Electronics and Computer Engineering Dept.
Telecommunications Division
Technical University of Crete

INSTANTANEOUS STREAM WEIGHT COMPUTATION IN AUDIO-VISUAL SPEECH RECOGNITION

SPIROS DIMOPOULOS

Supervisor: Alexandros Potamianos

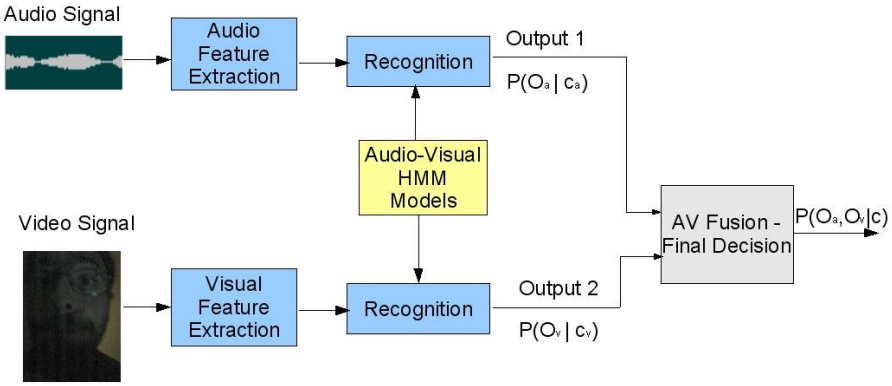
Committee member: Vasilios Digalakis

Committee member: Michail Zervakis

Chania, October 2007

INSTANTANEOUS STREAM WEIGHT COMPUTATION IN AUDIO-VISUAL SPEECH RECOGNITION

October 2007



Technical University of Crete
Department of Electronics and Computer Engineering
Telecommunications Laboratory
University Campus - Kounoupidiana
731 00 Chania
Hellas (Greece)

SUPERVISORY COMMITTEE
Supervisor: Alexandros Potamianos
Committee member: Vasilios Digalakis
Committee member: Michail Zervakis

ACKNOWLEDGEMENTS

First I would like to thank my professor Alexandros Potamianos. His help was really invaluable during the project. He also helped me organize my work. Also Eduardo Sanchez Soto gave me a really helpful introduction into the subject and subsequent help.

More personally I would like to thank my family for their moral support during all the years I was undergrad. I would like to thank and greet all my friends for the great times we had during these years. Also I would like to thank sweet Efi.

ABSTRACT

Different novel methods are tested in the ASR domain to upgrade the speech into a useful UI under all conditions. For robust ASR there is still way ahead until the speech recognition accuracy reaches adequate levels, for speech to be practically used in real world conditions. Audio-Visual ASR is based on the concept of bimodal speech production and perception from humans. We use optical and acoustic information to recognize the spoken word. The theory of multiple information stream decision fusion is used for the combination of audio and video streams. So the acoustic information decision is complemented with the visual information and a final recognition decision is made.

In this project we researched different AV-ASR techniques and especially a weighted fusion method that is based on computation of instantaneous stream weights. The reliabilities of the streams are computed from the single stream classification/recognition results. Then instantaneous weights are computed for the fusion of the streams in the time frame level. The results of the different AV-ASR techniques are compared. For this purpose we set up an AV-ASR recognition system simulating different environmental noise conditions. Some improvement in recognition results is confirmed when using AV recognition compared to the Audio-only case and also when using time-variant stream weights in contrast to static weights in AV fusion.

CONTENTS

List of Figures

List of Tables

Introduction

One of the main features of future human-computer interaction interfaces (HCI) is speech [1]. This will make the interaction more natural to the human user. In order to achieve this, research in Automatic Speech Recognition (ASR) is being made since the '50s [2]. In spite of the progress made in specific applications like dictation and medium vocabulary transaction processing tasks under controlled environments, when it comes to real world applications in unrestricted listening environments the performance deteriorates below the threshold of usefulness [3]. A major problem of ASR is robustness under channel and environmental noise. Many techniques have been used to improve the recognition under these noisy conditions, including mainly enhancement of the audio signal, applying noise resistant parameterization, and identifying speech in those sub-bands of the spectrum that the speech signal is dominant. In recent years, nontraditional approaches that use sources of information orthogonal to the audio signal are tested. One approach that has become a mainstream research area is supplementing the audio with the visual signal. The latter is not affected by the audio noise and can improve the final result of the recognition. [1],[3]

Audio-Visual ASR (AV-ASR) is based on the concept that human speech perception is benefited by visual stimuli. When people listen to a speaker they usually, consciously or not, see his facial expressions and do lipreading, especially under noisy conditions, to better understand what the speaker says. The main method used in AV-ASR,

for the analysis of visual information, is exploiting the mouth region of the speaker by analyzing 2-D images of that region and do the visual feature extraction. The audio and the visual information streams must be fused at some point of the recognition process in order to get some combined results. They can be fused either at the feature level or at the decision level. If they are fused at the decision level an information stream weighting scheme can be applied that can model the varying information stream reliability at different points in time. To compute the stream weights for the weighting scheme different statistical methods can be used. The weights can be precomputed and be fixed during the whole recognition process or can be dynamically computed and updated at the utterance, word, phoneme or time frame level. Also a possible asynchrony between the streams can be modeled using appropriate modeling when the fusion is at the decision level. [1]

The AV-ASR research community has developed many different data sets of Audio-Visual speech in many languages. But the need for a common database of Audio-Visual speech data became apparent when the approaches of the many research groups needed to be compared. The results of the tests should be extracted from similar data sets in order to be comparable. This need led to the development of the Clemson University Audio-Visual Experiments (CUAVE) database which is is a speaker-independent corpus of both connected and continuous digit strings of high quality video and audio of a representative group of speakers. [4]

In this Diploma Thesis Project we worked with the CUAVE database to get comparable results. The main recognition system was developed with the help of HTK [5], a tool to create recognition systems based on Hidden Markov Models (HMM). Some modifications were made to the original HTK code, to support the instant temporal change of the stream weights in multi-stream

recognition and also to output the instantaneous probability results in every time frame. The signal processing tasks, such as the separation of the Audio and Video signals from the original data, the pre-processing, the addition of noise to the audio signal and the analysis of the recognition results was done with the help of Matlab. Different AV-ASR methods were tested and compared to the standard Audio ASR to see if there was some improvement. The main method researched was the AV-ASR with time-varying stream weights computed from one or the combination of stream reliability indicators.

The organization of this thesis follows the next plan. It contains 6 chapters and 2 appendixes.

In *Chapter 1* a general introduction in Robust Speech Recognition is made, identifying the main problems of ASR under noisy conditions and proposing two solutions based on multi-stream information analysis. One solutions is the AV-ASR concept which is researched in this Project.

In *Chapter 2* different information fusion methods are described. Based on the re-evaluation of the stream weights the methods are separated on static or dynamic methods. Also the asynchronous methods are described.

In *Chapter 3* different methods are introduced for the evaluation of the stream weights. Two methods based on stream reliability indicators and the mapping to weights are described. Also a method of direct compuation based on K-means clustering is introduced.

In *Chapter 4* the detailed description of the recognition system that we have set up is given. All the components and the processes used are described.

In *Chapter 5* the results of the tests are presented for the different methods used. They are compared to each other and some preliminary conclusions are drawn.

In *Chapter 6* the final conclusions of our research work are made. Also some planning for future work on this subject is presented.

In *Appendix A* the graphs of the stream weight for 4 speakers and all utterances of the digits are shown for comparison.

Finally in *Appendix B* the changes we made to the original HTK code are presented.

Robust Speech Recognition CHAPTER 1

1.1 Speech Recognition under adverse conditions

Many improvements have been made in recent years, leading the way to better accuracy of results in speech recognition tasks. At present the results are almost perfectly accurate in specific tasks that use limited vocabulary of spoken words and at controlled environments where noise does not degrade the acoustic signal of speech. The real challenge now for ASR researchers is to make speech a competitive user interface comparable to that of keyboard, mouse and display under real world environments. These environments are often challenging for the speech recognition process. For example in an office there is noise caused by other people talking, by machines such as faxes and printers and phones ringing. To make things worse the speaker does not have unlimited time to speak slowly and clearly and cannot keep his temper if he has to repeat the same commands twice or more. The previous description of the office environment is a typical real world scenario where ASR still fails to achieve adequate accuracy.

Novel methods are tested to increase the accuracy of the final recognition result. One category exploits the combination of separate recognition results on the same data but at different either spectral bands or modalities (in case of AV data) to get a total recognition result that surpasses the original. The first method is based on the

spectral separatability of the audio and noise signals. When noise is added to original speech signal, the audio signal is degraded as a whole. But different types of noise exist according to their spectral characteristics. Not all parts of the original speech signal spectrum are degraded in case the noise is limited to a spectral region. Separating the original speech signal in multiple spectral regions, doing separate recognition in that parts and finally integrating these results to get the total result can yield some improvement on the whole [7]. The second method is exploiting the immunity of the video signal to acoustic noise. When speech recognition is applicable to Audio-Visual data, we can process the two information streams (audio stream, video stream) as separate and do the recognition by combining the information from both streams. This gives a standard advantage to the system in case of an audio noise scenario as compared to the use of only the audio signal information. The two methods are described in detail in the next two paragraphs.

1.2 Speech Recognition Using Multiple Information Streams

In general the speech recognition process uses the audio signal to extract the features and do the training and testing. The audio signal can be considered as an information stream that it is used to extract the appropriate information of the spoken word. In figure 1 we can see the typical recognition process flow. We assume that the models are

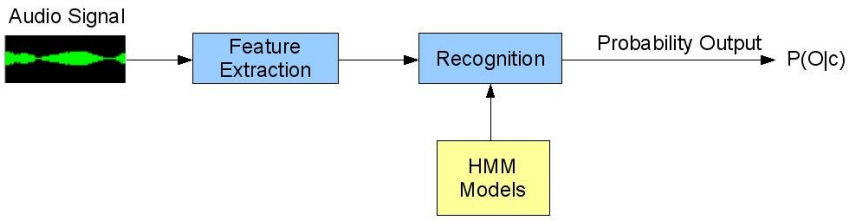


Figure 1*: Typical ASR System*

already trained. This system uses one information stream, the audio signal taken at the whole spectrum.

In [7], a new recognition system is proposed using multiple information streams from the audio signal by separating the audio signal spectrum in sub-bands and using each sub-band as a separate information stream. There are some reasons for doing this analysis. The first is some experimental results suggesting that maybe the human decoding of the acoustic information of speech is based on decisions made within narrow frequency sub-bands of the audio signal. The decisions are taken independently from each other and are combined at some level in time to make the global decision in a way that the global error rate is equal to the product of the error rates in the sub-bands. The other reasons are more practical and can be summarized in the following:

- The noise may have degraded the original speech audio signal only in some specific sub-bands. So making the decisions on sub-bands and then combining the independent decisions can improve the final result, as long as there are some sub-bands that have enough information to support the right decision.
- Some sub-bands may be better for recognizing some speech classes
- There is asynchrony in the transitions between stationary segments of speech at different frequency bands, and this method can help relax the synchrony constraint in typical HMM systems.
- Different recognition strategies can be applied in different sub-bands.

The process is described graphically in figure 2. First the number of sub-bands and their position in spectrum must be chosen. Then a ASR system is applied in each sub-band taking each separated signal

as an independent information stream. A synchronization of the streams in the time frame level is assumed. This makes possible the use of state-synchronous multi-stream HMMs. These HMMs have different emission probability parameters for each stream and combined transitions probabilities between the common states. In figure 3 the differences of the simple 1-stream HMM with the 2-stream HMM are explained. The 2-stream logic can be extended to n-stream HMMs by simply adding emission probability vectors for each new stream in every state. Then the probabilities are combined in a final decision step to take the final score for each class. The recombination step and the different methods that can be applied are described in Chapter 2.

