DSP CONCEPTS

1800 Wyatt Dr. • Suite 14
Santa Clara, CA 95054
408-747-5200

www.dspconcepts.com

FUNDAMENTALS OF VOICE UI

Voice UI–or voice user interface–features found in the Amazon Echo and Google Home have captured the attention of consumers. This paper outlines the basic concerns product developers face when creating and optimizing voice UI products; examines ways of measuring and evaluating them; and recommends best practices in voice UI systems engineering.

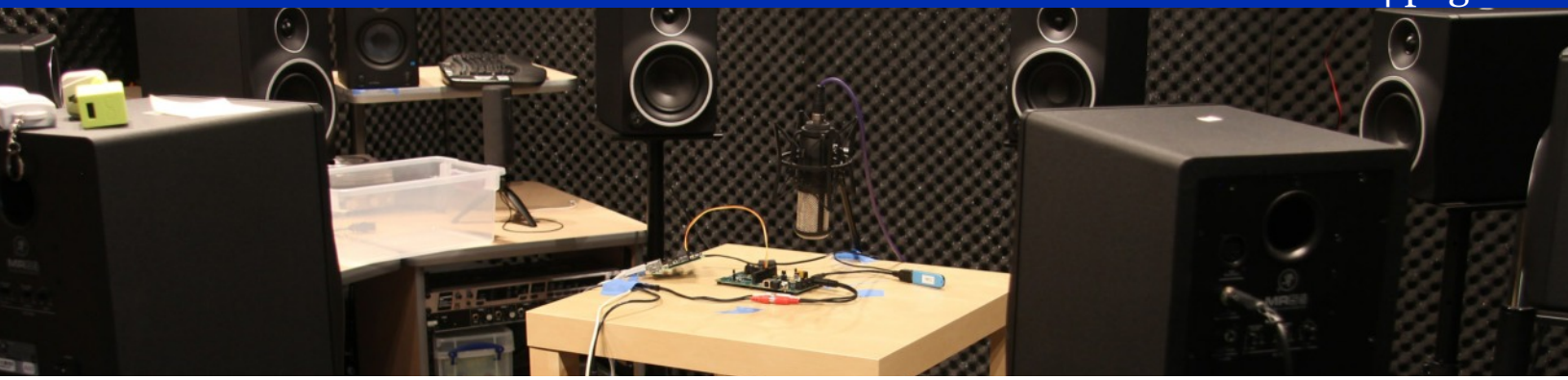
The runaway success of the Amazon Echo and Echo Dot smart speakers, of which 10 million are expected to ship in 2017 alone, has made voice command—often known as voice user interface, or voice UI—an in-demand feature in new tech products. The feature is already included in every smartphone and tablet, in most new automobiles, and in a fast-growing number of audio products. It's not unreasonable to expect that eventually, most home appliances, audio and video products, and even wearables such as fitness trackers, will feature voice command.

"The better the ratio of desired signal (the user's voice) to noise (any other sounds), the more reliably a voice UI system will work."

Now that millions of voice UI products are out in the field, we can start to see what consumers will expect of these devices—and how challenging it will be to meet those expectations. The limited success of previous, more primitive voice command products suggests that beyond learning a trigger word, such as "Alexa" or "OK Google," consumers are unwilling to adapt their behavior to the requirements that a more primitive voice command product may place on them, such as pushing a button to wake the device up, or speaking directly into a remote control. In the home, at least, consumers expect a voice UI product to respond to commands spoken from across a room, and if possible, even from a different room. They expect dependable voice recognition no matter what the acoustical properties of a room, and no matter where the product is placed in the room. And they expect voice UI to work even in the presence of moderately loud environmental noise.

Although sophisticated voice recognition systems rely on Internet-based computing power, much of the performance of a voice UI system depends on the quality of the voice signal that the system receives. The old maxim "garbage in, garbage out" applies as much to these systems as it does to any other technology. The better the ratio of desired signal (the user's voice) to noise (any other sounds), the more reliably a voice UI system will work.

Voice UI systems receive their commands using multiple microphones, usually arranged into arrays controlled through digital signal processing. Much of the accuracy of a voice recognition system depends on the



ability of these arrays to focus on the user's voice and reject unwanted stimuli, such as environmental noise or sounds emanating from the device itself. Most of the research into optimizing these arrays and the algorithms that control them is closely held by the companies that have pioneered these products, leaving product developers with few, if any, references that can help them make informed design tradeoffs.

Complicating matters is the unfamiliarity of microphone array design. Although countless engineers possess expertise in loudspeaker design and applications, far fewer have comparable experience with microphones—and while most engineers' ears can usually give them at least a rough idea of what's wrong with a speaker, it's much more difficult to assess microphone performance problems. The challenge becomes increasingly complex when the number of microphones is multiplied for an array. Now the engineer must determine which types of microphones will work best for the array, what number of microphones to use, and in what physical configuration to place them.

A processing algorithm is then needed to allow the array to identify the direction of the user's voice and focus on that voice while rejecting other sounds. Many such algorithms are available, and all must be optimized for the performance of the microphones, the size and configuration of the array, and the acoustical effects of the enclosure in which they are mounted.

The goal of this paper and the follow-up paper, "Optimizing Performance of Mic Arrays and Voice UI Systems," will be to address each of these variables, using data to show the effects of the various design choices, and to give product design engineers the information they need to make the most appropriate choices for their application. This paper will start by explaining the basics of microphone arrays and voice UI algorithm functions.

Microphones Used in Voice UI

Almost all of the microphones used in voice UI products are monophonic MEMS (Micro Electrical Mechanical Systems) types. MEMS microphones offer numerous advantages in the design of microphone arrays for voice UI products:

Compact size: MEMS microphones are typically no larger than 5mm on each side, making it possible to fit as many as seven mics inside a small product form factor. Surface-mount design further reduces their footprint.

