Speech Recognition and Voice Separation for the Internet of Things

Mohammad Hasanzadeh Mofrad and Daniel Mosse

Department of Computer Science
School of Computing and Information
University of Pittsburgh

The 8th International Conference on the Internet of Things (IoT 2018)
October 15-18, 2018 in Santa Barbara, California, USA

Discussion Outline

- **Motivations and contributions**
- **Background**
- Proposed voice-enabled IoT prototype
- Reconstruction lowpass filter for a voice-enabled IoT prototype
- Results
- Summary and conclusion

Motivation


- Ways of communicating with IoT devices
  - *Graphical User Interface (GUI)*
  - *Speech Interfaces*
- **Limitations** of the current smart home IoT devices (e.g. a smart speaker)
  1. Devices are not customizable: static functionality (voice commands and accuracy)
  2. Smart home speakers cannot handle complex scenarios such as:
     1. They fail processing combined commands separated by "**and**".
     2. They fail processing two **concurrent** commands


Contributions

- **Contributions** of this paper are two folds:
  1. Prototype: A customizable voice-enabled IoT system

  2. Model and Implementation: A model for handling two concurrent voice commands to a voice-enabled IoT device.
     - For example, the case a person says, “Dim the lights.” and at the same time the other person says, “Turn on the TV.”

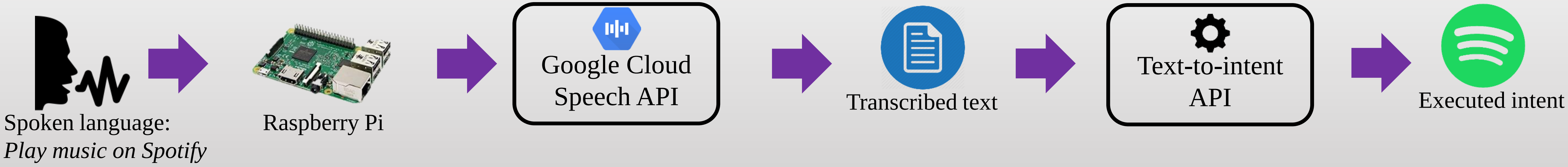
Background

- Smart home speakers
  - Voice-enabled device widely use *speech processing* and *natural language processing* to create a
    - Recording is done by the device
    - Processing is done in the Cloud
- Blind Source Separation (BSS)
  - The Cocktail party effect
  - The problem of processing multiple concurrent voice commands by a voice-enabled IoT device
- BSS solution: Independent component Analysis
- Low-pass filters in signal processing (we use the Butterworth filter)

Discussion Outline

- Motivations and contributions
- Background
- **Proposed voice-enabled IoT prototype**
- Reconstruction lowpass filter for a voice-enabled IoT prototype
- Results
- Summary and conclusion

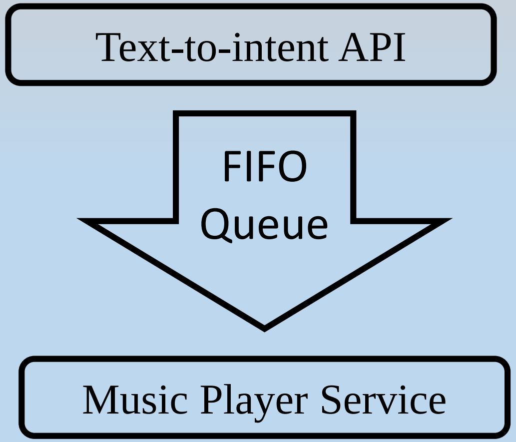
Proposed voice-enabled IoT Prototype



- The proposed model consists of the following components:
  1. The Raspberry Pi records voice and sends it to the Google Cloud speech-to-text API
  2. The Google Cloud speech-to-text API transcribes the voice into text
  3. The text-to-intent API receives the text and converts it to an intent and target device.

Proposed voice-enabled IoT prototype – Text-to-intent API

- Text-to-intent API receives the transcribed text from the Google Cloud speech-to-text API and extracts the followings using a simple language model:
    1. The *intent* of the voice message
    2. The target *device* that the command is intended to be executed on.
- The intents that are currently supported by our proposed prototype are
    - *Play music*
    - *Pause music*
    - *Resume music*
    - *Stop* music
- Device
    - *An open-source command-line music player*



