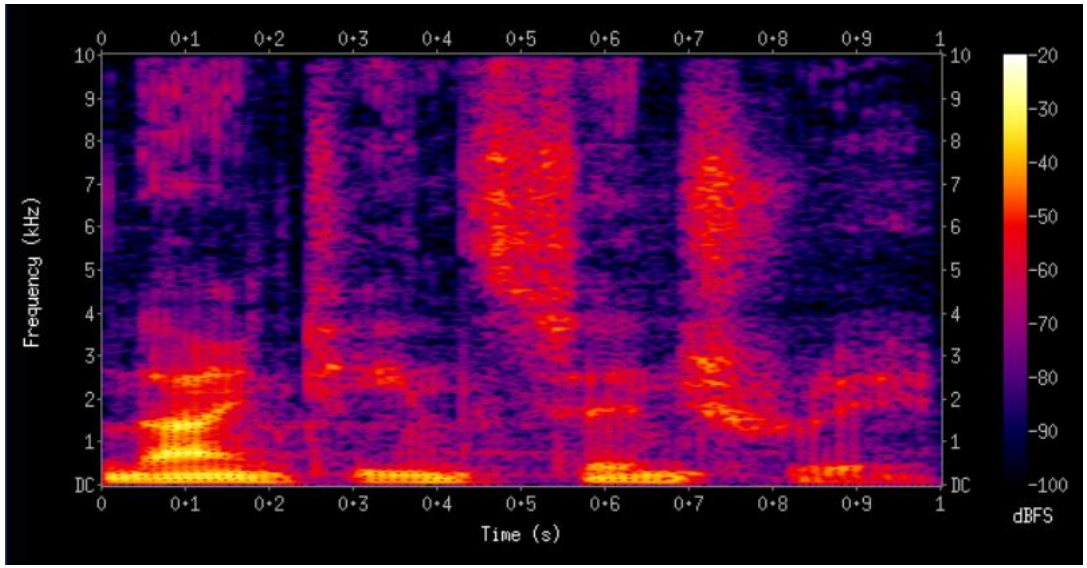
ii) **GPU 연산의 활용도** – 전처리 (gpu 연산 도입 후 매번 4초 절감) 등

iii) **모델의 복잡성** – 음성 및 비전 모델

iv) **Real-Time Recognition 시스템에서의 window 사이즈 및 Interval 설정**

5. 멀티모달 시스템

3. Real-Time Recognition 시스템

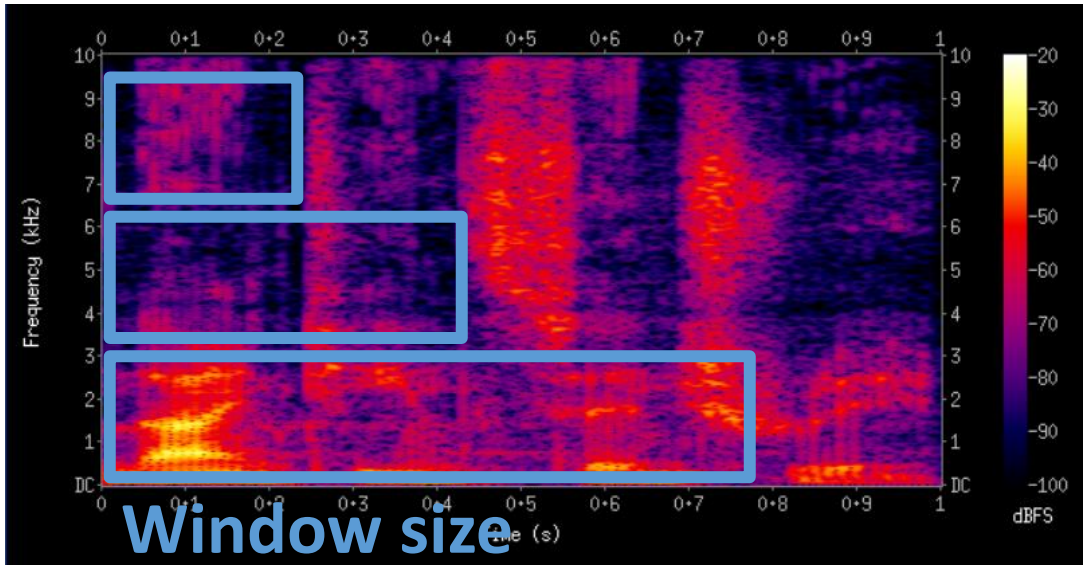


i) Window Size 설정 – 한 번에 얼마나 분석할 것이냐?

ii) Interval 설정 – 주기는 어떻게 설정할 것이냐?