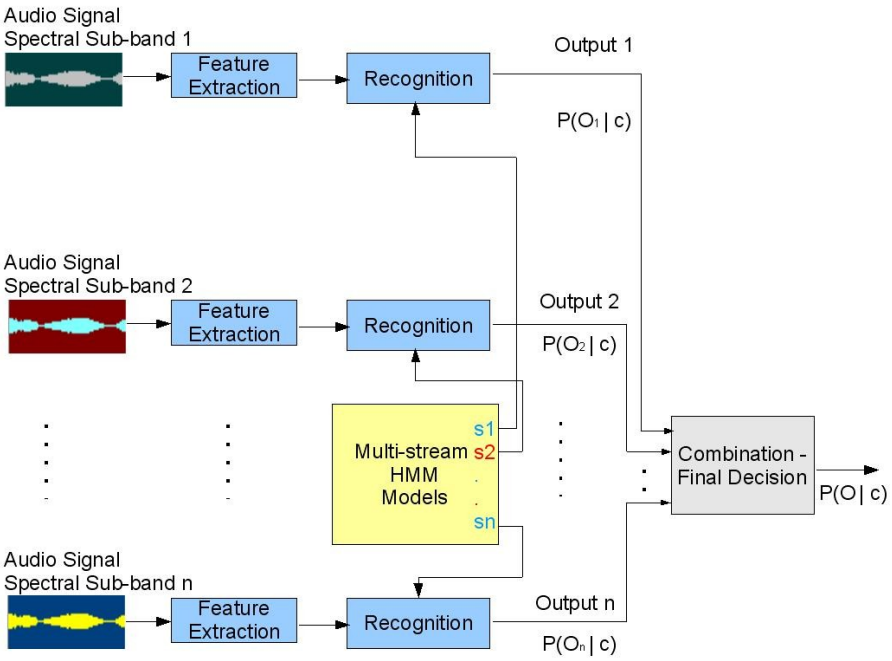


Figure 2*: ASR System using multiple information streams taken from diferrent spectral sub-bands of the original signal*

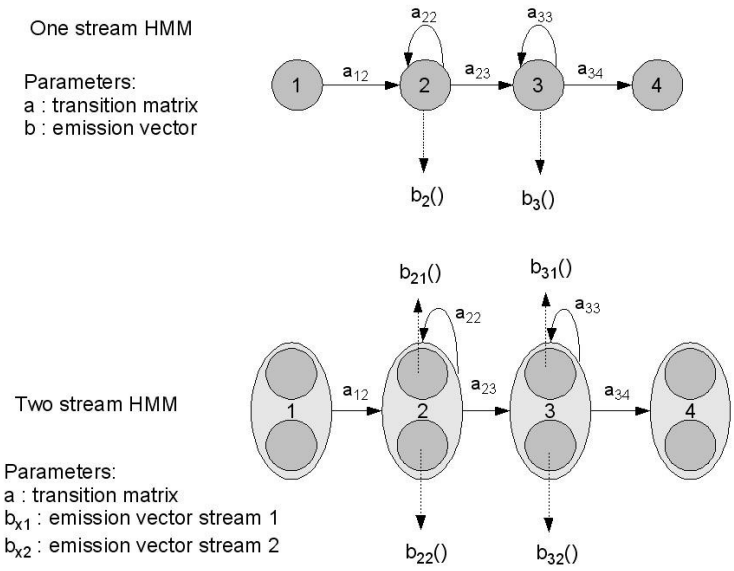


Figure 3*: One stream vs Two stream HMM properties*

1.3 Audio-Visual Speech Recognition

The Audio-Visual ASR approach can be considered as a multiple information stream recognition process consisting of two streams, the Audio and the Visual. The idea is based on the concept of bimodal nature of human speech production and perception [1] [3] [6]. The two modalities are the acoustic and the visual which both carry linguistic and para-linguistic information.

During the production of speech, several visual clues are visible that complement the acoustic signal. Speech segmental information are included in the visual modality. Also information about the place of articulation are present, because the tongue, teeth and lips are visible. During the perception of speech, bimodal integration has been presented by superimposing the sound /ga/ on a video of a /ba/ then most people perceive the sound as a /da/. Also speechreading, reading the lips of a speaker to better understand the spoken word, is usually used by a person with impaired hearing or generally when listening conditions are harsh.

The previous facts makes the research of a bimodal speech recognition system quite interesting especially under acoustic noise, as the visual modality is unaffected.

Many variations of an AV-ASR system can be identified. The main components are shown in figure 4, which is a variation of the multi-stream ASR system shown previously but with two streams – Audio and Video. This setup uses separate feature extraction of the two streams and recognition using 2-stream AV HMM models. The training of the AV Models is depicted in paragraph 1.3.2. Then combines the results of the two stream recognition in order to take a final decision; different methods can be used for the fusion described in Chapter 2. Another option is to fuse the modalities exactly after the

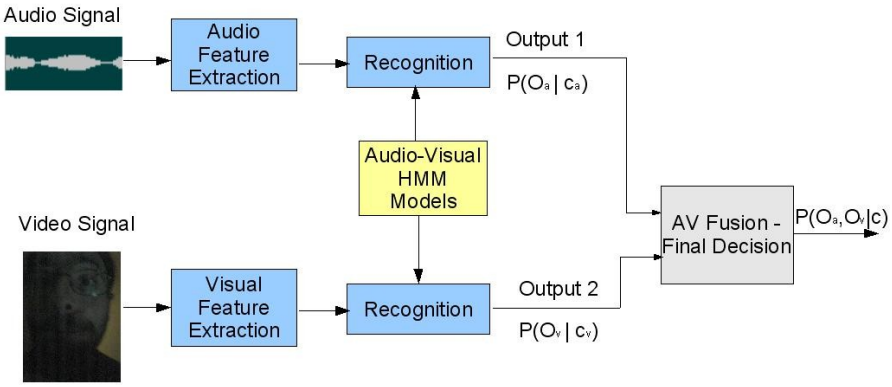


Figure 4*: The typical AV-ASR system with decision fusion*

feature extraction task by concatenating the feature vectors into one AV vector and run the recognition process as if it was one stream ASR. This approach is not tested in this project.

1.3.1 Feature extraction

The feature extraction process is done for both streams.

For the audio stream:

Initially we have the raw audio signal with the speech. We have to convert it to feature vector sequences. Mel Frequency Cepstral Coefficients (MFCC) are usually used as features because of their desirable characteristics. They are computed from the log filterbank amplitudes - that are taken from filterbank analysis - using the following formula:

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j-0.5)\right) \quad (1)$$

with N the number of filterbank channels. Also the first and second time derivatives of the original MFCCs can be used as features. In our setup we used 13 MFCC features (12 static + energy) plus their first and second derivatives, totaling 39 audio features in each time frame. The time frame was set to last 10msec so we have 100 feature vectors per second. The extraction was made using HTK tools [5].

For the visual stream:

The visual signal consists of speaker faces. The visual information of speech have to be identified and extracted. One technique is to use the mouth region of the speaker just as it is used during speechreading. When this region is identified as ROI, a number of 2-D DCT features are computed and saved.

The main problem of this analysis, which we used in our setup, is to keep track of the mouth region in sequential images. Initially the ROI must be explicitly set in the first image of the video sequence. We used a 110x60 pixels sized region as the ROI. Then to locate the ROI in the next image, we set a +20 pixels region in all 4 directions as shown in Figure 5 and in this we searched for a 110x60 pixels region

that correlates best with the ROI of the previous image. Normalized 2-D cross-correlation is used from the Matlab Image Processing Toolbox.

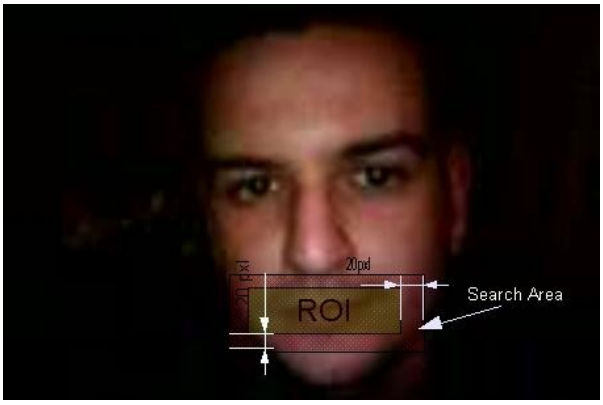


Figure 5*: ROI and the extended search area*

Next we extract the visual features from the ROI. We follow the steps:

1. Convert the image from RGB to B&W
2. Decimate the image to 16x16 matrix
3. Compute the 2-D DCT of the ROI
4. Keep the 6x6 DCT coefficients excluding the energy

After these steps we have 35 visual features for every video frame. Finally we up-sample the visual features which are extracted at 30 frames per second to 100 frames per second to match with the audio feature vectors frequency. The above process is illustrated in Figure 6.

1.3.2 HMM Training

The training process of the HMM parameters is done before any attempt to run recognition tests. One HMM must be created for every speech class unit that will be recognized, be it either phoneme or word. Some training data must be assigned from the data set with known labels, run the training algorithms with this data iteratively until some desired convergence is achieved.

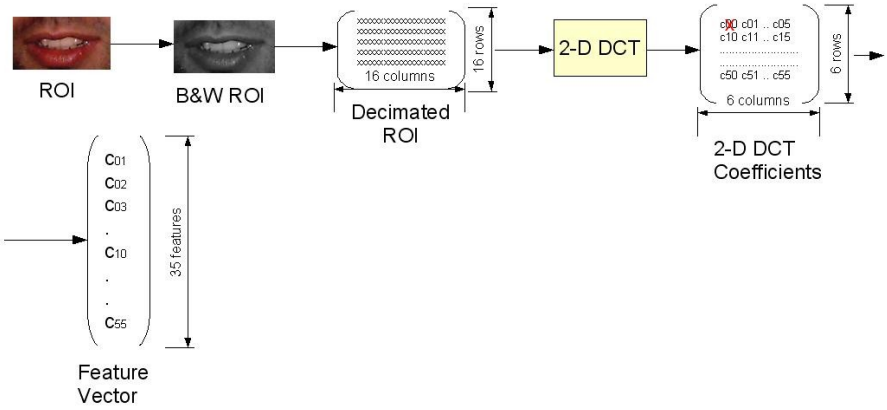


Figure 6: Visual feature extraction process from ROI

The training of the simple 1-stream HMM is done using the Baum-Welch re-estimation formula. The parameters that need training are the transition probability matrix a_{ij} between the model states and the emission probability vector b_j for every state.

In the case of the 2-stream Audio-Visual HMM model the parameters are again the transition probability matrix a_{ij} between the model states but now there are two emission probability vectors, one for every stream, ba_j and bv_j. There are two approaches in the training of the 2-stream HMM. In separate training of the parameters for every stream, the parameters are estimated separately and then the transition matrix is either set to the audio one, or to the product of the transition probabilities of the two HMMs. The other method is to jointly estimate the parameters in order to enforce state synchrony in training [1]. We used the second method in our setup, which is the method used by HTK [5] when training multi-stream HMMs.

Two parameters that need also to be set are the stream weight parameters. These define the weighting that would be applied to the stream intermediate results in order to get the final output score. These parameters must be set during the training process. The state-dependent AV emission depends on the fusion method used and is

better described in chapter 2. But for the training process we set these stream weighting parameters to global values dependent only on the stream type, Audio or Visual.