Proposed voice-enabled IoT Prototype – Hardware

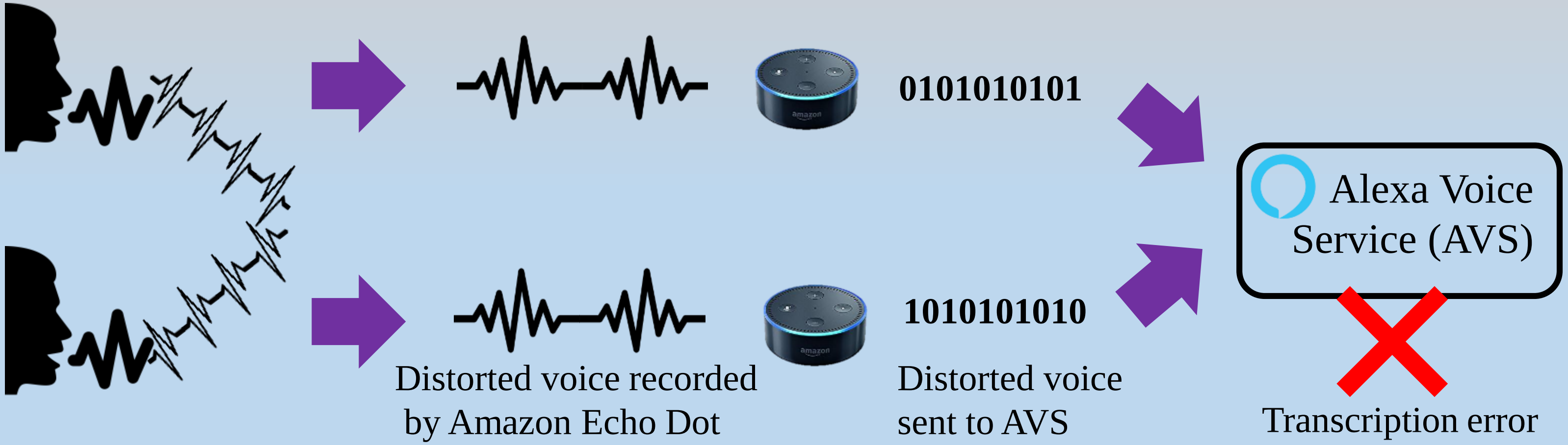
- Inexpensive prototype! **$68.42**
- The main hardware components are:
  - Raspberry Pi 3 Model B Motherboard, $35.80
    - Quad core Cortex A53 @ 1.2GHz
    - 1GB SDRAM
    - Wireless 802.11
    - Bluetooth 4.0
  - Kinobo USB 2.0 Mini Microphone, $4.65
  - Samsung 64GB Micro SD Card, $19.99
  - Raspberry Pi Case, $7.98
  - Other hardware: keyboard, cables, etc.
- Sofware: Raspbien, Python, Cloud API, …

Discussion Outline

- Motivations and contributions
- Background
- Proposed voice-enabled IoT prototype
- **Reconstruction lowpass filter for a voice-enabled IoT prototype**
- Results
- Summary and conclusion

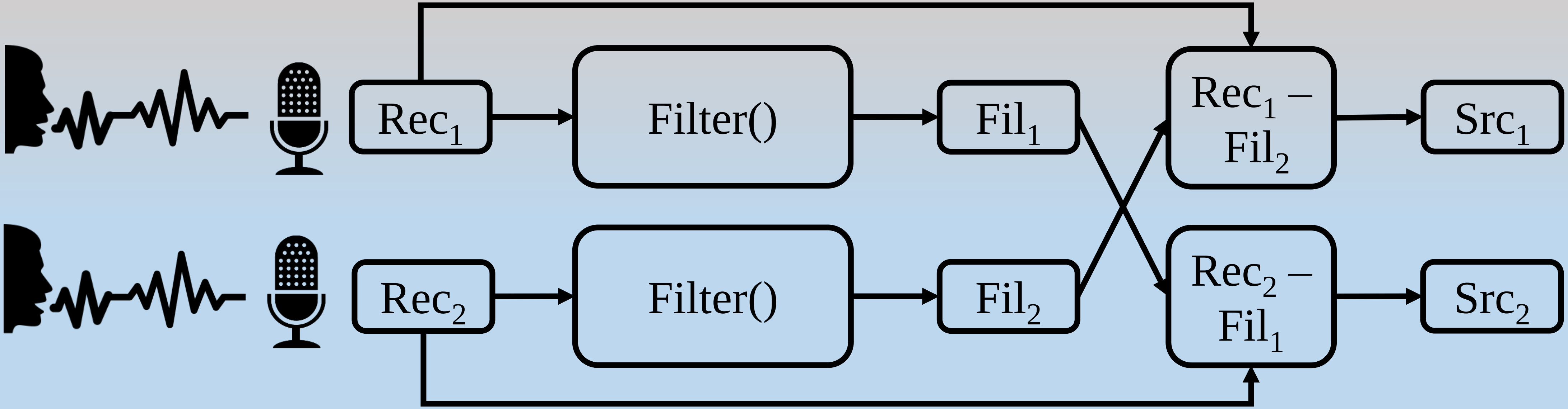
Reconstruction Low-pass Filter for a Voice-enabled IoT Prototype

- **Problem**: Two Echo Dots are placed at the proximity of each other and two persons simultaneously talk with their proximate Dot, the voice recorded by each Echo Dot is distorted by a low frequency voice of the other party.
- **Goal**: Process both recordings recorded by the Echo Dots and then extract and execute both issued commands.