Low cost: As the number of microphones in a product multiplies, cost can become an important consideration. MEMS mics are manufactured with the methods used for integrated circuits, so they tend to be inexpensive. They can also interface directly with processors that have PDM (Pulse Digital Modulation) ports, without needing costly A/D converters.

Consistency: Predictable functionality of a microphone array demands that the mics within the array be well matched. Because MEMS microphones are manufactured using a completely automated process much like that used to manufacture ICs, unit-to-unit consistency is typically good.

Most of the microphones used in voice UI products are omnidirectional, receiving sound equally from all directions. Because the directionality of a microphone array is created through an algorithm rather than through inherent directionality of the microphones, using omnidirectional microphones allows the algorithm full flexibility in the way it processes the various mic signals to create directional pickup beams.

Another benefit of omni microphones is that they have flatter frequency response than directional (cardioid) mics. This characteristic reduces the processing load on the algorithm, and thus orientation of omni mics in product assembly is not critical.

Within the available selection of MEMS microphones, the mic array designer can select from a range of capabilities and qualities, including sensitivity, noise, frequency response matching, and digital and analog output. Some of these characteristics will be addressed in the follow-up to this white paper, "Optimizing Performance of Mic Arrays and Voice UI Systems."

Components of a Voice UI Algorithm

The algorithm that makes a voice UI product possible is actually a collection of several algorithms, each with a specific function that helps the microphone array to focus on a user's voice and ignore unwanted sounds. Here is a brief description of the algorithms typically used in voice UI.

Trigger/Wake Word

A voice UI system uses an assigned trigger word—such as "Alexa" or "OK Google"—that the user employs to activate the voice UI device. Recognizing this trigger word presents challenges because the device must do the recognition immediately, using its own algorithm on device; using Internet resources would create too much delay. The device must always be active to some degree because it must constantly listen for the trigger word.

Choosing an appropriate trigger word to incorporate into the algorithm is critical to recognition of the trigger word and thus the operation of the voice UI device. The trigger word must be sufficiently complex to produce a distinctive waveform at the microphone output that the algorithm can easily distinguish from normal speech, otherwise the percentage of successful recognition may be unacceptably low. The trigger word must not be a word or phrase that will be commonly used, otherwise the rate of false triggering may be unacceptably high. It also should not be too long, because the longer the phrase, the more likely that a consumer will find using the device cumbersome. Typically, a trigger word using three to five syllables is the best choice.

When evaluating the performance of a trigger word algorithm, there are two main factors to consider. First, how often does the algorithm indicate a trigger when none is present? This is measured as false alarms per hour. Second, how well does the algorithm correctly detect the trigger phrase in the presence of background noise? This is measured as detection percentage.

Most trigger algorithms come in different sizes. Small algorithms take less memory and processing but make more mistakes; large algorithms require more resources but make fewer mistakes. Models are also tunable, allowing product designers to make them stricter (fewer false alarms but more difficult to trigger) or more lenient (more false alarms but easier to trigger). Most product designers choose stricter tunings, because while customers tend to accept the need to repeat themselves occasionally when issuing a command, they are less forgiving of false triggers.

"Choosing an appropriate trigger word to incorporate into the algorithm is critical to recognition of the trigger word and thus the operation of the voice UI device."

False alarms are measured by playing hours of spoken content and counting how often the algorithm produces a false trigger. Under this test, the difference in performance of the different model sizes becomes obvious. **Figure 1** on the next page compares the performance of a small, medium and large trigger models for different tuning points. These conditions are challenging, with almost continuous spoken dialog presented to the trigger word algorithm. Achieving fewer than two false triggers per hour under these conditions is a reasonable goal. The small model is able to achieve this only with the two strictest tunings on the far-left hand side of the graph. The medium and large models achieve this goal over a wider operating range.

.

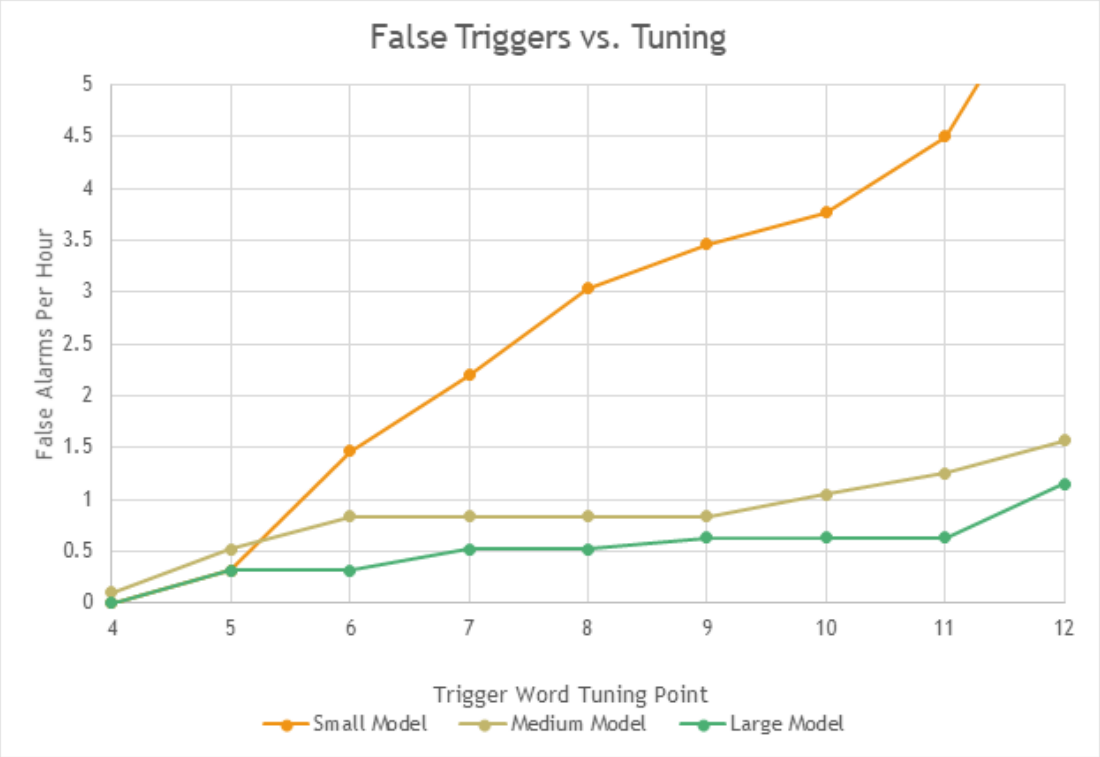


***Figure 1:* False alarms per hour tested with small, medium and large algorithm models, with stricter tuning to the left and more lenient tunings to the right.**