5. 멀티모달 시스템

3. Real-Time Recognition 시스템



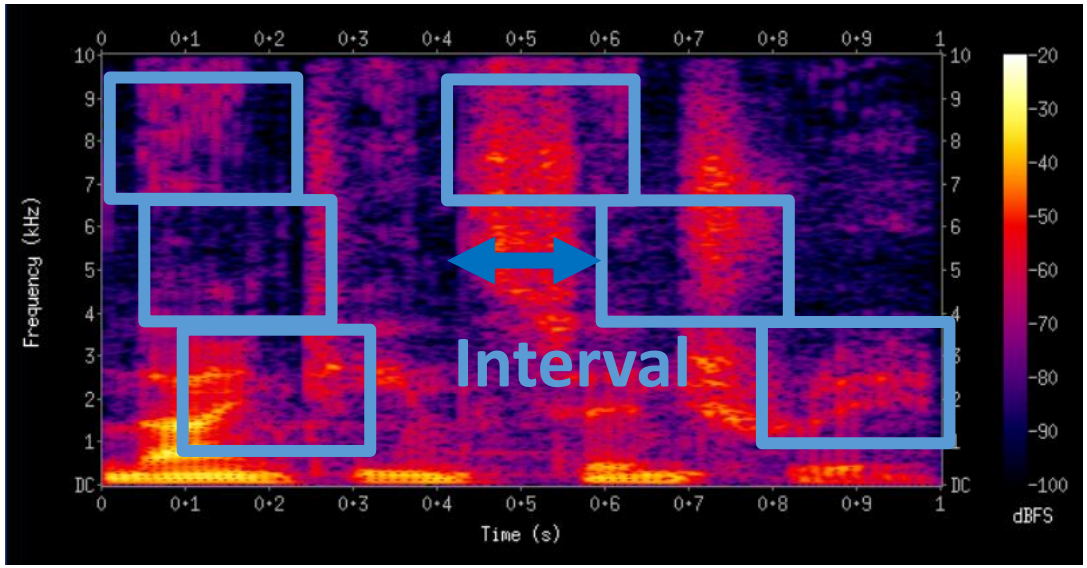
한 번에 너무 길면 추론시간 ↑ / 너무 짧으면 명령어 인식하기 힘듦

i) Window Size 설정 – 한 번에 얼마나 분석할 것이냐?

ii) Interval 설정 – 주기는 어떻게 설정할 것이냐?

5. 멀티모달 시스템

3. Real-Time Recognition 시스템



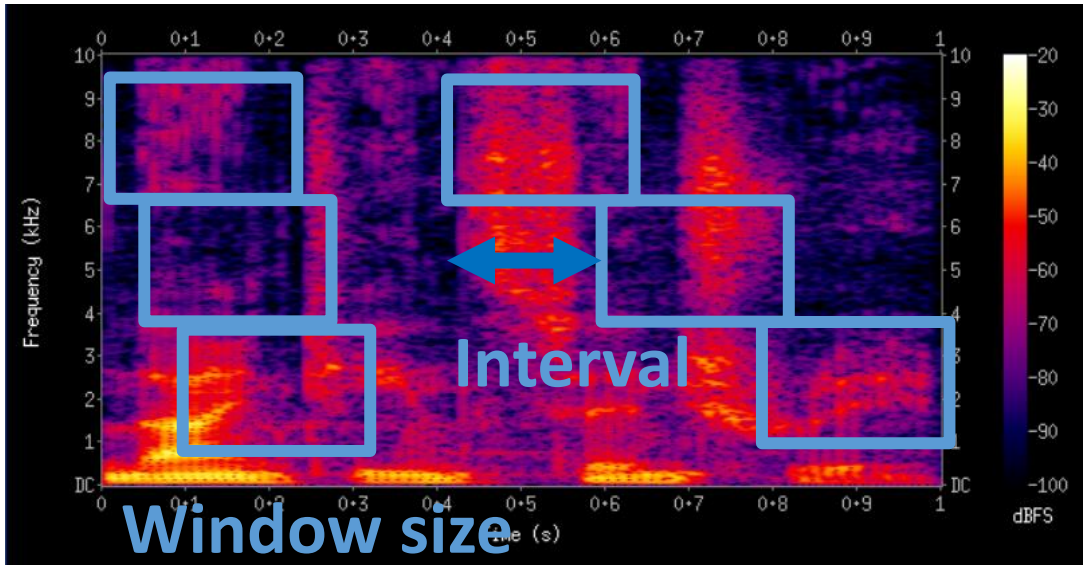
Interval 이 너무 짧으면 불필요한 연산 ↑ (딜레이 발생 가능) but 응답시간 빠름
너무 길면 응답시간 느려짐 (성능을 100% 끌어올리지 못함)

i) Window Size 설정 – 한 번에 얼마나 분석할 것이냐?

ii) Interval 설정 – 주기는 어떻게 설정할 것이냐?