1.4 The default information fusion method of HTK

HTK [5] is a toolkit for Hidden Markov Models. It is basically used for speech recognition using HMM. As we stated in the introduction, it is used as the main tool for the setup of our recognition system.

HTK can handle simple 1-stream HMMs as well as multi-stream HMMs with synchrony in the state level as shown in figure 3. The transition probability matrix in these multi-stream HMMs is common for all streams, but the state-dependent emission probabilities are estimated separately. This makes the need for a final fusion of the independent emission probabilities in order to get a final result. For each stream the output emission probability is given by the next formula:

$$b_{js}(\boldsymbol{o}_{st}) = \sum_{m=1}^{M_s} c_{jsm} N(\boldsymbol{o}_{st}; \boldsymbol{\mu}_{jsm}, \boldsymbol{\Sigma}_{jsm}) \tag{2}$$

where o_{st} is the stream feature observation vector in time frame t, M_j is the number of gaussian mixture components in stream s, c_{jsm} is the weight of the m'th component and $N(\cdot;\mu,\Sigma)$ is a multivariate Gaussian with mean vector μ and covariance matrix Σ, that is

$$N(\boldsymbol{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}-\boldsymbol{\mu})} \tag{3}$$

where n is the dimensionality of o. The combination of the stream emission probabilities is done with the product rule. The formula for the final emission probability used by HTK is:

$$b_j(\boldsymbol{o}_t) = \prod_{s=1}^{S} [b_{js}(\boldsymbol{o}_{st})]^{\gamma_s} \quad (4)$$

where S is the number of streams used in multi-stream HMM and γ_s is the stream weight. In HTK the stream weight is set during the creation of the HMM and by default remains constant during the training and testing. We kept this default behavior during the training of the HMMs. In chapter 2 different information stream fusion methods are explained and the modifications we made in HTK to support the methods we used.

CHAPTER 2

Multi-stream information fusion methods

Information fusion of different sources of data or streams of information can be applied at various levels during pattern recognition: data level, feature level or decision level. Also hybrid methods are sometimes used. [6]

At the data and feature level, we can say the fusion is applied low-level. Data level fusion is not applied in AV-ASR. Feature level fusion is usually implemented by concatenating the feature vectors of different streams. In the case of AV-ASR a new AV feature vector is created with dimension $d_{av} = d_a + d_v$. This may create a very large feature vector, so various methods to reduce the dimensionality are often used. This low-level fusion method is not tested in our project. Although some concatenation of features happens in some point of the process, this has to do rather with the form of input vectors HTK accepts when running on multi-stream HMMs.

The methods explained in the next paragraphs of this chapter have to do with decision level fusion. In this level of fusion the reliability of each stream weight, thus of each modality in AV-ASR, can be modeled. This option is important because of the varying speech information content in audio and visual streams. The most commonly used architecture for decision fusion is the classifier combination using paraller architecture, adaptive combination weights, and class score level information. The combination is done with the adaptive product rule of the likelihoods or by linearly combining the log-

likelihoods of the two single-stream classifier decisions. [1]

To implement this approach we used multi-stream HMMs as already explained in Chapter 1. Using the same speech classes for both the separate single information stream classifiers (phonemes-visemes, we can combine their result at the time frame level. The final 2-stream information emission score – is not a probability distribution after the fusion – is given by the general rule:

$$P(o_{av,t}|c) = P(o_{a,t}|c)^{\lambda_{a,c,t}} P(o_{v,t}|c)^{\lambda_{v,c,t}} \quad (5)$$

for every HMM state $c \in C$. If log-likelihoods are used (5) becomes

$$\log[P(o_{av,t}|c)] = \lambda_{a,c,t} \log[P(o_{a,t}|c)] + \lambda_{v,c,t} \log[P(o_{v,t}|c)] \quad (6)$$

$\lambda_{a,c,t}$ and $\lambda_{v,c,t}$ are the audio and video stream weights respectively that are used in the final decision fusion. These are nonnegative, model the information stream reliability and are a function of stream *s,* HMM state *c*, and time frame *t*. In the next paragraphs different methods of fusion are explained that take different approaches in updating these weights, or relax the synchrony assumption of the multi-stream HMM already described.

2.1 Static methods

When static methods are used for the fusion of the information stream at the decision level then the stream weights remain constant in time and in HMM states and depend only on stream *s*. This gives from (5) the static combination rule:

$$P(o_{av,t}|c) = P(o_{a,t}|c)^{\lambda_a} P(o_{v,t}|c)^{\lambda_v} \quad (7)$$

Now the stream weights are λ_a and λ_v and depend only on the stream, either audio or video.

This method uses static weights during all the recognition process or at the utterance level. The estimation of the stream weights can be

done using some training data and with grid search on the stream weights. Usually the weights are constrained to add up to one (λ_a + λ_v = 1). So the grid search should be feasible. The fusion process is shown in figure 7.

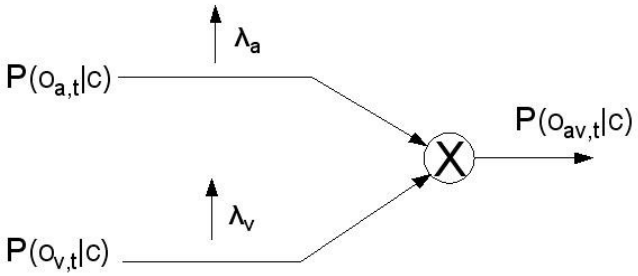


Figure 7*: Multistream recognition fusion with static weights*

The advantage of the static method is the easy and computational cheap implementation. The estimation of stream weights happen only once and are set to global static values. Also HTK [5] already supports this method, although no tools to estimate the stream weights have been implemented.

The drawback is that it cannot model the variability in time of the stream information reliability. During the recognition, the stream reliabilities change either inherently or because of external factors such as noise added to the information sources. The dynamic methods explained next try to model this variability.

2.2 Dynamic methods

Dynamic methods of fusion can be set up to model the variability in time of the information stream reliability in the multi-stream recognition process. In every information stream there is some time frames that the information carried in that stream is more accurate and reliable in order to make a decision on the recognition class. This reliability of the specific stream is not constant in time and can change

according to various factors. One factor is the inherent ability of the specific stream recognizer to better understand certain classes that are met compared to the other stream recognizers. The reason for this could be for example that some classes have more information in certain frequency bands. Also another factor is the degradation of the original signal from noise in certain time frames and in certain frequency band regions. This affects the accuracy of some recognizer results and not the others. Or one modality and not the other in the case of AV-ASR. All this variability can be modeled if the stream fusion weights are set to be adaptive in time.

For the bimodal AV-ASR, if the dependence on the state of HMM is removed the general formula (5) gives

$$P(o_{av,t}|c) = P(o_{a,t}|c)^{\lambda_{a,t}} P(o_{v,t}|c)^{\lambda_{v,t}} \quad (8)$$

and the stream weights become $\lambda_{a,t}$ and $\lambda_{v,t}$. Now the stream weights and consequently the final recognition score depend on the stream type (audio or video) and on the time frame *t*. The recognition process is shown in figure 8.

An important issue that emerges when using adaptive weights is how to estimate these weights. They must be based on some reliability indicator that shows the reliabilities of the different streams in time. This is the subject of chapter 3.

2.3 Asynchronous methods

When using multiple stream recognition there is an issue of synchrony between the streams. In the previous methods we assumed time synchrony of the streams. This is not always the case especially in AV-ASR. Up to 100ms asynchrony between the streams has been reported [1]. But also state asynchrony can be used in other recognition systems using different regions of the audio signal

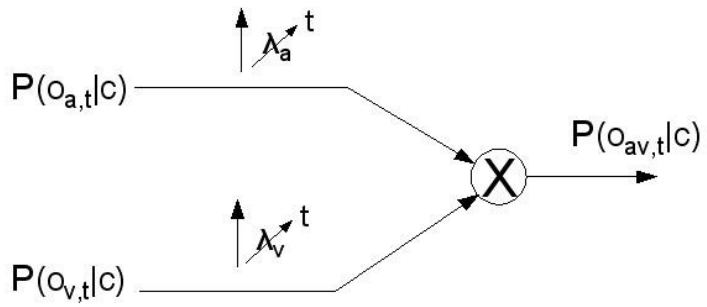


Figure 8*: Multi-stream recognition fusion with time varying weights*

spectrum as described in 1.2, because there is asynchrony in the transitions between stationary segments of speech at different frequency bands.

To model this asynchrony of streams we can use composite HMMs. These are a class of multi-stream HMMs that are created as the product of single stream HMMs [8]. Using product HMMs we can postpone the fusion of the likelihoods of the single stream classifiers at a later time. This can be the phoneme or word boundary. Allowing state asynchrony in the single stream HMM subcomponents of the product HMM, makes possible the modeling of asynchrony appearing between the streams.

The product HMM consists of composite states that are created by combining the states of the single stream HMM. For example the states of a product HMM that consists of 2 single stream HMMs with 3 states each has 9 states. Using mathematical formality to describe product HMM we can say in consists of composite states $c \in C^{|(S)|}$, $S=\{s_1, s_2\}$ in the case of 2-stream HMM, the cardinality of S equals 2 and the composite states are defined over the cartesian product $S \times S$ [1]. The combined emission output scores are

$$P(\boldsymbol{o}_{av,t}|c)=\prod_{s\in S} P(\boldsymbol{o}_{s,t}|c_s)^{\lambda_{c,s,t}} \tag{9}$$

where $c=\{c_s, s\in S\}$. This makes clear that the product HMM uses

the same number of parameters for mixture weight, mean and variance. But extra transitions are now needed between the composite states $P(\boldsymbol{c_1}|\boldsymbol{c_0}), \boldsymbol{c_1}, \boldsymbol{c_0} \in C^{|(S)|}$. A simplification can be applied giving the same number of transition probabilities as the state-synchronous multi-stream HMM, which gives $P(\boldsymbol{c_1}|\boldsymbol{c_0}) = \prod_{s \in S} P(c_{s1}|c_{s0})$. The degree of asynchrony can be limited by excluding composite states from the product HMM. In figure 9a a 2-stream state-synchronous HMM with 3 states in each single stream is shown, in figure 9b the equivalent product HMM with full asynchrony (2 states) is shown and finally in figure 9c the asynchrony is limited to 1 state.

