Implementing Maximum-Likelihood Beamforming on the SRI Speech Recognizer

Marc Ferràs Font

International Computer Science Institute, September 2005

ferras@icsi.berkeley.edu

Table of Contents

List of Tables

List of Figures

Abstract

Likelihood Maximization Beamforming (LIMABEAM) is a microphone array processing technique specially focused on speech recognition. Unlike other signal processing techniques which clearly assume the beamformer and recognition system to be cascaded, LIMABEAM uses a closed-loop approach for beamformer optimization. This allows the parameters of the beamformer to be chosen so that the likelihood of the state sequence is maximized and, thus, classification improved. This is achieved by means of an adaptive procedure for each input utterance to be recognized.

This report mainly focuses on a deeper understanding of LIMABEAM as well as implementation related information. This is the result of reading Michael L. Seltzer's PhD thesis and informal e-mail exchange between him and the author.

Chapter 1

Introduction

Most of the current microphone array processing techniques are thought of as an optimization problem. Typically, since the optimization procedure is performed at the signal level, the related objective functions are also derived in this domain. This fact may lead to processed waveforms which are not perceived by humans as enhanced, since a mathematically feasible cost function to optimize might not be relevant to us listeners. It is also not known what should be a relevant cost function, as it is not clearly understood yet the way humans perceive speech and audio signals.

In speech recognition we are dealing with machines, instead of human beings. This makes the process of automatic speech recognition a bit more understandable, since the system itself has been designed by humans. In practice, though, a complete understanding of the interaction among the modules in the speech recognizer becomes unattainable. Thereby, speech recognition systems typically assume a cascade approach. Here, speech information goes through a series of modules[1] till the systems outputs the corresponding hypotheses for the correct transcriptions.

LIMABEAM breaks the cascade hypothesis between the spatial processing and the recognition modules. This way, information from the recognition engine is fed

[1]Preprocessing, feature extraction and classification.

back for the the spatial processor to take advantage of it. Particularly, the speech models contain valuable information for classification which is, in the end, the goal of the recognizer.

Another important advantage of LIMABEAM is that no assumptions about interfering signals, microphone lay-out, speaker-to-receiver impulse responses, etc... are made. It is mainly a data-driven approach.

Chapter 2

Likelihood Maximization Beamforming (LIMABEAM)

As typically used by array signal processing techniques, LIMABEAM uses the filter-and-sum approach for beamforming. This means that the enhancement process itself consists in filtering the speech signals prior to being summed up, i.e.,

$$y(n) = \sum_{m=0}^{M-1} \sum_{p=0}^{P-1} h_m(p) x_m(n - p - \tau_m) \tag{2.0.1}$$

where $y(n)$ is the output signal, $h_m(n)$ is the filter associated with channel m. Note that $x_m(n)$ is delay-compensated by τ_m.

LIMABEAM tries to optimize the filter taps, $h_m(n)$, based on the likelihood calculation for every input utterance. Figure 2.1 shows how beamforming and the recognizer are integrated into the whole system.

Since the optimization procedure will be performed in an adaptive manner, convergence properties are a major concern. Using a time-domain filtering approach such as (2.0.1) can result in poor convergence properties due to, first, the large number of parameters to be optimized and, second, the correlation properties between these

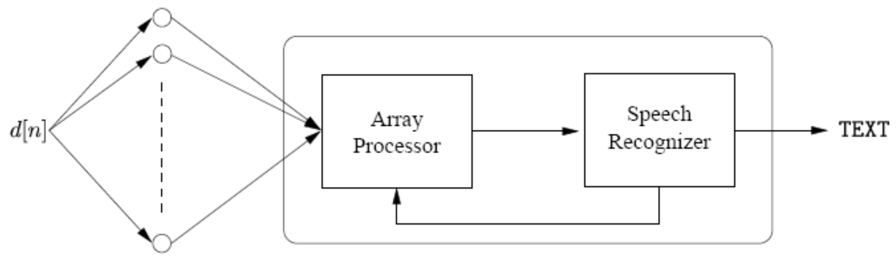

Figure 2.1: Closed-loop interaction between LIMABEAM and the speech recognizer (Taken from [1] page 4).

parameters as the filter is being adapted. Particularly, in noisy plus reverberant environments this might become a true downfall and might result in no convergence at all. As it is well-known in the adaptive filtering field, sub-band approaches can perform significantly better in these environments. Here, the input signal is analysed by a set of filterbanks and an adaptive filter for each band is used, as shown in Figure 2.2. Note that both the input, $x(n)$ and reference, $d(n)$, signals must be analyzed by the filterbank.