When measuring the performance of trigger algorithms in noise, our research shows that the main factor determining trigger detection rate in the presence of environmental noise is the signal-to-noise ratio (SNR) measured at the microphone. The "signal" represents how loud the person's voice is at the microphone and the "noise" is the level of the background noise. In our testing, we use "babble" noise that simulates typical noise and light conversation in a home. Graphs for the three model sizes are shown in **Figure 2** on the next page and all models have been tuned to achieve fewer than two false triggers per hour. The X-axis represents the SNR, with higher SNRs (cleaner signals) towards the right. The Y-axis is the probability of detection. For the most part, the algorithms have the same performance within 1 or 2 dB.

We should note here that an SNR of 10 dB or so may seem unacceptable when compared with SNR numbers of 80 to 120 dB for most audio playback equipment. However, in voice UI applications the user's voice is often only a few dB louder than the surrounding noise, and as the chart below shows, an SNR of 10 to 20 dB can deliver excellent results in voice UI applications. Accordingly, a 2 dB increase in SNR can significantly improve voice UI performance, even though the same increase would likely be subjectively unnoticeable in audio playback systems.

"... a 2 dB increase in SNR can significantly improve voice UI performance, even though the same increase would likely be subjectively unnoticeable in audio playback systems."

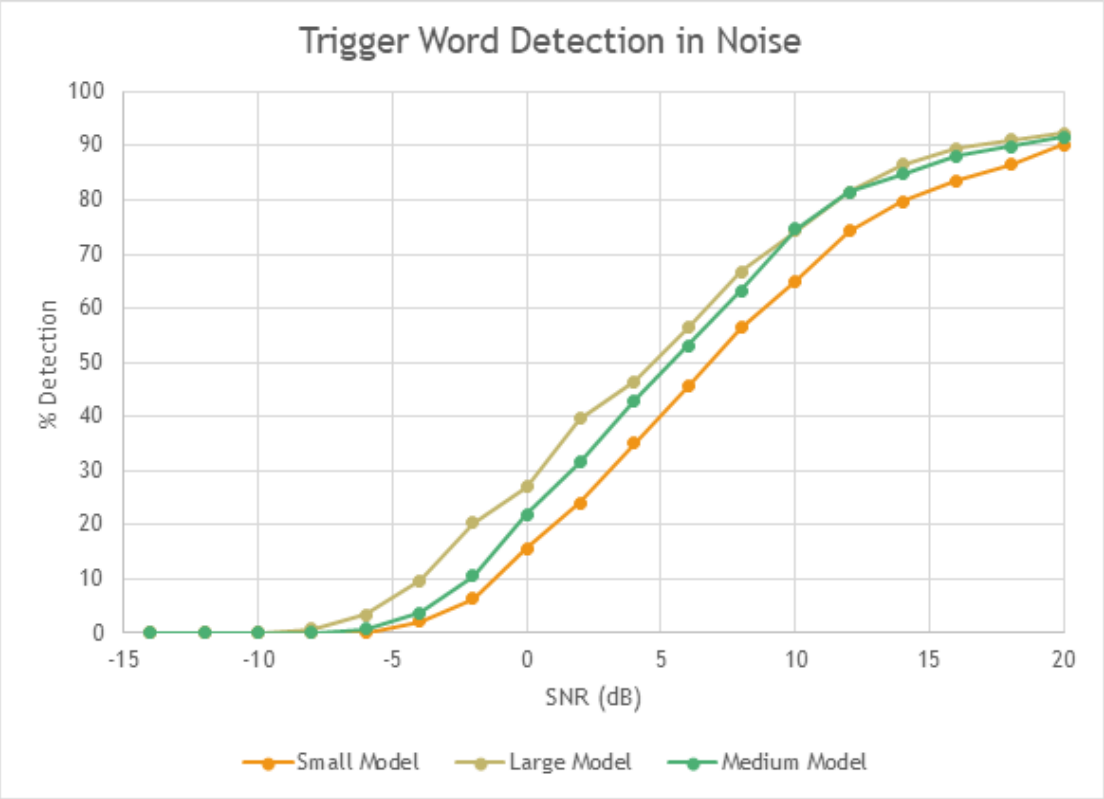


Figure 2: ***Performance of trigger word detection as a function of SNR. Three different model sizes were tested and the larger the model the better it performs.***

DOA (Direction of Arrival)

Once the trigger word has been recognized, the next step is to determine the direction of arrival of the user's voice. Once the direction is determined, the DOA algorithm tells the beamformer algorithm in which direction it should focus.

The core function of the DOA algorithm is to examine the phase or time delay relationship of the signals coming from the different microphones in the array, and use this information to determine which microphone received the sound first. However, the task is far complicated than it may seem. Because of reflections from walls, floor, ceiling and other objects in the room, the sound of the user's voice will also be arriving from other directions, not just directly from the user's mouth. The initial sound is all that is wanted for DOA determination, and the later reflections must be filtered out. To this end, a DOA algorithm includes precedence logic, which separates the louder initial arrival from the quieter reflections. This function electronically eliminates acoustical reflections within a room, and if carefully tuned, the algorithm is even able to reject reflections off nearby surfaces, such as a wall directly behind a smart speaker.