2.4 HTK modifications

In 1.4 the default multiple stream information fusion method of HTK was explained. The formula to compute the final emission score of the multi-stream HMM was

$$b_j(\boldsymbol{o_t}) = \prod_{s=1}^{S} [b_{js}(\boldsymbol{o_{st}})]^{\gamma_s} \tag{10}$$

with γ_s the stream weight. For AV-ASR this corresponds to static information fusion method with the formula

$$P(o_{av,t}|c) = P(o_{a,t}|c)^{\lambda_a} P(o_{v,t}|c)^{\lambda_v} \tag{11}$$

and weights λ_α and λ_v that remain fixed during all the recognition process. Some changes in this default behavior of HTK were necessary to support the dynamic methods explained in 2.2. With our modifications the HTK formula for the information fusion was modified to

$$b_j(\boldsymbol{o_t}) = \prod_{s=1}^{S} [b_{js}(\boldsymbol{o_{st}})]^{\gamma_{s,t}} \tag{12}$$

with $\gamma_{s,t}$ the stream weight. The weights now can vary in time and this corresponds to the fusion formula

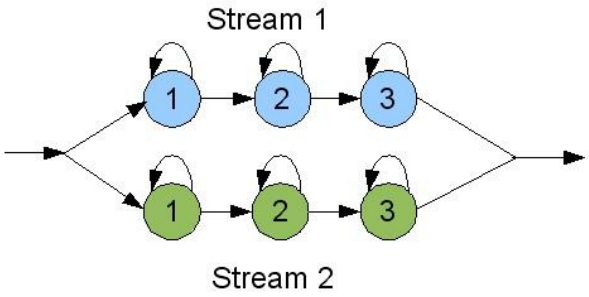


Figure 9a: 2-*stream state-synchronous HMM*

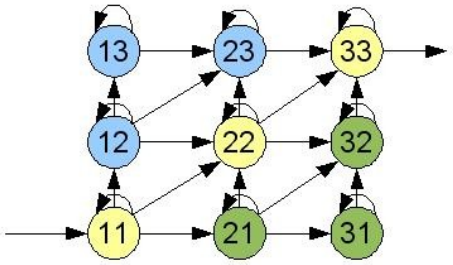


Figure 9b: 2-*stream product HMM with 2 state asynchrony*

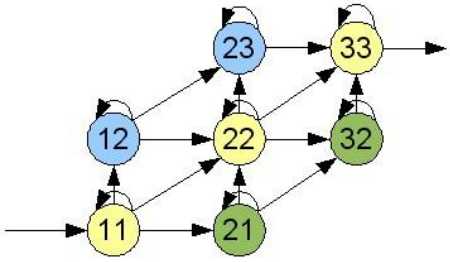


Figure 9c: 2-*stream product HMM with limited 1 state asynchrony*

$$P(o_{av,t}|c)=P(o_{a,t}|c)^{\lambda_{a,t}}P(o_{v,t}|c)^{\lambda_{v,t}} \tag{13}$$

of 2.2 with stream weights $\lambda_{a,t}$ and $\lambda_{v,t}$.

The HTK file format was used, the same that HTK uses for the input of the feature vectors. The stream weight vectors have the size of the number of streams and they count equal to the feature vector

number. The modifications were based on the weighting method used in [13] for time variable gaussian mixture weighting. The modifications of the HTK source code are presented in Appendix B.

CHAPTER 3

Information stream weights computation

When the multiple stream information fusion methods with stream weighting were described, the problem of stream weights estimation emerged. The problem is to find some methods to estimate the reliability of each stream in the final recognition decision. This is implemented by computing stream reliability indicators. There have been proposed different methods for this purpose. Two methods are described that are based on the single stream classification results [9][10]. After the reliability indicators have been computed then a mapping of this reliabilities to the information stream weights must be accomplished. Also a new method of computing stream weights using K-means clustering is explained [11] [12].

3.1 Stream Reliability Indicators

The stream reliability indicators are values that represent the reliabilities of the streams in multi-stream recognition. The estimation of the indicators is based on the single-stream classification / recognition probabilities. If one class of speech unit (phoneme, word) has very high probability and the other classes have very low probabilities then this stream is quite reliable. In contrast, when all classes have rather equal probabilities then this stream is not reliable for the final recognition decision. Two different approaches to the previous idea are described next.

3.1.1 Entropy method

The entropy method uses the entropy of the a posteriori probabilities of the single-stream recognition system to compute a reliability indicator [9]. These probabilities are

$$P(c_{i,t}|\boldsymbol{o}_{s,t}) \tag{14}$$

for class $c_{i,t}$ given the feature vector $o_{s,t}$ of stream *s* in time frame *t*. The average entropy over *N* time frames and *K* number of speech recognition classes is

$$H = -\frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{K} P(c_{i,t}|\boldsymbol{o}_{s,t})\log_2 P(c_{i,t}|\boldsymbol{o}_{s,t}) \tag{15}$$

This can be used as stream reliability indicator for the computation of the stream weights. Experiments in [9] have shown that for the computation of the entropy, only time frames for which the silence state is not among the 4 most probable must be taken into account.

3.1.2 Distance method

The distance method uses the difference of the class-conditional observation likelihoods of the N-best most likely generative classes. The likelihoods are

$$P(\boldsymbol{o}_{s,t}|c) \tag{16}$$

and they are ranked for the N-best selection. The stream reliability indicator that uses the log-likelihoods for better calculation is

$$L_{s,t} = \frac{1}{N-1}\sum_{n=2}^{N}\log\frac{P(\boldsymbol{o}_{s,t}|c_{s,t,1})}{P(\boldsymbol{o}_{s,t}|c_{s,t,n})} \tag{17}$$

for every stream *s* and time frame *t*.

In [10], it has been shown that this indicator monotonically changes with the degradation of the audio stream signal with noise. Also in the same work, a correlation of the indicator with the single-stream classification Word Error Rate (WER) has been shown to be -0.74 for the audio stream and -0.22 for the video stream (audio and video

stream reliability indicators respectively) on AV-ASR. These correlation results from [10] are presented in Table 1.

Reliability Indicator	Correlation with audio-only WER	Correlation with video-only WER
L_a	-0.74	0.02
L_v	0.10	-0.22

Table 1*: Correlation of stream reliability indicator with audio and visual-only WER [10]*

3.2 Reliability Indicator to Stream Weight Mapping

After estimating the stream reliability indicators we want to compute the actual stream weights to do the multi-stream information fusion. A mapping from the reliability indicators to stream weight values must be set.

In the case of the AV-ASR, we set up the stream weights to have some properties. We want the weights to sum up to 1:

$$\lambda_{a,t} + \lambda_{v,t} = 1 \tag{18}$$

From (18) we have

$$\lambda_{v,t} = 1 - \lambda_{a,t} \tag{19}$$

so we can set a mapping from the stream reliability indicators only to audio stream weight and then derive the visual stream weight from equation (19).

Usually a sigmoid function is chosen for the mapping. It has nice properties: it is bounded between zero and one, it is monotonic and smooth [10]. In formula (20) the sigmoid mapping is shown:

$$\lambda_{a,t} = \frac{1}{1 + \exp(-\sum_{i=1}^{N} w_i d_{i,t})} \tag{20}$$

where w_i are the mapping parameters corresponding to each reliability indicator, $d_{i,t}$ is the reliability indicator i at time frame t and N is the number of reliability indicators used. The mapping parameters must

be estimated from training data to maximize the final recognition results. There have been proposed different methods. We used a grid search method to find the value of w that minimized the total WER in the multi-stream recognition process. The implementation is described in chapter 4.

3.3 K-means Clustering Method

A new approach to compute the stream weights has been proposed in [11] and [12]. This method does not use a reliability indicator and a mapping function, but computes the weights directly. The idea is that the stream weights that minimize the total classification/estimation error are inversely proportional to the single-stream pdf estimation error. Also under certain conditions the optimal stream weights are inversely proportional to the single-stream classification error.

It has been shown in [11] for the two-stream classification case, when the classification error of the single-stream classifiers is equal $p(o_1|c_2) \approx p(o_2|c_1)$ then the stream weights that minimize the estimation error are

$$\frac{\lambda_1}{\lambda_2} = \frac{\sigma_{s_2}^2}{\sigma_{s_1}^2} \tag{21}$$

where $\sigma_{s_1}^2$ and $\sigma_{s_2}^2$ is the pdf estimation error variance of the first and second stream. Also when the single-stream estimation error variances are equal $\sigma_{s_1} = \sigma_{s_2}$ then for a region of interest where the single-stream classification errors are comparable according to equation

$$-1.5 \leqslant \frac{p(o_1|c_2)}{p(o_2|c_1)} \leqslant 1.5 \tag{22}$$

the stream weights should be inversely proportional to the above single-stream classification errors

$$\frac{\lambda_1}{\lambda_2} \approx \frac{p(o_2|c_1)}{p(o_1|c_2)} \tag{23}$$

If we want to extend the previous results to the multi-class recognition case then we can consider a class of discriminant functions $f_{i,j}(x)$ for each pair of classes w_i and w_j and also express the error as $P(error) = 1 - P(correct)$.

In the case of AV-ASR the computation of the stream weights can be done using the formula

$$\frac{\lambda_a}{\lambda_v} = \frac{\sigma_{sv}^2}{\sigma_{sa}^2} \frac{100 - WACC(m_v, D)}{100 - WACC(m_a, D)} \tag{24}$$

where $WACC(m_x,D)$ is the percent word accuracy of the single-stream classification done using model m_x for stream x. When this formula is applied to recognition the insertion and deletion errors must be handled accordingly. Also this formula assumes that the single-stream classification process is supervised, thus the class labels are known. For the a real world unsupervised scenario a new approach must be proposed.

In [12] a K-means clustering approach for the computation of the stream weights is proposed. The classification is done using a class and an anti-class model (class-specific background model) for each class. This reduces the multi-class problem to multiple single-class ones.

Anti-models for one class let's say c_i are created during training from data belonging to all other classes except class c_i. The method does unsupervised k-means classification using the class and anti-class. Then estimates the quantity $D = |(\mu_1 - \mu 2)| / \sigma$ by using inter- and intra-class distances to estimate the quantities in nominator and denominator. The inter-class distance is the average distance between the means of each class and the intra-class distance is an estimate of the average class variance. The stream weights for the case of two class classification are given by the next formula:

$$\frac{s_1}{s_2} = cf\left(\frac{d_{inter}(1,2;2)/\sum_i d_{intra}(i;2)}{d_{inter}(1,2;1)/\sum_i d_{intra}(i;1)}\right) \tag{25}$$

where c is a constant accounting for the difference is estimation error of the two streams, $f(\ )$ is a function relating D to the Bayes error, $d_{inter}(x, y; z)$ is the inter-class distance between classes x and y for stream z and $d_{intra}(x; y)$ is the intra-class distance for class x and stream y.

If we consider the multi-class case formula (25) becomes

$$\frac{s_1}{s_2} = cf\left[\sum_k \frac{d_{inter}(m_k, am_k; 2)/\sum_{i=(m_k, am_k)} d_{intra}(i;2)}{d_{inter}(m_k, am_k; 1)/\sum_{i=(m_k am_k)} d_{intra}(i;1)}\right] \tag{26}$$

where m_k and am_k are the centroids of the model and anti-model for class k computed with k-means clustering initialized with the model and anti-model means, and $\sum_k$ is over all classes. Note that the single stream estimation error variance is approximately constant for each stream under the recognition process.

3.4 HTK modifications

In our project we tested the distance method described in 3.1.2 to compute the reliability indicators of each stream. This method uses the class-conditional observation likelihoods of the N-best most likely generative classes. The class-conditional observation likelihood is $P(\boldsymbol{o}_{s,t}|c)$. So for every time frame we had to find these likelihoods. HTK unfortunately does not report these likelihoods although it uses them for the recognition results. So we had to make some tweaking in the HTK in order to report these likelihoods in every time frame. The classes we used are the speech class model states. The modifications are reported in Appendix B. In chapter 4 a more detailed description of the recognition system is available.

Recognition System CHAPTER 4

After describing the main theoretical concepts of our project in the previous chapters, we will now describe in chapter 4 the recognition system setup we used for the experiments done in multi-stream recognition. The streams we used are two, the audio and the visual and our system is characterized AV-ASR. Next we describe the various components of our system.

4.1 The data (CUAVE)

The data we used came from the CUAVE database [4]. This database came to fill in the gap in a common Audiovisual data set for the AV-ASR research community. Until then every researcher who wanted to make experiments in the AV-ASR area had to create his own data set, this made the results not directly comparable to other's. CUAVE is an Audiovisual speaker-independent database of connected (or isolated) continuous digit strings of high quality audio and video of a representative group of speakers. Different realistic conditions are included except the standard static speaker, such as moving speaker and multiple speakers.

