ROBOTYKA AFEKTYWNA

Wykład 3: Percepcja audio i fuzja multimodalna

Automatyka i Robotyka, studia II stopnia

dr inż. Mateusz Pomianek • Politechnika Rzeszowska

Po tym wykładzie będziesz umiał/a:

- Wyliczyć podstawowe cechy akustyczne mowy: pitch, energię, MFCC
- Wyjaśnić różnicę między klasyfikacją na cechach a podejściem end-to-end, na przykład wav2vec2
- Zaprojektować architekturę fuzji: early, late albo hybrid
- Zsynchronizować strumienie audio i wideo w ROS2 przy pomocy message_filters
- Zaprojektować bezpieczne zachowanie systemu przy braku jednego z kanałów



SZYBKIE PRZYWOŁANIE Z WYKŁADU 2

Jaki wektor zwraca klasyfikator emocji z W2? Co jeśli twarz jest zasłonięta?



Dziś rozszerzamy etap Sensoryka o kanał audio i uczymy się łączyć oba kanały.

Co słyszy robot, czego nie widzi kamera

Operator stoi tyłem do kamery, twarz niewidoczna. W słuchawce robota słychać jednak podniesiony, drżący głos i przyspieszone tempo mowy.

Kanał wizyjny z W2 w tej chwili nie działa w ogóle.

Głos niesie sygnał niezależny od pozycji ciała, to naturalna redundancja, którą można wykorzystać.



Pipeline percepcji audio



- Akwizycja: tablica mikrofonowa czterokanałowa z W1, beamforming
- Preprocessing: pre-emphasis, czyli wzmocnienie wysokich częstotliwości
- Ramkowanie: sygnał dzielony na ramki o długości od 20 do 40 ms, z zachodzeniem
- Ekstrakcja cech: to jest etap 2 z pipeline'u W1, tym razem kanał audio

Cechy akustyczne mowy

- **Pitch, F0, czyli częstotliwość podstawowa**: ton głosu, rośnie przy pobudzeniu (arousal), spada przy smutku
- **Energia (RMS)**: głośność wypowiedzi, nagłe skoki mogą sygnalizować złość lub strach
- **Tempo mowy**: sylaby i słowa na sekundę, przyspiesza przy stresie, zwalnia przy zmęczeniu
- **Pauzy w mowie i ich rozkład**: częstsze lub dłuższe pauzy mogą wskazywać na niepewność, wahanie, obciążenie poznawcze

MFCC, czyli Mel-Frequency Cepstral Coefficients



Wzór DCT, dyskretnej transformaty kosinusowej, na K pasm melowych:

$$c_n = \sum_{k=1}^{K} \log(S_k) \cdot \cos\left[n\,(k-0{,}5)\,\frac{\pi}{K}\right]$$

Typowo używa się pierwszych 13 współczynników c_n jako wektora cech na ramkę. Skala melowa odwzorowuje nieliniową percepcję wysokości dźwięku przez ucho ludzkie.

Jitter i shimmer, czyli mikrozaburzenia głosu

Jitter

Zmienność okresu podstawowego, F0, między kolejnymi cyklami drgań strun głosowych.

Wzrost jitteru wskazuje na napięcie mięśni krtani, stres, zmęczenie głosowe.

Shimmer

Zmienność amplitudy sygnału między kolejnymi cyklami.

Wysoki shimmer koreluje z chrypką, napięciem emocjonalnym i niestabilnością fonacji.

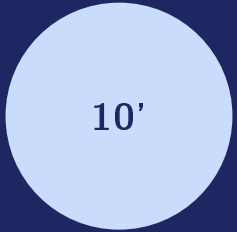
Klasyfikacja: cechy ręczne a podejście end-to-end

Cecha	SVM na cechach (MFCC, pitch)	wav2vec2 (end-to-end)
Dane treningowe	Działa przy małych zbiorach	Wymaga dużych zbiorów lub fine-tuningu
Koszt obliczeniowy	Niski, działa na CPU	Wysoki, zwykle potrzebny GPU lub edge-AI
Interpretowalność	Wysoka, cechy mają znaczenie fizyczne	Niska, reprezentacje uczone
Typowa dokładność	Dobra baza odniesienia	Wyższa przy wystarczających danych

SZYBKIE PRZYWOŁANIE

Wymień trzy cechy akustyczne i powiedz, co sygnalizuje wzrost jittera.

30 sekund indywidualnie, potem porównaj odpowiedź z sąsiadem



PRZERWA

10 minut, wracamy o godzinie [uzupełnić]

Szybkie przypomnienie

- Kanał audio: MFCC, pitch, energia, jitter i shimmer, razem tworzą wektor cech
- Kanał wizyjny z W2: landmarki i Action Units tworzą osobny wektor cech

Teraz zajmiemy się tym, jak połączyć oba kanały w jedną, spójną decyzję o stanie afektywnym.