The DOA algorithm's operation is enhanced by automatically adjusting for the ambient noise level. The algorithm measures average noise level in the room, and will only recalculate the position of the user's mouth if the incoming signal is at least a certain number of decibels above the level of the ambient noise. This way, the system can lock onto a specific direction without being "distracted" by noise that is relatively low in level.

We measure the accuracy of the DOA algorithm by surrounding the microphone array with eight speakers uniformly spaced on a circle of radius 1 meter. All eight speakers play diffuse field background noise while one speaker plays the trigger phrase in addition to the noise. Voice level is fixed at 60 dBA measured at the microphones and the level of the diffuse field noise is varied. We vary which speaker is used to play the trigger phrase and measure which speaker we classify the sound as arriving from. This produces the chart shown in **Figure 3** on the next page.

.

	0	1	2	3	4	5	6	7
0	3300	365	0	299	0	0	0	0
1	0	3629	0	0	0	335	0	0
2	0	0	2692	0	1000	0	0	129
3	308	0	0	3656	0	0	0	0
4	0	344	0	0	3420	200	0	0
5	0	480	0	0	0	3484	0	0
6	0	316	0	0	0	148	3000	500
7	0	0	0	0	52	0	200	3701

***Figure 3:** Confusion matrix showing the results of the direction of arrival tests. The row index corresponds to the actual direction that the sound arrived from; the column index indicates which direction was returned by the DOA algorithm. Ideally, the only non-zero values should be along the diagonal of the matrix running from top left to bottom right.*

It is more useful to condense a confusion matrix down into a single number which represents the overall accuracy of the algorithm at a specific noise level. In the DOA algorithm, we weight errors based on how far they are from the correct value, so the single-number result we use is the error in degrees at a certain SNR. A result of 0 would be perfect DOA determination, and a result of 180 (the opposite direction from 0 degrees) is equivalent to the algorithm getting the most inaccurate possible result. **Figure 4** below shows that the DOA algorithm under test here starts to perform reliably at SNR above 0 dB. From then on, the algorithm performs well with a very small average error.

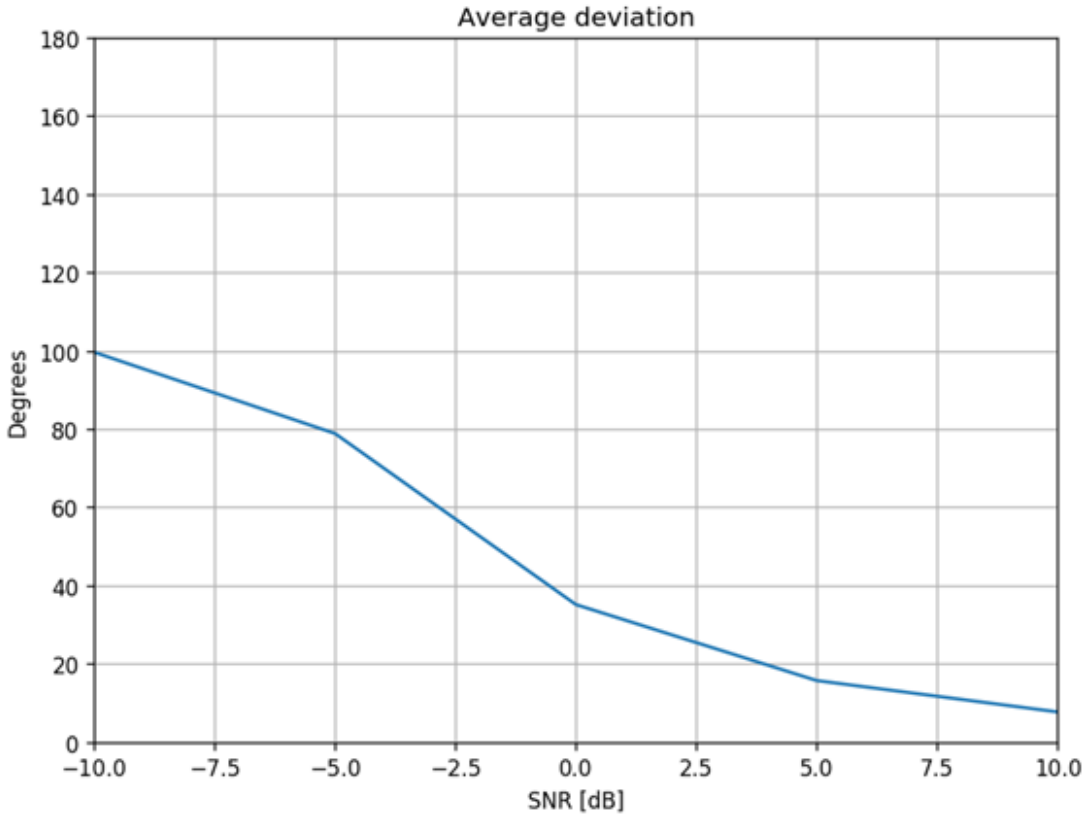


***Figure 4:** Consolidated DOA results. The X axis represents the signal-to-noise ratio of the wake word utterance. The Y axis is the deviation error in degrees. The DOA algorithm starts to deliver useful performance at SNR above 0 dB, and accurate performance at SNR is above 5 dB.*

AEC (Acoustic Echo Canceller)

In a voice UI device that incorporates speakers, such as a smart speaker or car audio system, one source of noise that interferes with voice commands is the speakers themselves, which may be playing voice feedback, music, radio, etc. The voice UI device must subtract the sounds its speaker produces from the sounds picked up by the mics.

This may seem as simple as blending a phase-reversed version of the program material into the signals coming from the microphones, with a slight delay added to compensate for the time it takes for the sound to travel from the speaker to the microphones. However, this process is merely the starting point of an AEC algorithm; it is inadequate to deal with the numerous complications that real world applications present.

The first complication is that the waveform of the program material is altered by the speaker, by the DSP used to equalize the speaker, and by the microphones used in the array. Fortunately, it is possible to compare the outputs of the microphones with the original (pre-DSP) input signal and calculate correction curves that the algorithm can use to subtract the direct sound from the speaker from the waveform of the voice command.

