A Simple and Lightweight Algorithm for Social Robot Speech Turn Taking

Adriana Lorena González
University of Manitoba
Winnipeg, MB, Canada
gonzala1@myumanitoba.ca

James E. Young
University of Manitoba
Winnipeg, MB, Canada
young@cs.umanitoba.ca

ABSTRACT

Simple, but effective, social robot voice-based interaction designs are possible with only rudimentary speech analysis. Robust full analysis, including syllable, word, and meaning extraction, is still an open research problem, with existing solutions being computationally expensive (and thus a power drain) while suffering from high error rates. Instead, we note that it is sufficient for many social interactions for a robot to simply distinguish when a person starts and finishes talking, and indeed argue that designing social robot interactions around such a constraint may – in the short term – result in more robust behaviors. We introduce a simple solution, using standard lightweight signal processing techniques (i.e., involving the derivative of the audio's RMS value), that detects the beginning and end of speech utterances, including a preliminary evaluation. We envision that this simple, easy to implement algorithm may be useful for researchers aiming to simply and quickly implement basic robotic speech turn taking on low-capability or power-constrained robot devices. Further, the approach can support innovation in simple conversation with social robots.

CCS CONCEPTS

Human-centered computing ~ Human computer interaction (HCI) ~ Interaction techniques

KEYWORDS

Social Robots, Technical Implementation

1 Introduction

Although human-like conversation with a social robot is highly sought after in human-robot interaction, accurate and reliable speech recognition is unfortunately recalcitrant. In practice, speech recognition errors such as misinterpreting what a person said or failing to acknowledge a person is talking [1], [2], severely hinders interaction. We propose a constrained design challenge that focuses on robust, but simplified, robotic capability: speech-based interactions which do not require a robot to understand speech content, but only to identify when a person starts and stops talking (e.g., see [3]). For example, a pet-like robot might want to wait for the person to finish talking to it before doing their sleep routine or making a sound to show that it wants to play more. Although this does not reduce the need for more complex speech recognition, we believe this approach to be more feasible and robust in the short term for deployed social robots. In this paper, we present a simple and lightweight algorithm for realizing this, enabling a social robot to understand when a person begins and stops talking.

Recent full-featured speech recognition solutions often require significant computation resources, and as a result are typically deployed on cloud-based services (e.g., Google's [1] or Microsoft's [2] solutions); however, these still have accuracy problems [4], [5], particularly given fast or slow speech, non-common words, or accents [6]. Further, on-line systems introduce network infrastructure, cost, and delay challenges. Completely offline systems, with reduced capability for less-capable computers, tend to lack the accuracy to be useful for many tasks [2]. These challenges are due to the shear complexity of true speech recognition, which requires an algorithm to correctly identify syllables, words, and full phrases that the person is saying, despite background noise, changes in speech, and unexpected transient sounds. The output then must be analyzed and used within context to create an appropriate response.

For research, a common solution is to employ the Wizard of Oz technique (researcher controlling the robot remotely without a participant knowing [7]); some claim this is the most reliable way to do speech recognition [1], [4]. However, this approach is not deployable for use in practice, or unsupervised research such as longitudinal field studies. For deployed robots, solutions typically employ enforced structure or stratification to simplify the problem space. A common approach is key-phrase spotting, where algorithms only search for static predefined phrases [4], [8] (e.g., "How's the weather?", "Can you dance?"). This unfortunately suffers from rigidity, where a person can be ignored or completely misunderstood with even small variants from a pre-programmed phrase (e.g., "what is the weather today" versus "how's the weather?").

In this paper we present a simple, light-weight algorithm using standard signal processing techniques that enable a social robot to easily recognize speech start, pause, and stop.