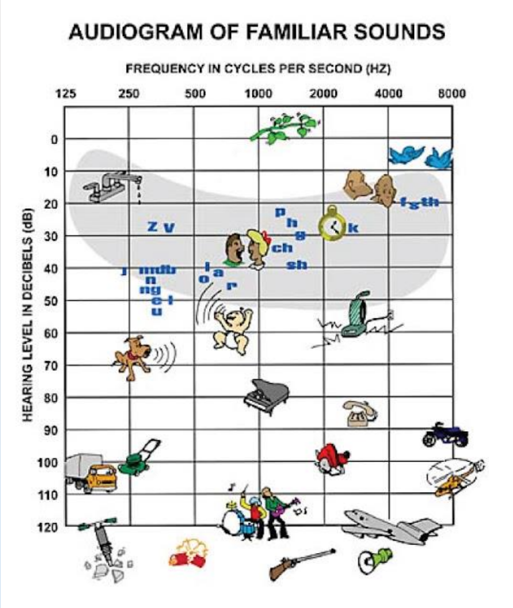
Proposed Reconstruction Lowpass Filter (RLF)

- The Butterworth filter is used to build the proposed Reconstruction Lowpass Filter (RLF)



Proposed Reconstruction Low-pass Filter (RLF)

- Consider the recorded voice from each microphone rec_i is a mixture of source signals src_i, noise signals $noise_i$, where $i \in \{0, 1\}$ and filtered voice fil_j is an approximation of the noise:



  - $rec_i = src_i + noise_{(i+1 \bmod 2)}$
  - $src_i = rec_i - noise_{(i+1 \bmod 2)}$
  - $src_i = rec_i - fil_j \qquad i \neq j$

- In this work we used a 6th order Butterworth filter with the cut-off frequency of 500 Hz.

Dataset for Blind source Separation

- Two Persons are participated in the study
  - Voices are stored as *wav* audio format
  - Available online: https://github.com/hmofrad/viota
- Different proximities to the microphones (Person$_i$, microphone$_i$)
- Common smart speaker commands are used.

Dataset	Number of sentences	Microphone proximity
Dataset 1 (near)	30	Near
Dataset 2 (far)	44	Far

Discussion Outline

- Motivations and contributions
- Background
- Proposed voice-enabled IoT prototype
- Reconstruction lowpass filter for a voice-enabled IoT prototype
- **Results**
- Summary and conclusion

Results

- Performance metric we use is Word Error Rate, WER = (S + D + I)/N
  - #**S**ubstitutions
  - #**D**eletions
  - #**I**nsertions
  - #**N**umOfWords
- WER is widely used in speech processing and NLP
- Algorithms are:
  - Baseline model which uses the raw recording files
  - Reconstruction Independent Component Analysis (RICA)
  - The proposed Reconstruction Lowpass Filter (RLF)

Results

- RICA performs the worst because it overfits the input recordings.
- The proposed RLF has overall improvement of 2-3% compared to the Baseline model
- Our results are always better for both datasets.

Dataset	Microphone	Baseline	RICA	RLF
Dataset 1 (near)	Mic_1	0.96 ± 0.11	0.91 ± 0.22	**0.99 ± 0.03**
	Mic_2	0.95 ± 0.13	0.35 ± 0.37	**0.96 ± 0.12**
	$(Mic_1+Mic_2)/2$	0.95 ± 0.12	0.63 ± 0.29	**0.97 ± 0.08**
Dataset 2 (far)	Mic_1	0.96 ± 0.10	0.95 ± 0.13	**0.98 ± 0.04**
	Mic_2	0.44 ± 0.39	0.18 ± 0.39	**0.47 ± 0.40**
	$(Mic_1+Mic_2)/2$	0.70 ± 0.24	0.56 ± 0.26	**0.73 ± 0.22**

Discussion

- The 2-3% improvement may not be a groundbreaking improvement at the first glance but
  - Our results are better than both Baseline and RICA models
  - At scale it significantly contributes to the Cloud throughput, availability, and utilization by reducing the number of commands send by users.
- Avoid potential Cloud upgrades and expansion
  - Reduce number of retries due to accuracy
  - Keep the number of requests low
  - Requests are now less noisy ➔ will result in intended action

Summary and Conclusion

- A customizable voice-enabled IoT prototype is proposed which can be used as a preprocessing step to the speech-to-text API
  - Raspberry Pi
  - Google Cloud speech-to-text API
  - Text-to-intent API
- Devising a method for voice separation in IoT environment.
  - Reconstruction Lowpass Filter (RLF)
- Takeaways
  - A good preprocessing can eliminate potential retries on the Cloud
  - This is achievable with a inexpensive hardware.