The data included are separated in different parts and tasks. In part 1 there is only one speaker and in part 2 there is a pair of speakers. Also in different tasks the speaker is still, moving or in profile view. The speakers can be connected or continuous. In Table 2, there is a description of the CUAVE data in detail. For our experiments we used

Part	Task	Movement	Number of digits	Mode
(1) Individual	1	Still	50 x 36 speakers	Connected
	2	Moving	30 x 36 speakers	Connected
	3	Profile	20 x 36 speakers	Connected
	4	Still	30 x 36 speakers	Continuous
	5	Moving	30 x 36 speakers	Continuous
(2) Pairs	6	Still	(30 x 2) x 20 pairs	Continuous

Table 2*: CUAVE data set*

only task 1 of the database, which consists of 36 speakers, each one uttering 5 times the digits from zero to nine connected while standing still. Some natural movement of the speakers was unavoidable. The speakers were chosen to be as representative as possible, with 17 females and 19 males, with different skin tones.

The video was recorded at 720 x 480 resolution with 29.97 fps (NTSC) in full colour. The sound was recorded in 16-bit stereo at 44 Khz. Also 16-bit mono .wav files at 16 Khz with checked synchronization are included which we used in our system.

During our tests we used the round robin method due to limited training and testing data. Each time the training set was 30 speakers and the testing set was 6 speakers. The separation of the data set is shown in Table 3.

4.2 Noise types

In our recognition tests we injected some different types of audio noise in the audio data, in order to degrade the original audio speech signal. The noise data were acquired from the Signal Processing Information Base (SPIB) repository (http://spib.rice.edu/spib.html). The noise types we used were:

(a) Speech babble

(b) Factory floor noise 1

Test # / Speaker	1	2	3	4	5	6
1	TEST	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN
2	TEST	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN
3	TRAIN	TEST	TRAIN	TRAIN	TRAIN	TRAIN
4	TEST	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN
5	TRAIN	TEST	TRAIN	TRAIN	TRAIN	TRAIN
6	TRAIN	TEST	TRAIN	TRAIN	TRAIN	TRAIN
7	TRAIN	TRAIN	TRAIN	TEST	TRAIN	TRAIN
8	TRAIN	TRAIN	TRAIN	TRAIN	TEST	TRAIN
9	TRAIN	TRAIN	TEST	TRAIN	TRAIN	TRAIN
10	TRAIN	TRAIN	TEST	TRAIN	TRAIN	TRAIN
11	TRAIN	TRAIN	TRAIN	TRAIN	TEST	TRAIN
12	TRAIN	TRAIN	TRAIN	TEST	TRAIN	TRAIN
13	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TEST
14	TRAIN	TRAIN	TRAIN	TEST	TRAIN	TRAIN
15	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TEST
16	TRAIN	TRAIN	TRAIN	TRAIN	TEST	TRAIN
17	TRAIN	TRAIN	TEST	TRAIN	TRAIN	TRAIN
18	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TEST
19	TRAIN	TEST	TRAIN	TRAIN	TRAIN	TRAIN
20	TEST	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN
21	TRAIN	TEST	TRAIN	TRAIN	TRAIN	TRAIN
22	TRAIN	TRAIN	TRAIN	TRAIN	TEST	TRAIN
23	TRAIN	TRAIN	TRAIN	TEST	TRAIN	TRAIN
24	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TEST
25	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TEST
26	TRAIN	TRAIN	TEST	TRAIN	TRAIN	TRAIN
27	TRAIN	TRAIN	TRAIN	TRAIN	TEST	TRAIN
28	TRAIN	TRAIN	TRAIN	TEST	TRAIN	TRAIN
29	TRAIN	TRAIN	TRAIN	TEST	TRAIN	TRAIN
30	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN	TEST
31	TRAIN	TRAIN	TRAIN	TRAIN	TEST	TRAIN
32	TRAIN	TEST	TRAIN	TRAIN	TRAIN	TRAIN
33	TEST	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN
34	TEST	TRAIN	TRAIN	TRAIN	TRAIN	TRAIN
35	TRAIN	TRAIN	TEST	TRAIN	TRAIN	TRAIN
36	TRAIN	TRAIN	TEST	TRAIN	TRAIN	TRAIN

Table 3*: Separation of the initial data set to training and testing for different independent experiments (round-robin)*

(c) Jet cockpit noise 1

(d) Car interior noise

These represent different scenarios of environmental noise that the AV-ASR could be used in. The noise files were sampled at 20 Khz so we had to downsample first at 16 Khz to match with the audio data files.

4.3 Features

The feature extraction process from the data files was already described in 1.3.1 for both audio and visual streams. We used a vector of 39 features for the audio stream and a vector of 35 features for the video stream. We also concatenated the two vectors into one AV vector only because HTK uses one vector for both streams during multi-stream recognition for the input of the features. In fact the vector had nothing to do with the concatenated vector used in some feature fusion techniques where one common recognizer is used for both streams.

4.4 Recognizer Setup

In fact we setup three different recognizers to compare the results of the AV-ASR approach with audio-only ASR and to have an idea of the video contribution to the process. So we setup an audio-only, a video-only and finally a combined AV recognizer. All recognizer setups are explained in the next paragraphs.

4.4.1 HMM Definition

We performed isolated digit recognition, which means that each digit was set as a speech class. We created 10 HMMs corresponding to whole digits from zero to nine. Each model had 9 emitting states, and 1 input and 1 output state as required from HTK that are non-

emitting. We also created 1 HMM for the silence with 3 emitting states. For the audio-only recognizer the state mean and variance parameters were 39, one pair for each feature. For the video recognizer we used also the first and second derivatives of the DCT features so we had 105 mean and variance parameters in each state. For the AV recognizer we used the built-in multi-stream method of HTK with 2-streams. So the model was 2-stream HMM with 39 mean and variance pairs for the audio stream and 35 pairs for the video stream. The transition parameter matrix was common for both streams. In figure 10, parts an HMM file in HTK format are shown before the training for the three types of recognizers.

```
~h "proto"
~o <VecSize> 39 <MFCC_D_A>
<BeginHMM>
 <NumStates> 11
 <State> 2
  <Mean> 39
    0.0 0.0 0.0 .....
  <Variance> 39
    1.0 1.0 1.0 .....
 <State> 3
  <Mean> 39
    0.0 0.0 0.0 .....
    ..................

<TransP> 11
       0.0 1.0 0.0 0.0 ...
       0.0 0.5 0.5 0.0 ...
       0.0 0.0 0.5 0.5 ...
       ...................

 <EndHMM>
```

Figure 10a: *Audio-only recognizer HMM file part*

```
~h "proto"
~o <VecSize> 105 <MFCC_D_A>
<BeginHMM>
 <NumStates> 11
  <State> 2
   <Mean> 105
    0.0 0.0 0.0 ......
   <Variance> 105
    1.0 1.0 1.0 ......
  <State> 3
   <Mean> 105
    0.0 0.0 0.0 ......
    ..................

<TransP> 11
       0.0 1.0 0.0 0.0 ...
       0.0 0.5 0.5 0.0 ...
       0.0 0.0 0.5 0.5 ...
       ...................

 <EndHMM>
```

Figure 10b: *Video-only recognizer HMM file part*

```
~h "zero"
<BeginHMM>
 <StreamInfo>2 39 35
 <VecSize> 74
<MFCC><DIAGC>
 <NumStates> 11
 <State> 2
  <SWeights> 2
  0.8 0.2
  <Stream> 1
   <Mean> 39
    1.0 1.0 1.0 .....
   <Variance> 39
    1.0 1.0 1.0 .....
  <Stream> 2
   <Mean> 35
    1.0 1.0 1.0 .....
   <Variance> 35
    1.0 1.0 1.0 .....
 <State> 3
  <SWeights> 2
  0.8 0.2
  <Stream> 1
   <Mean> 39
    1.0 1.0 1.0 ....
```

Figure 10c: *Audiovisual recognizer HMM file part*

4.4.2 HMM Training

After defining the HMM for each speech class, we continued with the training of the parameters of the models. The data we used (CUAVE) were already labeled and HTK format label files were included. So the training process was done automatically with the following procedure:

- First we used HInit from HTK [5] to provide initial estimates of the parameters of HMM using the training data. HInit repeatedly uses Viterbi alignment to segment the training observations and then recomputes the parameters by pooling the vectors in each segment.
- Then we used HREST to perform basic Baum-Welch re-estimation of the parameters of HMM using again the training data set.

At this point we had the trained models for each of the 10 digits plus the silence. The process of training was the same for all three types of recognizers, except different training observation vectors were used according to modality. The training of the AV models was done with jointly estimated transition probabilities for both streams and with stream weights set to $\lambda_a = 0.8$ and $\lambda_v = 0.2$ during the training.

4.4.3 Dictionary and Word Net

The dictionary we used was simply the 10 digits (zero to nine) plus the silence token.

The word network we created is shown in figure 11. It allows

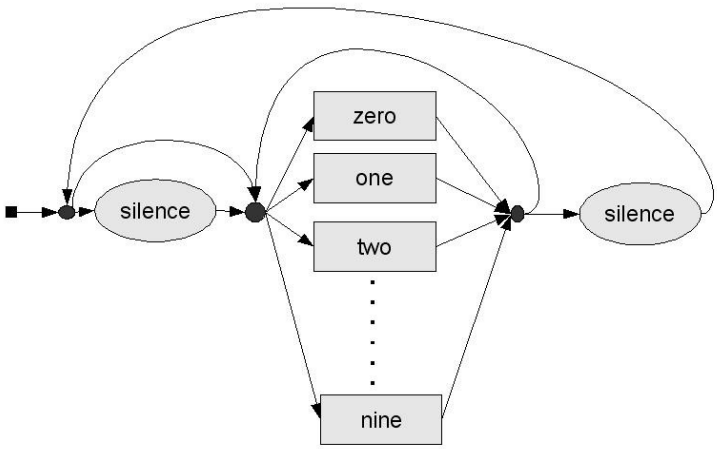


Figure 11: Word Network

optional silence to appear before and after each digit.

4.5 Weight Computation

For the Audiovisual recognizer, the fusion of the audio and visual stream information is a main task. As described in the previous chapters, we used decision level fusion with stream weighting. This involves the problem of computing the stream weights. We used both static and dynamic methods of weighting as described in Chapter 2.

For static methods we setup a standard weighting system with fixed weights during recognition at $\lambda_\alpha = 0.8$ and $\lambda_v = 0.2$. We also made another setup and done grid search to find the best static stream weights for every situation in our experiments. This was done in order to compare the best results achieved with the static method to the dynamic methods. The grid search was done between the values $0.5 \leqslant \lambda_\alpha \leqslant 1$ with step 0.04. Although this method to find the best static weights is not practical, because it requires constant computation of stream weights from training data (with known labels) every time the environmental situation changes, we used it to make a comparison of the methods.

For dynamic methods we first had to compute the reliability indicators of the streams. We computed two reliability indicators, one for each stream, as described in 3.1.2 using Distance method. For the computation of the audio stream reliability indicator we used the equation (17) and 4-best method so we had:

$$L_{a,t} = \frac{1}{3} \sum_{n=2}^{3} \log \frac{P(\boldsymbol{o}_{a,t} | c_{a,t,1})}{P(\boldsymbol{o}_{a,t} | c_{a,t,n})} \tag{27}$$

and for the visual stream reliability indicator we used 2-best method so we had:

$$L_{v,t} = \log \frac{P(\boldsymbol{o}_{a,t} | c_{a,t,1})}{P(\boldsymbol{o}_{a,t} | c_{a,t,2})} \tag{28}$$

After computing the reliability indicators, we made 2 different

mappings to the stream weights as described in 3.2. The first mapping uses the audio stream reliability indicator and maps it to the audio stream weight. So we have:

$$\lambda_{a,t}=\frac{1}{1+\exp(-w_a L_{a,t})} \tag{29}$$

and the visual stream weight is $\lambda_{v,t}=1-\lambda_{a,t}$.