[1] https://cloud.google.com/speech-to-text/
[2] https://azure.microsoft.com/en-ca/services/cognitive-services/speech-to-text/

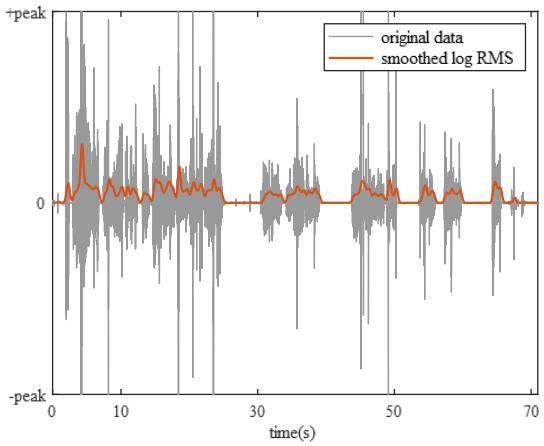

Figure 1: Audio from robot (71 seconds, grey) and the smoothed RMS (1/4 second window, orange). Note how the RMS clearly and stably shows amplitude across frequencies.

2 Approach

To detect important parts of speech: the start, small pauses, and end, we analyze the amplitude of a robot microphone audio signal. However, simply monitoring the amplitude (e.g., and thresholding it) is fragile, as it rapidly changes, and the target peaks (speech loudness) would depend on ambient noise, distance of speaker, and other related factors. Our solution is the following:

1) Re-sample the audio to a lower-than-typical rate, to around 600 Hz. *Why*: high frequencies may be important for understanding details of speech, but are irrelevant for detecting start and stop of talking. Lower sample rate dramatically reduces computational cost.
2) Calculate Root Mean Square (RMS) of the waveform with a 0.25 second window. *Why*: RMS provides a more accurate measure of how power (loudness) changes over a time window and frequency range than simple amplitude [9], which is rapidly changing. 0.25s was selected ad-hoc to balance large window (smoother result) with minimizing delay caused by require look-ahead.
3) Calculate the logarithm of the RMS. *Why*: perception of audio loudness has a logarithmic relationship to power. This transform compresses the range, making the low amplitude signals higher and lowering the high amplitude, resulting in a more balanced dynamic range reflective of how people hear.
4) Smooth the log RMS using a gaussian kernel over a 1 second window. *Why*: reduce the impact of rapid changes (less than 1 second) and focus more on the general volume over the 1 second window. (Original and smoothed RMS data seen in Fig. 1). This adds a slight additional delay to detection.
5) Take the derivative of the smoothed data. *Why:* enables us to analyze the *change* in RMS amplitude as well as the RMS amplitude itself. We use this to detect sudden increases and decreases, irrespective of changing background noise or closeness to the microphone.

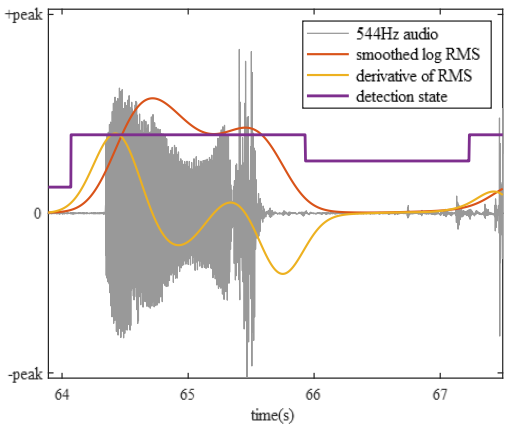

Figure 2: All main stages of signal processing, from raw audio signal (grey), to smoothed RMS (red), derivative (yellow), and finally threshold result (purple); threshold is low (not talking), high (talking), middle (paused). Note how the RMS, and thus the thresholding, *looks ahead* due to the window size. This introduces a detection delay of approximately 0.2 seconds.

6) Test the derivative and filtered RMS value against thresholds to detect phase of speaking. Starting from no talking detected, monitor the derivative for a positive spike to mark beginning of talking. While talking, if there is a simultaneous low RMS value and negative derivative, assume a pause (see fig 2). If the pause continues without a new sufficient positive derivative spike, assume talking has ended.