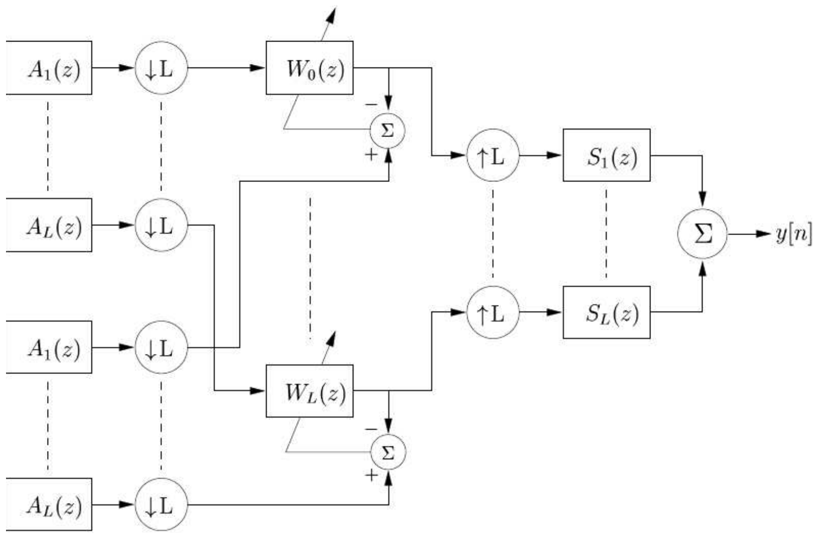

Figure 2.2: A typical sub-band adaptive filtering framework (Taken from [1] page 77).

There are two main advantages of using a sub-band approach. First, the filterbank outputs are aimed and assumed to be independent (there might be some leakage due to the use of non-perfect filters). This means that the optimization can be performed on each of the bands simultaneously. This way, there is no interaction between optimized parameters as the optimization goes on. The second advantage is that, if enough bands are assumed in the filterbank, each of the filterbank outputs can be assumed to be narrowband. This can be specially beneficial since the noise, for instance, can be considered as nearly white within that bandwidth. These are the typical conditions where adaptive filtering performs best.

An efficient way to decompose the input signal into sub-bands is to use a DFT filterbank for every input frame. Thus, the time-domain taps to be optimized become complex-valued coefficients in the spectral domain. Using a 256-point DFT is equivalent to optimizing a 256-tap filter in the time-domain. For this DFT approach[1],

$$Y(k) = \sum_{m=0}^{M-1} H_m^*(k) X_m(k) \tag{2.0.2}$$

To resynthesize the enhanced speech signal, the processed spectrum, $Y(k)$, should be inverse Fourier transformed by means of an IDFT and an overlap-and-add technique should be used to smooth the time-domain output. Nonetheless, since the next step for the system will probably be to extract features from that processed spectrum, time-domain resynthesis could be skipped for most of the types of features we'll be interested in. It will depend on how easy it is to interact with the speech recognizer. For the SRI speech recognizer we are using at ICSI, I would think the easiest is to resynthesize the time-domain output and let the SRI front-end be the one to extract features, as it is more transparent and probably safer.

Although Seltzer himself proposed, in informal e-mail exchange (see Section A.1), to go directly for the sub-band approach (S-LIMABEAM), there are relevant issues

[1]Note that here only one frame span filters are assumed, as opposed to the multi-frame approach in [1] page 79

regarding it, as he already pointed out in his dissertation. The independence assumption between filterbank outputs turns more tricky, now. Assuming MFCC features,

$$\mathbf{f} = \mathbf{D}^{-1} \log \mathbf{Mx} \tag{2.0.3}$$

, where $\mathbf{f}$ is the extracted feature for that particular frame, $\mathbf{D}^{-1}$ is the inverse Discrete Cosine Transform transformation matrix, $\mathbf{M}$ is the mel-filterbank transformation matrix and $\mathbf{x}$ is the energy spectrum of the input frame, that is $X(k) = |Y(k)|^2$, each feature component in $\mathbf{f}$ is tied to ALL of the input frequency bins in $\mathbf{x}$ through $\mathbf{D}^{-1}$, but also partially through $\mathbf{M}$. Although using diagonal covariance matrices in the HMM models leads to an independence assumption for the feature components, this independence cannot be ported down to the parameters to be optimized. Therefore, optimizing a certain filterbank spectral coefficient will affect other components from other channels as well.

Appendix A

Further Information

A.1 E-mail exchange

Here you will find informal e-mail exchange between Michael Seltzer and the author, in order to possibly clarify some of the aspects of LIMABEAM. Michael volunteered to provide source code for LIMABEAM, although it is likely that it cannot be directly applied to the SRI speech recognizer. Nonetheless, it can be a guide to our implementation.

```
Hi Marc,

Thanks for your mail. I'm glad you like my thesis. Good to know
that someone is reading it after all that effort :)

Anyway, those are all good questions. I've tried to answer them
inline below. Definitely feel free to ask me any other questions.
Also, I'd need to check with my advisor, but it's possible that I
could share the code. You wouldn't be able to use it directly
since it uses several libraries you probably don't want to
integrate (like the CMU Sphinx III recognizer and my own personal
libraries I used in grad school). But, the main architecture might
```

be useful. I’ve got a really full plate right now, so I probably wouldn’t be able to package it up until after Eurospeech.