The second mapping we tested, uses both the audio and visual stream reliability indicators and maps them to the audio weight using the equation:

$$\lambda_{a,t}=\frac{1}{1+\exp(-w_a L_{a,t}-w_v L_{v,t})} \tag{30}$$

and the visual stream weight is again $\lambda_{v,t}=1-\lambda_{a,t}$.

For the first mapping we have to estimate the mapping parameter value w_a that best computes the audio stream weight from the audio

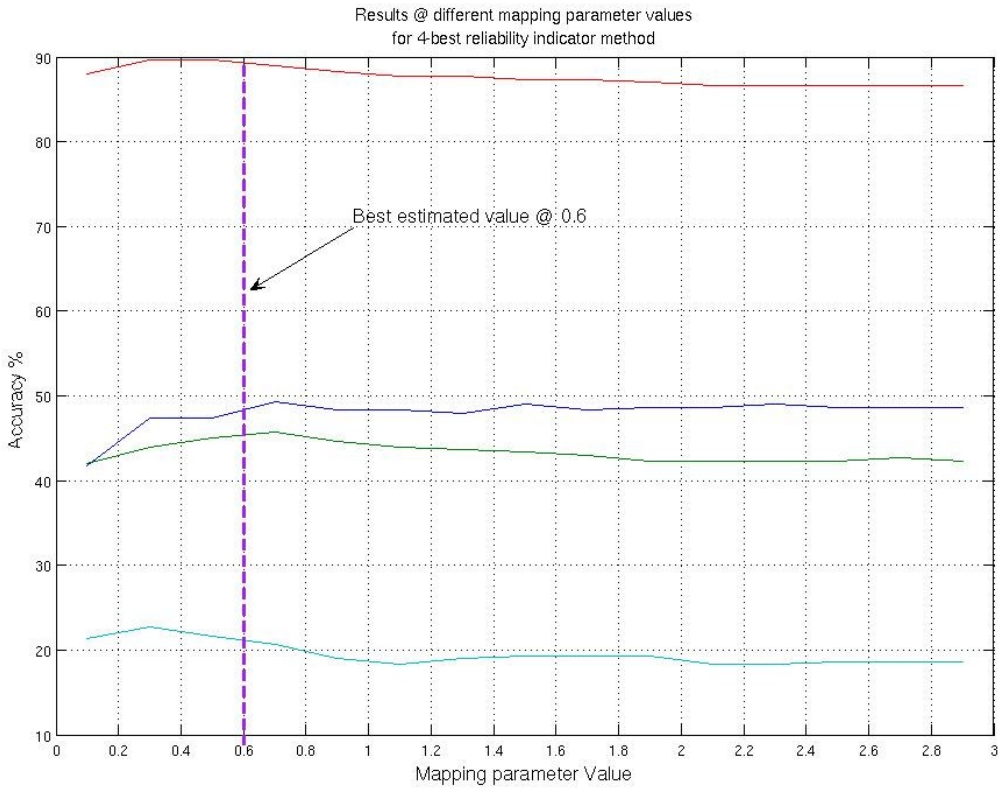


Figure 12*: Grid search results of the mapping parameter value for the general case*

reliability indicator. We searched for the best mapping parameter in two scenarios, in the first we made a general mapping that minimizes the WER under all noise conditions, and in the second we made a mapping specifically for each environmental situation with the same goal of minimizing the WER but under that noise condition only.

The best parameter value was selected with grid search. To know the region in where to search for the best parameter, we first created a histogram of the audio reliability indicator. The indicator is mostly between the values 0.5 and 3. Near value 0.5 the stream is quite unreliable, so a stream weight not near 1 is required. If we set $w_a = 4$ and for $L_{a,t}=0.5$ equation (29) gives $\lambda_{a,t}=0.88$, a value that is quite close to 1. For mapping parameter values greater than 4 the mapping function would give stream weight values even closer to 1 for $L_{a,t}=0.5$, something not desired. So the max limit of the grid search was set to 4. The region we searched was set to $0<w_a\leqslant 4$ with step 0.2.

In the first scenario where we wanted to establish a general mapping for all environmental noise conditions, we wanted a mapping parameter that minimizes the WER for all noise conditions. Now we can explain why we have chosen the 4-best method to compute the audio reliability indicator. As shown in table 4 we tested different N-best methods with N from 2 to 5 for 4 different environmental noises, Babble speech at 3dB SNR, Factory floor noise at 3dB SNR, Car

	Babble @ 3db		Factory @ 3db		Car @ 0db		Jet 0db	
Method	Value	Acc %	Value	Acc %	Value	Acc %	Value	Acc %
2-best	2.3	49	1.2	45.33	0.6	90.33	0.5	22.33
3-best	2.7	49.33	0.9	46	0.3	89.67	0.3	22
4-best	**0.7**	**49.33**	**0.7**	**45.67**	**0.4**	**89.67**	**0.3**	**22.67**
5-best	2.7	49.33	0.7	45.33	0.5	89.67	0.3	22

Table 4*: Different N-best method results for the computation of the audio reliability indicator*

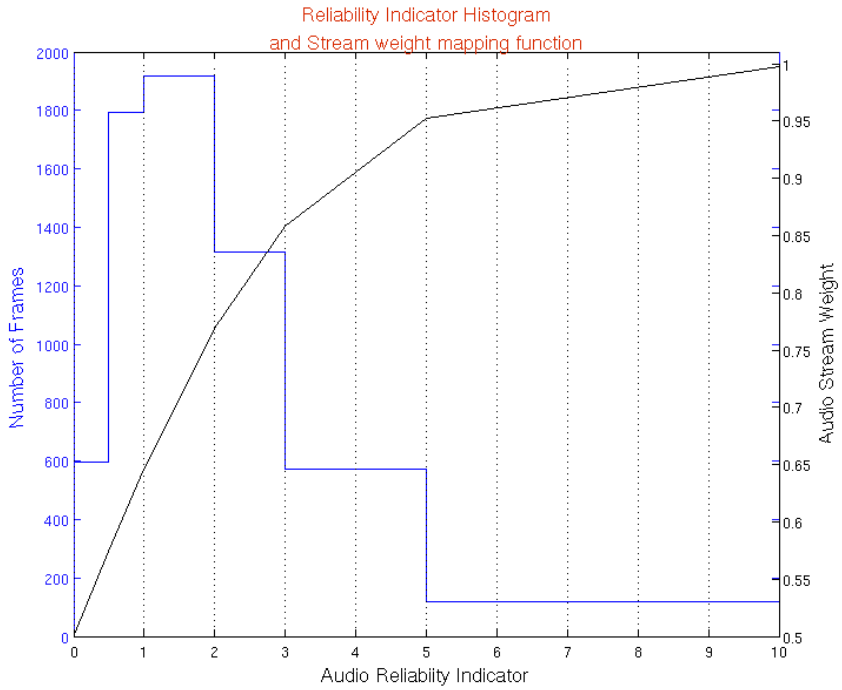


Figure 13*: Audio reliability histogram with audio stream weight mapping function graph*

interior noise at 0dB SNR and Jet cockpit noise at 0dB SNR and the results we got for the best parameter reveal that the 4-best is ideal for the creation of a general mapping because it allows a relative convergence of the mapping parameter between different environmental noises.

The results of the grid search for the 4 environmental noise conditions described above are shown in figure 12. The best estimated parameter value was found to be $w_a = 0.6$. Also in figure 13 are shown the reliability indicator histogram extracted from audio signal with factory floor noise at 10dB SNR plotted together with the mapping function with parameter $w_a = 0.6$.

In the second scenario, we wanted to establish an adapted mapping for every environmental noise condition. So we wanted to find the best mapping parameter for each noise type. We have done separate grid search for all 4 noise types and for SNR values 10, 3 and 0 dB. The