Dlaczego fuzja multimodalna

- Redundancja: gdy jeden kanał zawodzi, hałas albo okluzja, drugi wciąż działa
- Komplementarność: audio i wideo niosą częściowo niezależną informację
- Wyższa odporność, ang. robustness, niż pojedynczy klasyfikator w warunkach przemysłowych
- Koszt: dodatkowa złożoność, trzeba zsynchronizować i połączyć strumienie

Architektury fuzji: early, late i hybrid

Early fusion

Surowe cechy audio i wideo łączone, konkatenacja, przed jednym wspólnym klasyfikatorem.

Late fusion

Dwa niezależne klasyfikatory, audio i wideo, których decyzje i wyniki łączone są na końcu.

Hybrid fusion

Cechy pośrednie łączone w warstwie ukrytej wspólnej sieci, kompromis między early i late fusion.

Matematyka: ważona fuzja late

$$P_{\text{fused}}(e) = w_v \cdot P_{\text{video}}(e) + w_a \cdot P_{\text{audio}}(e), \qquad w_v + w_a = 1$$

Wagi oparte na pewności klasyfikatora, ang. confidence-weighted fusion:

$$w_v = \frac{\text{conf}_v}{\text{conf}_v + \text{conf}_a} \qquad w_a = \frac{\text{conf}_a}{\text{conf}_v + \text{conf}_a}$$

Gdy jeden kanał ma niską pewność, na przykład audio przy ciszy, jego waga w fuzji maleje automatycznie. System elegancko przechodzi wówczas do drugiego kanału, zamiast się zawieszać.

Problem synchronizacji strumieni

- Kamera: klatki co około 33 ms, czyli 30 fps, patrz W1
- Audio: cechy MFCC liczone na ramkach od 20 do 40 ms, agregowane do okna decyzyjnego około 1 sekundy
- Różne częstotliwości i różne opóźnienia dotarcia komunikatów do węzła fuzji
- Bez synchronizacji fuzja łączy dane z różnych momentów w czasie, co daje błędny wynik

ROS2: message_filters i ApproximateTimeSynchronizer

- message_filters.Subscriber opakowuje temat, czyli topic, do użycia w synchronizatorze
- TimeSynchronizer wymaga identycznych znaczników czasu, co rzadko bywa realistyczne
- ApproximateTimeSynchronizer dopuszcza tolerancję czasową dzięki parametrowi slop
- queue_size ogranicza bufor, co zapobiega narastaniu opóźnień przy różnicach częstotliwości

Kod: synchronizacja topików audio i wideo

```
import message_filters
from affect_msgs.msg import AffectState

video_sub = message_filters.Subscriber(
    self, AffectState, '/affect/video_state')
audio_sub = message_filters.Subscriber(
    self, AffectState, '/affect/audio_state')

sync = message_filters.ApproximateTimeSynchronizer(
    [video_sub, audio_sub], queue_size=10, slop=0.15)
sync.registerCallback(self.on_synced_pair)
```

Kod: fuzja late w węźle ROS2

```
def on_synced_pair(self, video_msg, audio_msg):
    cv, ca = video_msg.confidence, audio_msg.confidence
    wv = cv / (cv + ca) if (cv + ca) > 0 else 0.5
    wa = 1 - wv
    fused = AffectState(
        valence=wv*video_msg.valence + wa*audio_msg.valence,
        arousal=wv*video_msg.arousal + wa*audio_msg.arousal,
        confidence=max(cv, ca))
    self.fused_pub.publish(fused) # -> /affect/state
```

To jest implementacja wzoru ze slajdu 15, wagi wynikają wprost z pewności klasyfikatorów.

Braki danych i redukcja opóźnień

- Brak audio, na przykład cisza albo awaria mikrofonu: confidence audio dąży do zera, więc waga automatycznie spada do zera
- Timeout: jeśli drugi komunikat nie nadejdzie w oknie slop, publikuj dane z jednego kanału
- Bufor ograniczony przez queue_size odrzuca stare, nieaktualne dane, więc nie akumulują one opóźnień
- Budżet z W1: fuzja musi zmieścić się w pozostałym marginesie czasowym, około 150 ms, patrz W1 slajd 17

Kluczowe pojęcia dzisiejszego wykładu

MFCC, pitch, jitter i shimmer	**Early, late i hybrid fusion**	Confidence-weighted fusion
SVM a wav2vec2	**ApproximateTime Synchronizer**	Degradacja łagodna przy braku danych

Quiz kontrolny

1. Wideo ma confidence 0,2, audio 0,8. Jakie wagi otrzyma każdy kanał w fuzji?
2. Dlaczego stosujemy ApproximateTimeSynchronizer, a nie TimeSynchronizer, dla audio i wideo?
3. Kanał audio milknie na 5 sekund. Co powinien zrobić system fuzji?

Wykład 4: sygnały fizjologiczne i integracja sprzętowa

- Czujniki: EDA/GSR, HRV (PPG), EEG konsumenckie, na przykład Muse, OpenBCI
- Protokoły komunikacji: BLE oraz Lab Streaming Layer (LSL)
- Integracja urządzenia wearable z ROS2: własny driver i węzeł

Praca własna przed W4:

Zmierz swój puls, na przykład aplikacją w telefonie, przed i po krótkim wysiłku, i zaobserwuj zmianę HRV.

Dziękuję za uwagę

Pytania?