Good luck!

Please give my regards to Morgan and the rest of the ICSI folks!

Best, Mike

-----Original Message-----
From: Marc Ferras [mailto:ferras@ICSI.Berkeley.EDU] Sent: Tuesday, August 23, 2005 5:11 PM To: Mike Seltzer Subject: Comments about LIMABEAM...

Hello Michael,

I’m a visiting researcher at ICSI, working on multi-microphone techniques for ASR. We are currently considering implementing your LIMABEAM technique on our speech recognition system aimed at improving WER on the meetings task.

I’ve been reading your thesis which, by the way, i sincerely think it is really good piece of work. Since our aim is meetings speech recognition, which typically contain only moderate reverberation, my focus would be implementing the time-domain version, in an unsupervised way. I have, though, several doubts about it, and maybe you could give me your impressions about them:

<<MIKE>> I would actually recommend using the S-LIMABEAM method with 1-tap per subband. It was always as good as the LIMABEAM

method, you don't need to worry about how many taps to you (20 vs. 30 vs. 40, etc - 1 tap is basically all of those things in one!), and the convergence is significantly faster. I actually did some experiments in the thesis with S-LIMABEAM with 1 tap on the ICSI meeting, which I'm sure you've seen.

1. You are using a conjugate gradient descent approach for the optimization procedure. Have you tried using LMS gardient descent, instead? Going a bit further, is it very feasible to fall into local minimums? Is that the reason why you used conjugate gradient, or it is convergence speed, for the most part of it?

<<MIKE>> I used CG, mostly b/c I didn't want to deal with step-size and all that jazz. CG figures all that out on the fly using approximations to the Hessian. I later found that another quasi-Newton method, called BFGS optimization (also in NRinC) is quite a bit faster, at a very small performance hit. Here are MSR, other folks who do similar optimization problems have used these methods, and then later settled on Stochastic Gradient Descent. This method has actually worked the best for them. It requires some tricks to get it to work though. I believe, for example, that your input features have to be normalized to mean 0 and variance 1. Anyway, if you wanted I could look into this.

2. I see you used a parallel HMM to ease optimization on the log mel spectra. I guess you eventually came up with this option after trying it on cepstrum directly. Based on this, did you experience serious problems optimizing over cepstrum features? Here, I mean whether convergence was able to be achieved or not, and if that was the reason to use two parallel HMM models.

<<MIKE>> Yes, I was never able to get cepstrum optimization to work directly, or else I would have dumped the log spectral models. I had a hand-wavy explanation for why ceptrum wouldn't work but I was never really satisfied with it. Some folks at Univ of Karlsruhe in Germany (Raab and McDonough) did some work based on S-LIMABEAM, and interestingly, they got cepstrum to work but not log spectrum. They used a different mic array architecture so the results weren't comparable.

3. The last concern is regarding computational cost and practical implementation issues. I've seen that your implementation is x10 times real-time for 1 channel and upper for several channels. Did you implement LIMABEAM as part of the speech recognition system source code to get those times? Or was it a cascade of independent binaries? That's just to figure how it could be in my case.

<<MIKE>> No, it was a cascade of independent binaries. The time did not include the recognition time. I used libraries from the recognizer to get the functions for HMM model I/O and state sequence I/O, but other than that, it was standalone code. Like I said above, S-LIMABEAM is faster than LIMABEAM. Also, one thing that would really speed things up I think is that for "academic" reasons, I started every utterance from a reset D&S initialization. However, if your utterances are in a sequence in time (like in a meeting) then you could initialize your current utterance with the converged parameters from the previous utterance. Also, if you go with S-LIMABEAM, you can try some of the parameter sharing techniques I proposed. Finally, for all of this, I recommend the FFTW library ("Fastest FFT in the West"). It's a free library that's downloadable on the web, and does very fast FFTs. Since this algorithm is definitely FFT-intensive, it

was a big time savings.

Well, i would really appreciate your comments on this. If you have any further comments which you think are relevant, I'd appreciate them as well.

<<MIKE>> Two things: 1) I did time-alignment of the channels before hand. For normal array configurations (not the 4 PZM mics), this was shown not to be really necessary (see the LIMABEAM paper in Sep 2004 issue of IEEE TSAP). 2) For unsupervised processing, it was fairly critical to have "enough" speech data in each utterance. My thesis shows that (somewhere there is a plot of performance versus utterance length). And of course, you don't want this to be dominated by silence. Otherwise, since the optimization is unconstrained, it will send everything to zero to match the target silence. So it the utterance is 10 sec but only 1 sec is speech and 9 sec is silence, this is not good.

Thanks a lot in advance!

--Marc

Bibliography

[1] M. L. Seltzer, *Microphone array processing for robust speech recognition*, Ph.D. thesis, Department of Electrical and Computer Engineering, 2003.