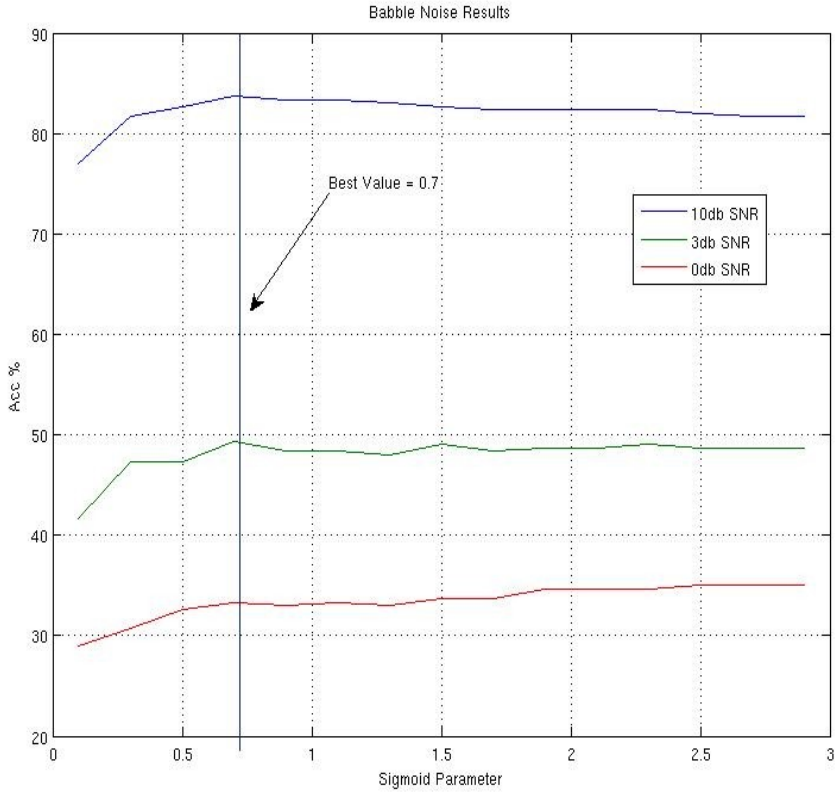


Figure 14(a): *Babble speech mapping parameter grid search results*

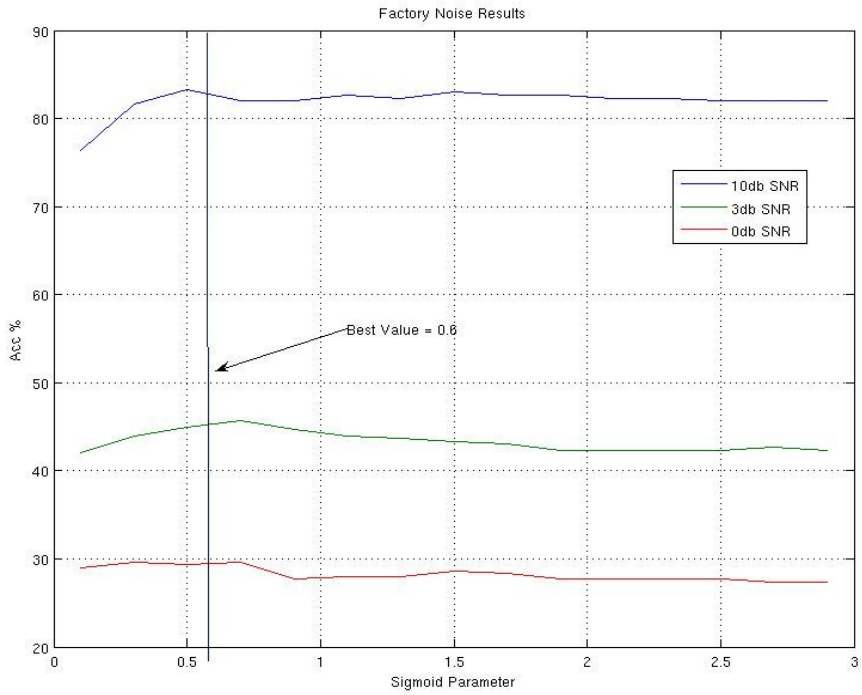


Figure 14(b)*: Factory floor noise mapping parameter grid search results*

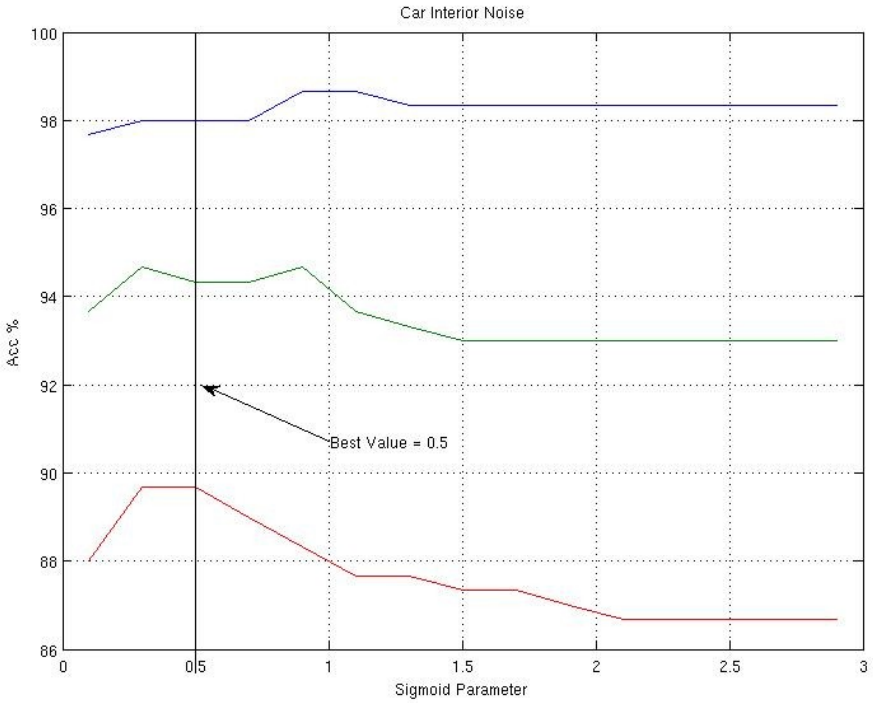


Figure 14(c): *Car interior noise mapping parameter grid search results*

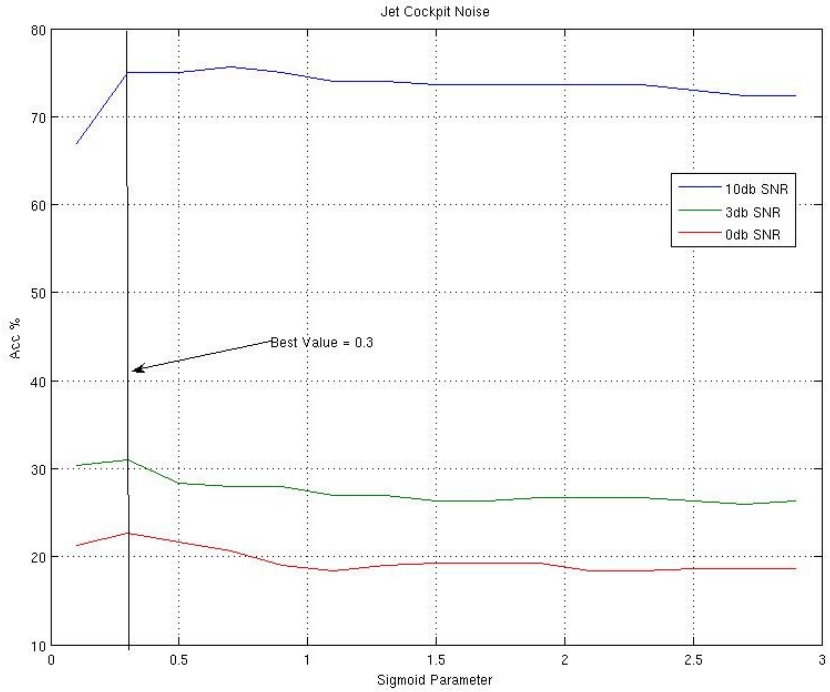


Figure 14(d): *Jet cockpit noise mapping parameter grid search results*

best values for the mapping parameter are shown in table 5. The grid search results are shown in figure 14.

Noise type	Babble speech	Factory floor noise	Car interior noise	Jet cockpit noise
w_a parameter value	0.7	0.6	0.5	0.3

Table 5: *Mapping parameter value for noise adapted mapping*

For the second mapping we used both audio and visual stream reliability indicators to compute the audio stream weight and then the video stream weight. So we wanted to estimate two mapping parameters w_a and w_v. First we created the histogram of the visual reliability indicator as we did for audio above. Because of the positive correlation of the visual reliability indicator with the audio only WER and the linear combination of the indicators inside the exp function of the sigmoid, we now allow the mapping parameters to get negative values. We set the grid search region to be $-1 < w_a \leqslant 3$ and $-0.5 < w_v \leqslant 1.5$. We used variable step which was 0.1 near 0 and about 0.3 near the edges. The best mapping parameter pairs for each environmental noise type are shown in table 6. Multiple best pairs were extracted in some cases, representing the flexibility in choosing the best pair, a flexibility thanks to the linear combination of the indicators. Detailed results are shown in figure 15.

	Babble @ 3 dB	Factory @ 3 dB		Car @ 0 dB				Jet @ 0 dB
w_a	1.2	1.2	1.6	0.5	0.8	1	1.2	1
w_v	0.5	0.3	0.1	0	-0.1	-0.1	-0.3	-0.3

Table 6: *Mapping parameter values when both indicators are used.*

The final mapping parameter values pair that gave the most promising results for all environmental condition situations was w_a = 1.2 and w_v = -0.1. The negative value in w_v is explained by the fact that the greater reliability of the video stream would reduce the audio stream weight, thus increasing the visual stream weight from $\lambda_{v,t} = 1 - \lambda_{a,t}$.

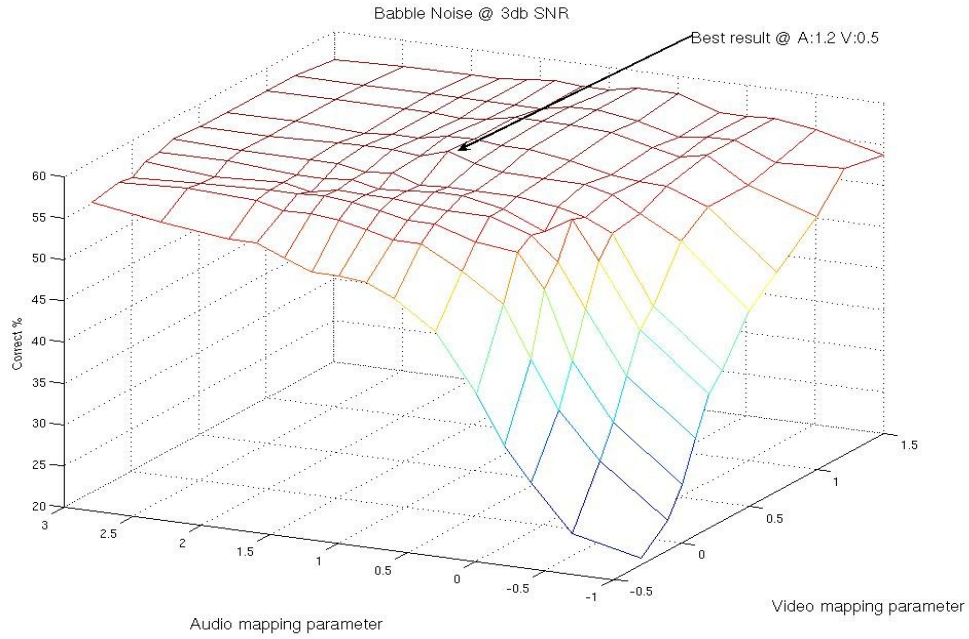


Figure 15(a): *Grid search for audio and video mapping parameters (Babble speech)*

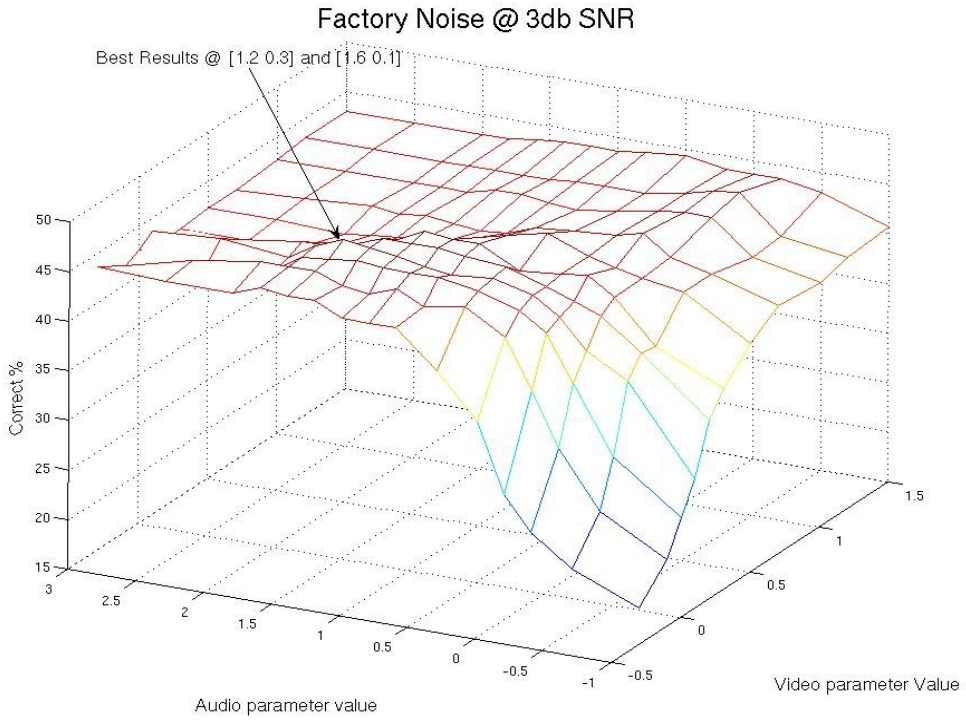


Figure 15(b): *Grid search for audio and video mapping parameters (Factory floor noise)*

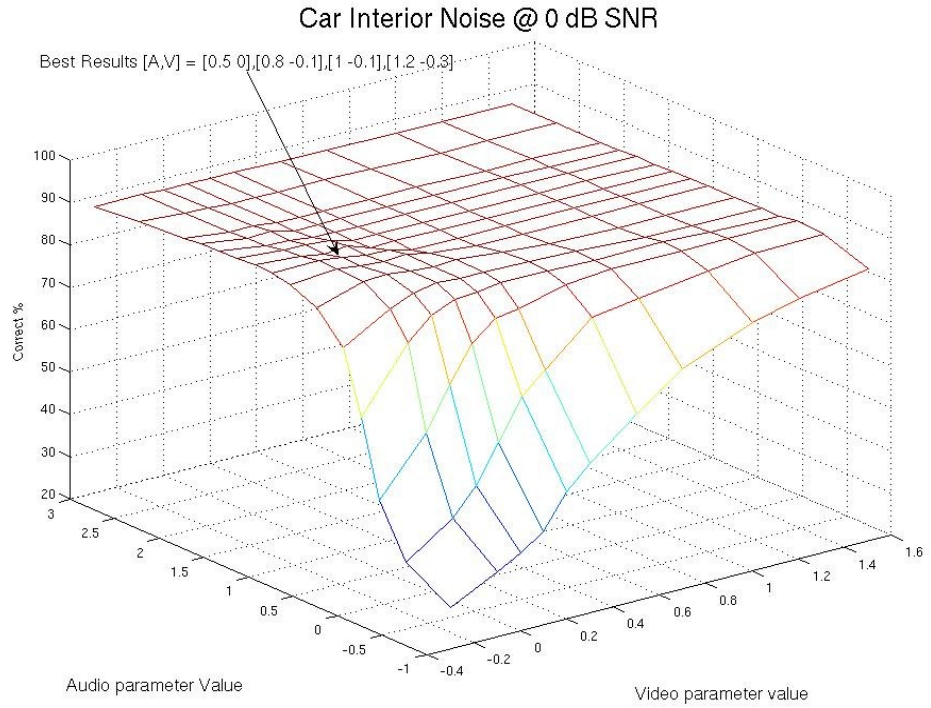


Figure 15(c): *Grid search for audio and video mapping parameters (Car interior noise)*