However, the program material is also affected by acoustical reflections. These reflections may number in the thousands, and in a large living room they might arrive at the mics as much as 1 second after the direct sound from the speaker. The spectral content of the reflections will differ from the content of the direct sound from the speaker, depending on room modes and the absorptive effects of room furnishings. These effects will be different in every setting, and will change as people and pets walk around the room, or as the number of occupants in a vehicle changes.

In order to subtract enough of these acoustical echoes from the microphone signals to achieve an acceptable SNR, the AEC algorithm must “look for” sounds that match the program material within a certain margin of error (to compensate for changes to the waveform caused by acoustics), and over a defined time window that corresponds to the expected reverberation time. Because of the distances between the microphones in an array, each microphone receives a slightly different set of echoes and a different direct sound from the speaker, so achieving maximum SNR requires separate AEC processing for each microphone.

The performance of an echo canceller is usually defined by its “echo return loss enhancement,” or ERLE. This is the gain reduction, in dB, that the echo canceller is able to reduce the speaker signal at the microphone. Echo cancellers should achieve at least 25 dB of cancellation for good performance. The best ones are able to cancel over 30 dB.

The time period over which the AEC looks for reflections is called the “echo tail length.” The longer the echo tail length, the more reflections can be canceled and the better the algorithm performs. Longer tails, however, require more memory and more processing. **Figure 5** below shows the echo return loss as a function of the tail length. This measurement was done in a semi-anechoic sound room. You can see that most of the cancellation has been achieved with 200 msec of tail length and longer tails provide only marginal improvements.

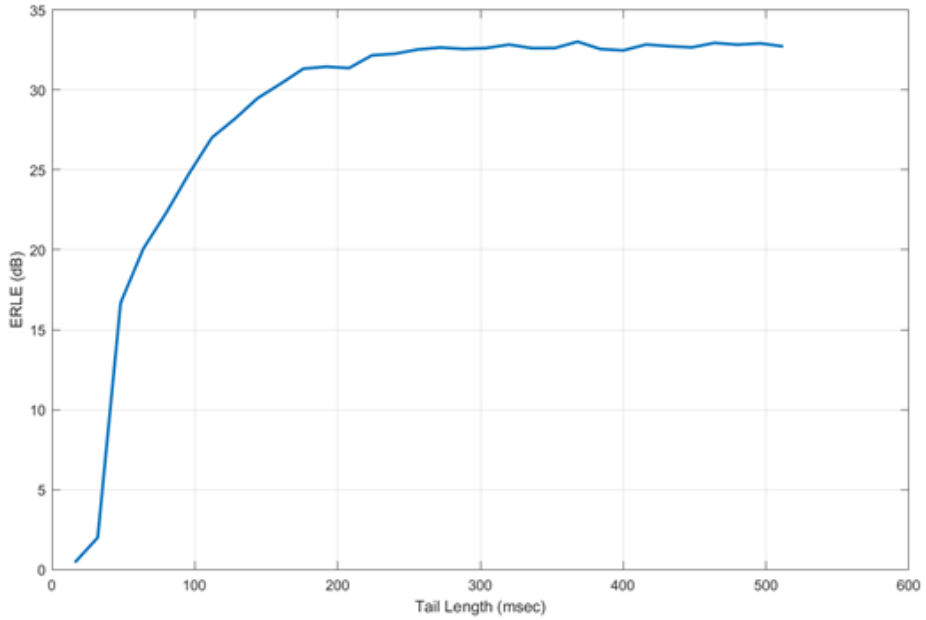


***Figure 5:** Echo canceller performance as a function of the tail length. This measurement was done in a semi-anechoic sound room and it shows that there is little improvement after 200 msec.*

A semi-anechoic room is fairly easy to deal with and does not represent real-world usage. **Figure 6** below shows the performance of the echo canceller in progressively more reverberant rooms. The need for longer echo tails is now obvious and the most reverberant space could benefit from an even longer echo tail.

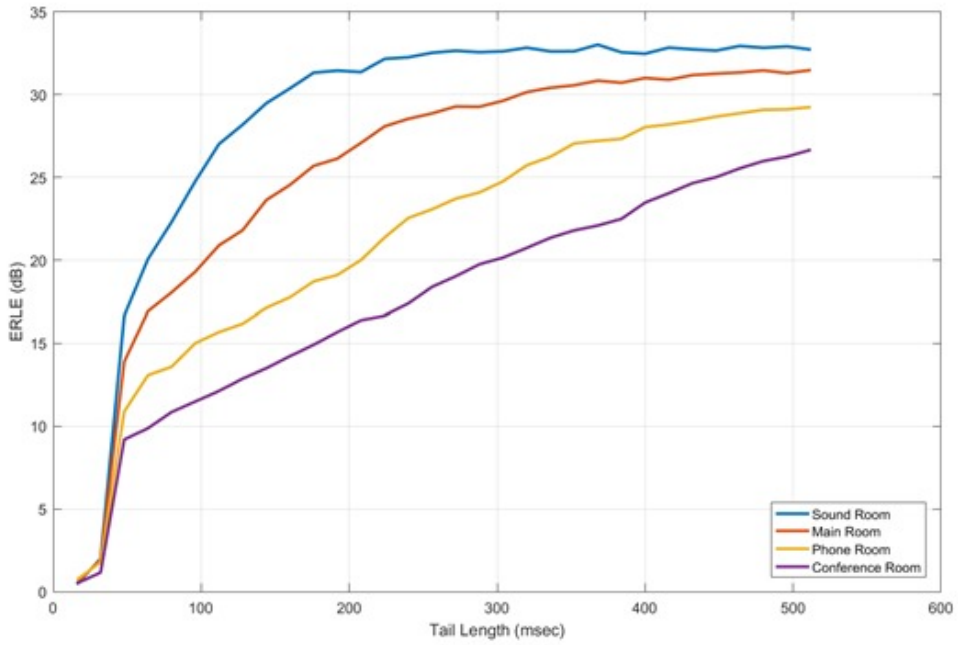


***Figure 6:** Echo canceller performance in four rooms with increasing reverberation time. The larger rooms benefit from algorithms using long echo tails.*

AEC algorithm performance is better when the speaker performs in a linear fashion. If the speaker exhibits distortion at loud levels, then distortion components (harmonics) will be generated and the AEC will not recognize these as reflections of the original program material, and thus cannot cancel them. The total harmonic distortion (or THD) of a loudspeaker is a measure of its linearity. THD is reported as a percentage of signal level and the lower the THD, the more linearly the speaker behaves. The distortion components of the loudspeaker will be present in the output of the AEC since the AEC is unable to cancel them.