Figure 3 demonstrates all the stages of our processing in a 14 second window, from raw data, to smoothed log RMS, derivative of the log RMS, and finally the thresholding result showing a short pause and a stop. Currently, our exact window sizes and threshold values (see next section) were chosen through trial and error. We are working toward automatically tuning these values based on the environment, for example, using baseline RMS noise over a given timeframe to set the target thresholds.

3 Sample Code

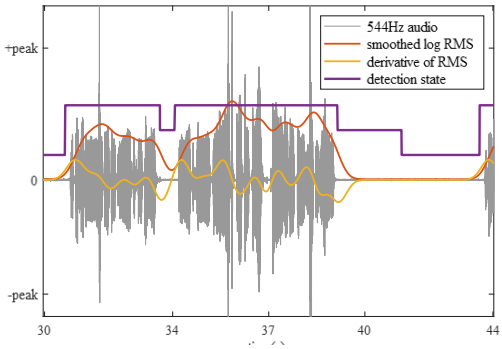

Figure 3: Detection of talking state in all the phases, shown by the purple line; threshold is low (not talking), high (talking), middle (paused). Note how the pause eventually resolves to talking stopped.

We provide a sample code sketch here, including our specific parameters, to clarify the algorithm. A complete working MATLAB sample file is available permanently online <link removed for anonymization>; this code was used to generate graphs 1-3.

```
data = csvread("44.1KHzPCMdatafile.csv");

% down sample PCM data, 44.1 kHz to 544.4Hz
reducedData = resample(data, 1,81);

% get RMS values and use the kernel to smooth the data
RMSData=zeros(size(reducedData,1),1);

for i = 137:size(RMSData,1) % .25 sec is 136 samples
  RMSData(i) = rms(reducedData(i-136:i));
end

RMSData = log10(RMSData+1); % match perception

% use 1 second Gaussian filter
RMSDataFiltered = conv(RMSData, GaussianFilter, "same");

% get derivative of RMS data
Derivative = conv(RMSDataFiltered, [1 -1], "same");

% set treshold. RMS treshold based on "silence" at start
DerivativeThresh = .004;  % chosen via trial and error
RMSThresh = max(reducedData(1:250))*1.5;

%test for status
for i = 1: size(reducedData,1)
    if (status ~= talking && ...
            Derivative(i) > DerivativeThresh)
        status = talking;
    end
    if (status == talking && ...
            Derivative(i) < -DerivativeThresh && ...
            RMSDataFiltered(i) < RMSThresh)
        PauseStart = i;
        status = smallPause;
    end
    if (status == smallPause && ...
            (i - PauseStart) > 1088) %pause > 2 sec
        status = stop;
    end
    detection(i) = status;
end
```

This code works by having full access to the waveform and processing it at once. For a live robot, we simply process live by adding the blocks received from the sound driver to our processing chain as they arrive. The only resulting delay is the larger of either our window size or the driver block size.

5 Informal Validation

We performed an initial validation by applying our algorithm against a range of samples from LibriSpeech [10], a dataset of read audiobooks in English sampled at 16 kHz widely used to train automatic speech recognition models; thus, we down sampled by a factor of 27, instead of the 81 previously used. We used 10 samples from their test set: 5 from their "clean" speech samples and 5 of "other" speech. We chose the longest samples we could find, ranging from 19 to 33 seconds and from different readers. Because these audio samples are cut to not have long pauses, we can only test the ability to recognize pauses and not fully stopping to talk. However, this is not a problem since the stopped state is only when the pause has been going on for too long.

For these tests, the derivative threshold was set to 0.004 and the RMS threshold was tuned using the baseline approach

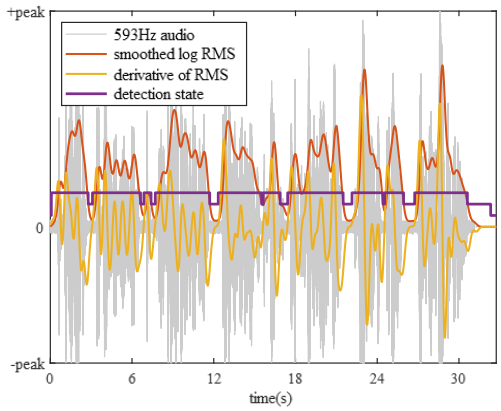

Figure 4: Test with dataset sample. Talking state only goes to a pause whenever it is a definite short pause and not while the person is still uttering something.

mentioned earlier section 2: the RMS threshold was calculated as 50% higher than the highest value in the first 250 samples of silence. We calculated success through visual inspection.