5. 멀티모달 시스템

3. Real-Time Recognition 시스템



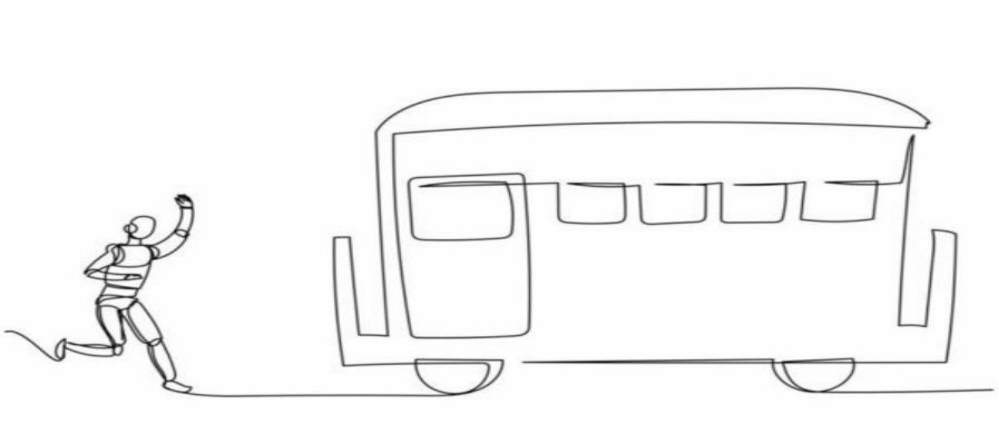
결론적으로는,

최소한의 명령 인식을 위한 시간 → Window Size
모델의 대략적인 평균 추론 시간보다 조금 더 짧게 → Interval
(window size 이 interval 보다 크면 놓치는 신호가 생길 수 있으니 유의 필요)

로 설정하는 방법을 추천함.

5. 멀티모달 시스템

3. Real-Time Recognition 시스템



결과적으로, **Interval 1초에 Window Size 2초를 바탕으로 설정**하였음.

100회 실험에 대한 평균 응답 시간의 경우, **음성 1.7초 / 비전 2초**

멀티모달 시스템까지 합쳐지면 약 2.5초 내외로 작동 가능함.

목차

6. 결론

1. 인공지능 인터페이스 적용 방식

문제점 해결 시나리오

소음 환경 → 소음 제거, 발음 기호 및 발음 유사도 함수 적용

저조도 환경 → 다양한 환경에서의 데이터셋 적용

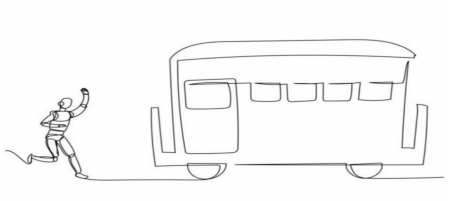
다양한 작업 환경 → Adaptive Threshold 적용

실제 적용을 위한 준비

Real-Time System 구현 → 적절한 Window Size 및 Interval 설정

6. 결론

2. 인공지능 인터페이스 적용 효과



Multimodal

Speech Recognition

Gesture Recognition

앞서 설명한 방법을 바탕으로 주변 소음 및 저조도 환경에서의 인공지능 인터페이스 적용에 대한 가능성을 보여주었음.

Thank you!

MILab Undergraduate student, Kim Taehyeon

2023. 11. 14

Reference

- https://aihub.or.kr/aihubdata/data/view.do?dataSetSn=71417
- https://aihub.or.kr/aihubdata/data/view.do?dataSetSn=496
- https://github.com/stannam/hangul_to_ipa
- https://github.com/open-mmlab/mmaction2
- https://github.com/sooftware/kospeech
- https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8578026
- https://pypi.org/project/denoiser/
- https://auto.gluon.ai/
- http://arxiv.org/pdf/1512.02595v1.pdf
- https://github.com/DevTae/AdaptiveThresholdTest
- http://miraecity.com/home.html
- https://dongsarchive.tistory.com/63
- https://github.com/timsainb/noisereduce