It is possible to quantify how the distortion will impact the performance of the AEC. For example, if the loudspeaker has 1% THD, then the distortion components will be 40 dB lower than the signal level. If the echo canceller has an ERLE of 30 dB, then a THD of 1% is acceptable. Now consider a THD of 10%. In this case, the distortion components are 20 dB below the signal level and this will swamp the AEC. A THD of 3% will generate distortion 30 dB down and this will still impact the AEC. We recommend a THD less than 2% so as not to impact an AEC with an ERLE of 30 dB.

It is important to measure the THD of the *entire system*, incorporating the speaker *and* the microphones. Simply measuring the acoustic output of the speaker is insufficient because the plastic enclosures used for many voice UI products can conduct vibrations directly from the speaker to the microphones. Consider the graph shown in **Figure 7** on the next page. This graph shows the THD of a loudspeaker measured with an external reference microphone. Each line represents a different playback level. For each playback level, we record the measured SPL and also the THD at numerous frequencies across the entire audio spectrum. The circular bubbles on the graph indicate the measured THD appear only at levels where THD is above 3%. The speaker behaves linearly, and distorts to a significant degree only when played at loud levels.

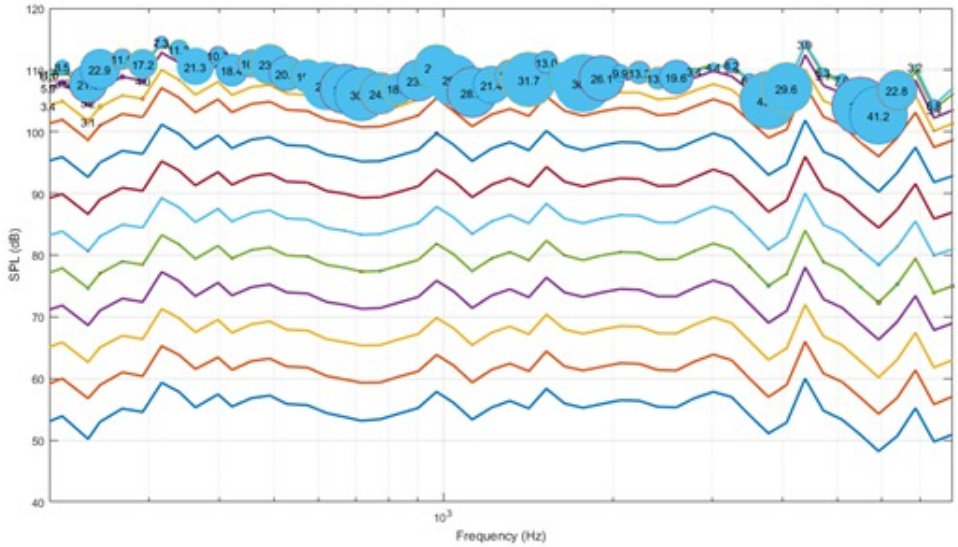


Figure 7: ***Loudspeaker distortion measurement using an external microphone. The speaker is linear, and distorts only at high SPL.***

This measurement is now repeated using the onboard voice pickup microphones located at the top of the plastic enclosure of a typical “smart speaker” with voice UI. In this case, there is a resonance of the enclosure which couples strongly (and nonlinearly) with the microphones in the 500 to 800 Hz range, as shown in **Figure 8** below. This is unacceptable; the enclosure must be redesigned for added stiffness and better acoustic isolation.

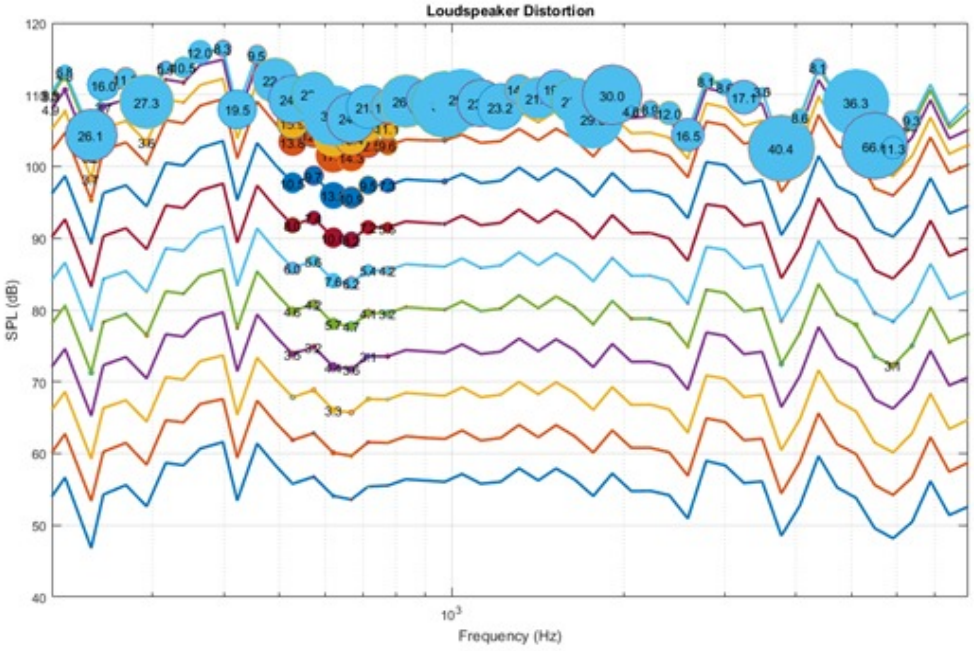


Figure 8: ***The same loudspeaker with the distortion measured by the microphones in the product itself. In this case, there is conducted sound which causes distortion in the range of 500 to 800 Hz.***

Beamforming

The reason multiple-microphone arrays are commonly used in voice UI systems is that having more than one mic allows the array to become directional—to focus on sounds coming from a particular direction. This process is called beamforming. It improves SNR because it helps isolate the user's voice while rejecting sounds from other directions.