Figure 4 shows one example, where the reader of the book section did multiple short pauses. All samples tested, from both the clean and other datasets, always registered a pause when a pause was present. If a person was breathing heavily between utterances, it would sometimes fail to recognize this as a pause.

6 Future Work

While our initial prototype for detecting start, pause, and stop in speech has initial signs of success, we need to implement automatic detection and tuning of all parameters based on background noise and qualities of a particular robot microphone. Further, to better understand the limits of our approach we need to conduct a more formal and thorough evaluation, including a wider range of conditions and situations (e.g., background conversation noise, robot motor noises, etc.).

7 Conclusion

In this paper we present a simple and lightweight solution to capturing the beginning and end of utterances and conversation by a user. This leverages simple signal processing and is able to give reliable results without the need of computationally expensive algorithms or libraries that might not be available in different programming languages or robotic platforms. We conduct an informal evaluation of our solution where we observe initial positive results. This simple, lightweight algorithm supports exploring novel social robot designs and speech-based interactions that do not rely on the content of the speech. We envision that this approach can be used when computing power is limited or robust interaction is prioritized, and our algorithm can be a starting point for those wishing to prototype and explore such behaviors.

REFERENCES

[1] O. Mubin, J. Henderson, and C. Bartneck, "You just do not understand me! Speech Recognition in Human Robot Interaction," *IEEE RO-MAN 2014 - 23rd IEEE Int. Symp. Robot Hum. Interact. Commun. Human-Robot Co-Existence Adapt. Interfaces Syst. Dly. Life, Ther. Assist. Soc. Engag. Interact.*, pp. 637–642, 2014.

[2] V. A. Kulyukin, "On natural language dialogue with assistive robots," *HRI 2006 Proc. 2006 ACM Conf. Human-Robot Interact.*, vol. 2006, pp. 164–171, 2006.

[3] A. L. González and J. E. Young, "Please Tell Me about It: Self-Reflection Conversational Robots to Help with Loneliness," *HAI 2020 - Proc. 8th Int. Conf. Human-Agent Interact.*, pp. 266–268, 2020.

[4] J. Kennedy *et al.*, "Child Speech Recognition in Human-Robot Interaction: Evaluations and Recommendations," *ACM/IEEE Int. Conf. Human-Robot Interact.*, vol. Part F1271, pp. 82–90, 2017.

[5] J. Novoa, J. Wuth, J. P. Escudero, J. Fredes, R. Mahu, and N. B. Yoma, "DNN-HMM based Automatic Speech Recognition for HRI Scenarios," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 150–159.

[6] S. Goldwater, D. Jurafsky, and C. D. Manning, "Which words are hard to recognize? Prosodic, lexical, and disfluency factors that increase speech recognition error rates," *Speech Commun.*, vol. 52, no. 3, pp. 181–200, 2010.

[7] L. D. Riek, "Wizard of Oz Studies in HRI : A Systematic Review and New Reporting Guidelines," vol. 1, no. 1, pp. 119–136, 2012.

[8] C. Breazeal, C. D. Kidd, A. L. Thomaz, G. Hoffman, and M. Berlin, "Effects of nonverbal communication on efficiency and robustness in human-robot teamwork," *2005 IEEE/RSJ Int. Conf. Intell. Robot. Syst. IROS*, pp. 708–713, 2005.

[9] Wikipedia, "Audio Power," 2018. [Online]. Available: https://en.wikipedia.org/wiki/Audio_power#Measurements. [Accessed: 13-Dec-2020].

[10] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2015-Augus, pp. 5206–5210, 2015.