For example, if the user is on one side of the microphone array and an air conditioner is on the other side, the sound from the air conditioner arrives first at the microphone opposite the user, then arrives a fraction of a second later at the microphone closest to the user. The beamformer algorithm uses these time differences to null out the air conditioner sound while preserving the user's voice.

The more microphones in an array, the more effective beamforming can be. An array with two microphones has a limited ability to cancel sounds, but an array with multiple microphones can cancel sounds coming from more directions. The fewer microphones, the more the performance will vary as the look angle—the angle between the user's voice and the front axis of the voice UI product—changes.

A beamforming algorithm can optimize SNR by dynamically adjusting its performance to suit the conditions. The beam width can be tightened to focus better on the user's voice and more effectively reject sounds from other directions, but the voice UI system will then need to evaluate and adjust DOA more frequently to make sure the beam stays focused on the user. This effort increases demands on the system, so most beamformers maintain a fairly wide beam. For example, a typical seven-microphone array has a beamwidth of approximately 60 degrees, or ±30 degrees relative to DOA.

The spectrograms in **Figure 9** below demonstrate the ability of beamforming to remove background noise. The top figure shows the spectrogram of a single microphone. The bottom figure is the output from a seven-microphone beamformer. The horizontal stripes are the harmonics associated with the speech signal and the background orange/red color is babble noise. The ideal result would be clearly defined stripes surrounded by dark regions. In the measurement from the beamformer, the speech is preserved and the background noise is attenuated by 6 to 7 dB. This provides a noticeable improvement in speech recognition.

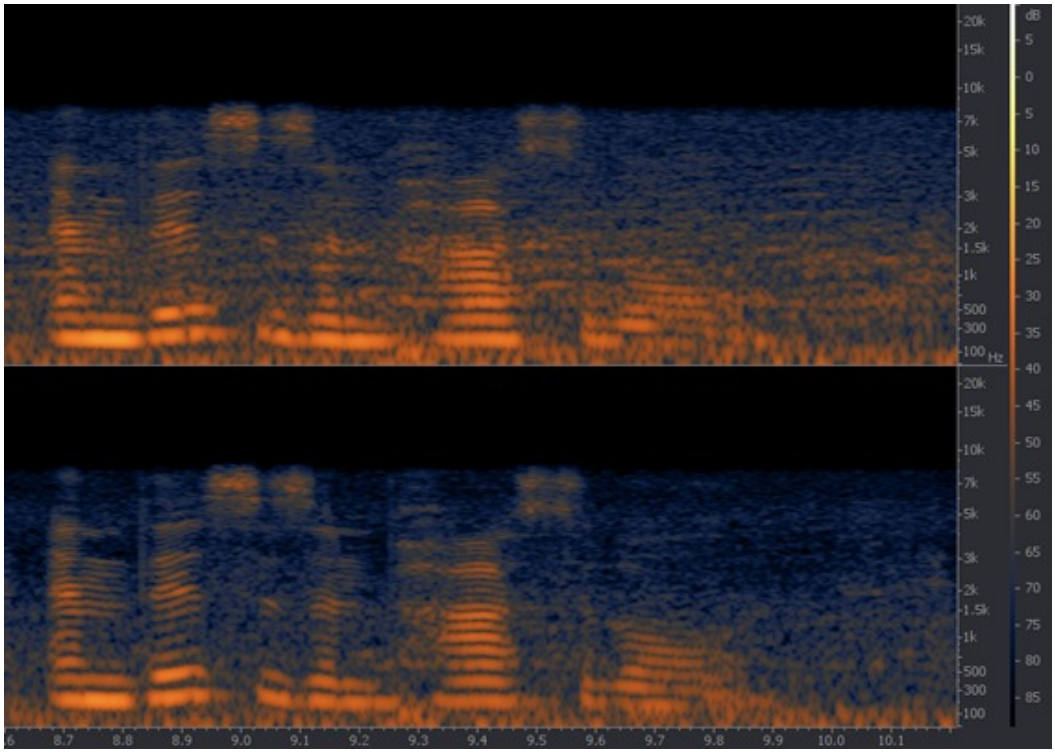


***Figure 9:** Spectrogram showing the reduction of background noise that can be achieved by the beamformer. Dark sections correspond to lower signal levels. The original speech is untouched.*

Noise Reduction

Although microphone array systems use directional pickup patterns to filter out unwanted sounds (i.e., noise), some of these unwanted sounds can be attenuated or eliminated with an algorithm that recognizes the characteristics that separate them from the desired signal and then removes the unwanted sounds, much as someone who dislikes lemon flavor might ignore the yellow candies in a bowl. A noise reduction algorithm can run on a single microphone or an array, so it can assist with trigger word recognition and also improve voice UI performance after all the other algorithms have done their jobs. Thus, noise reduction might be used in multiple stages of a voice UI signal processing chain.

Voice commands are momentary, as opposed to steady-state, events. Any sound that is always present, or that is repetitive, can be detected and removed from the signal coming from the microphone array. Examples include road noise in automobiles, and dishwasher and HVAC system noise in homes. Sounds that are above or below the frequency spectrum of the human voice can also be filtered out of the signal.

Noise reduction algorithms have been commonly used for many years, but most are optimized for cellphone applications rather than voice UI. They tend to highlight the frequency spectrum most critical for human comprehension, rather than the frequency spectrum most critical for an electronic system to isolate and understand voice commands. Most noise reduction algorithms that are tuned for cellphones actually degrade voice UI performance. To put it simply, humans listen for different things than voice UI systems do.

One measure of how well a noise reduction algorithm works is to see how many additional dB of signal reduction it provides at the output of the echo canceler. **Figure 10** below shows the performance of DSP Concepts frequency domain based noise reduction algorithm, reducing residual echoes by up to 12 dB.

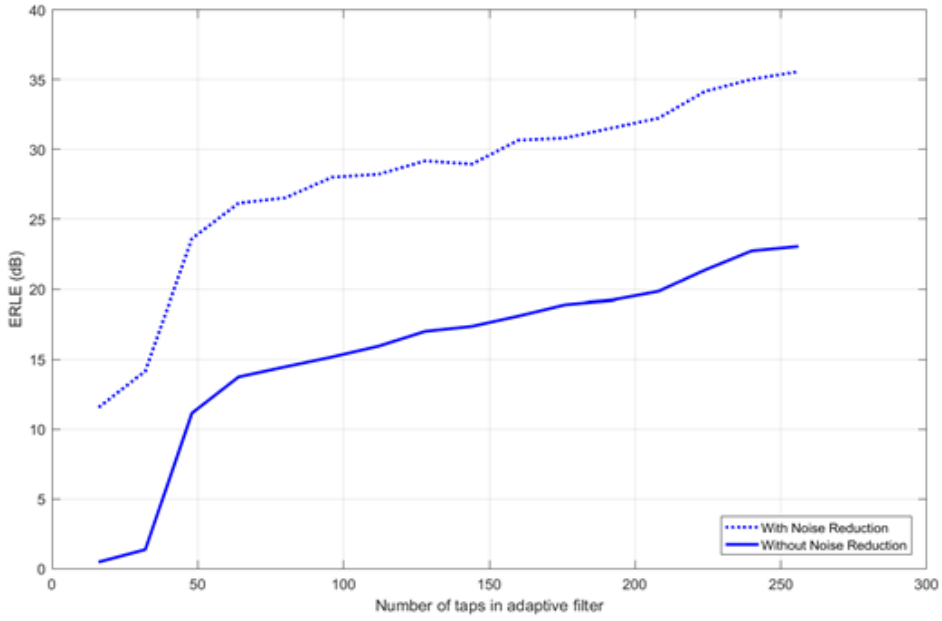


***Figure 10:* Effects of a noise reduction algorithm on ERL. The higher the curve, the more attenuation and thus the better the algorithm performs.**

The subjective improvement in sound quality is instantly recognized, but will it improve the performance of the speech recognition algorithm? This requires additional measurements to quantify. **Figure 11** below reproduces ones of the curves from Figure 2 but does it with and without noise reduction. The noise reduction shifts the curve 2 dB to the left as compared to the original content. This shows that the noise reduction algorithm improves overall speech recognition by 2 dB.

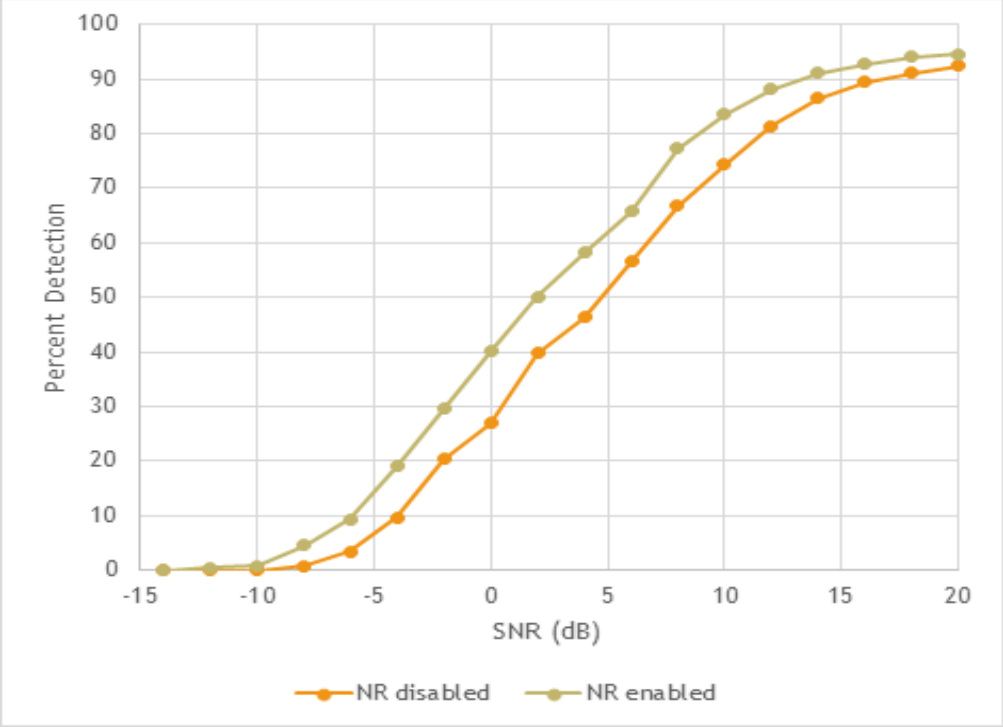


***Figure 11:* Effects of a noise reduction algorithm on ERL. The higher the curve, the more attenuation and thus the better the algorithm performs.**

In the next paper …

This concludes our discussion of the fundamentals of voice UI systems. In our next paper, “Optimizing Performance of Mic Arrays and Voice UI Systems,” we will examine the effects of different microphone array configurations and different microphone choices. After examining these effects, we will make specific recommendations that engineers and product design teams can employ to get the most reliable performance from their voice UI product designs.