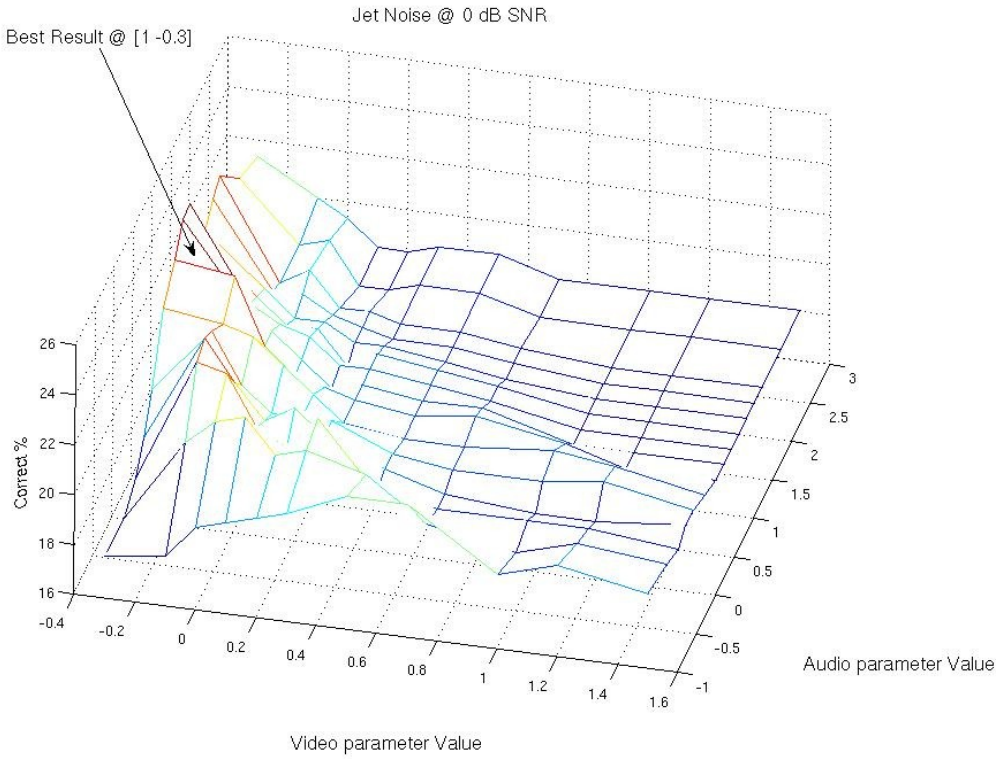


Figure 15(d): *Grid search for audio and video mapping parameters (Jet cockpit noise)*

Finally after the mapping function was set and the audio stream weight was computed we tested a mean smoothing function on the stream weight. The results for Speech babble injected Audio signal at 10dB SNR and different sizes of the filter are shown in table 7. The 3 frames mean smoothing filter was chosen.

	No smoothing	3 frames mean filter	5 frames mean filter
Accuracy %	48.67	49.33	48.33

Table 7: *Mean smoothing filter test results*

From the audio stream weight we computed the visual stream weight as shown previously.

Tests Results CHAPTER 5

The results in this chapter are presented in a format that allows easy comparison between the different AV methods described above and the Audio-only recognizer. Also the results of the Visual-only recognizer are included. So in every situation the results of the single-stream recognizers are included.

5.1 Audio stream reliability mapping recognizer

Here we present the results for the audio stream reliability mapping recognizer. The setup that uses a general mapping for all environmental conditions is presented as "General Mapping" and the setup that uses adapted mappings for each environmental noise type is presented as "Adapted Mapping". Also two static method results are included, the first uses fixed stream weights at $\lambda_\alpha = 0.8$ and $\lambda_v = 0.2$ and the second pre-computes the best static pair that maximize the accuracy on the training set. The original audio signal is injected with the 4 types of noise described in 4.2 at different SNR level: 15dB, 10dB, 3dB, 0dB, -3dB. The Visual-only recognizer results are not affected by audio noise and are fixed. The Audio-only and AV results are affected. Finally the percent reduction on WER between the static method with fixed weights $\lambda_\alpha = 0.8$ and $\lambda_v = 0.2$ and the two dynamic methods is presented for comparison given by the formula

$$relative\ WERR\ reduction = \frac{WACC_{new} - WACC}{100 - WACC} \quad . \quad (31)$$

Speech Babble Noise

SNR Recognizer type	15 dB	10 dB	3 dB	0 dB	-3 dB
Audio-only	93,02	80,9	49,08	35,83	27,17
Video-only	28,35				
AV Global (0.8-0.2)	90,3	81,58	48,83	35,66	27,06
AV-Global Best Values	93,91	82,08	49,94	36,4	28,01
AV-Instant General Mapping	93,18	82,25	49,44	36,33	27,45
AV-Instant Adapted Mapping	93,4	82,25	49,21	36,63	27,45
AV-Instant-GM rel. WERR red.	29,69%	3,64%	1,19%	1,04%	0,53%
AV-Instant-AM rel. WERR red.	31,96%	3,64%	0,74%	1,51%	0,53%

Table 8: *Speech Babble accuracy results for Audio reliability mapping method*

Jet Cockpit Noise

SNR Recognizer type	15 dB	10 dB	3 dB	0 dB	-3 dB
Audio-only	89,96	71,24	31,87	22,68	17,72
Video-only	28,35				
AV Global (0.8-0.2)	91,14	73,48	33,48	23,58	18,5
AV-Global Best Values	91,52	74,65	35,1	25,09	19,72
AV-Instant General Mapping	91,25	75,32	35,1	24,42	18,72
AV-Instant Adapted Mapping	90,74	72,69	33,98	25,15	19,11
AV-Instant-GM rel. WERR red.	1,24%	6,94%	2,44%	1,10%	0,27%
AV-Instant-AM rel. WERR red.	-4,51%	-2,98%	0,75%	2,05%	0,75%

Table 9: *Jet Cockpit noise accuracy results for Audio reliability mapping method*

Car Interior Noise

SNR Recognizer type	15 dB	10 dB	3 dB	0 dB	-3 dB
Audio-only	98,6	98,1	93,64	88,99	83,27
Video-only	28,35				
AV Global (0.8-0.2)	98,27	97,56	93,31	89,35	83,2
AV-Global Best Values	98,72	97,71	93,92	90,07	84,32
AV-Instant General Mapping	98,66	97,78	93,81	89,91	84,21
AV-Instant Adapted Mapping	98,38	97,6	93,87	90,02	84,32
AV-Instant-GM rel. WERR red.	22,54%	9,02%	7,47%	5,26%	6,01%
AV-Instant-AM rel. WERR red.	6,36%	1,64%	8,37%	6,29%	6,67%

Table 10: *Car Interior noise accuracy results for Audio reliability mapping method*

Factory Floor Noise

SNR Recognizer type	15 dB	10 dB	3 dB	0 dB	-3 dB
Audio-only	93,03	81,41	45,47	30,75	18,5
Video-only	28,35				
AV Global (0.8-0.2)	93,8	82,3	45,57	31,47	20,89
AV-Global Best Values	94,25	83,03	46,92	32,76	22,29
AV-Instant General Mapping	94,08	82,57	46,96	32,64	21,34
AV-Instant Adapted Mapping	94,08	82,57	46,96	32,64	21,34
AV-Instant-GM rel. WERR red.	4,52%	1,53%	2,55%	1,71%	0,57%
AV-Instant-AM rel. WERR red.	4,52%	1,53%	2,55%	1,71%	0,57%

Table 11: *Factory floor noise accuracy results for Audio reliability mapping method*

The results are shown in tables 8 – 11 for different noise types.

By carefully examining the results, we can support the conclusion that the Audiovisual methods are almost always superior to the Audio-only method despite the fact that the Video-only single-stream classifier accuracy is relatively low. The difference of the AV recognizers from the Audio-only increases as the SNR decreases and the audio stream becomes less reliable. Also the instantaneous stream weight computation AV methods have in general better results from the fixed AV method as shown from the two last lines of the tables, except in the case of the Adapted Mapping in the Jet Cockpit case that is probably caused by a loose adaptation scheme. The superiority of the time-variable weighting methods shows that the recognition is better adapted to the variability of the audio stream reliability during the process. Also shows the flexibility of the methods as the SNR changes. When comparing the instantaneous methods with the best pre-computed static weights method, they have quite similar results. Sometimes the best static method has better results, a fact that shows that the instantaneous methods can give even better results with maybe some tweaking in the stream weights computation part. Also when comparing the two time-variable weights methods, we can support that the general all-noise-condition mapping recognizer has very good results and the adapted to noise mapping recognizer is only better in very low SNR but with almost identical results. This makes the adaptation to noise type redundant, based on our results, which is a good thing assuming that in real world the noise types may change from time to time. In figure 16 the % WERR reduction is plotted for different noise types and SNR for the two time-variable weights AV methods. In figure 16(d) where the factory noise results are presented, there is only one visible line because the general and the adapted mapping setups are identical. No special adaptation of the mapping

parameterization is needed in the case of factory floor noise

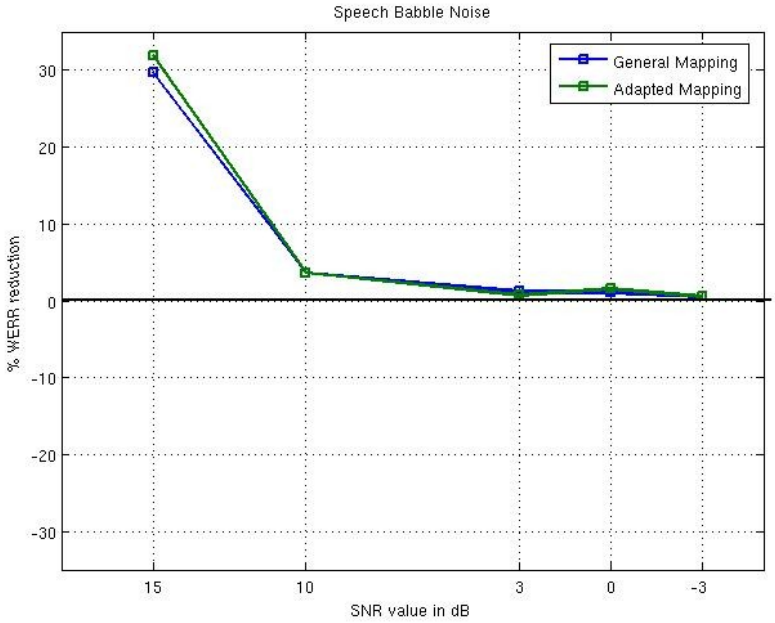


Figure 16(a): *Speech Babble noise results (Audio mapping)*

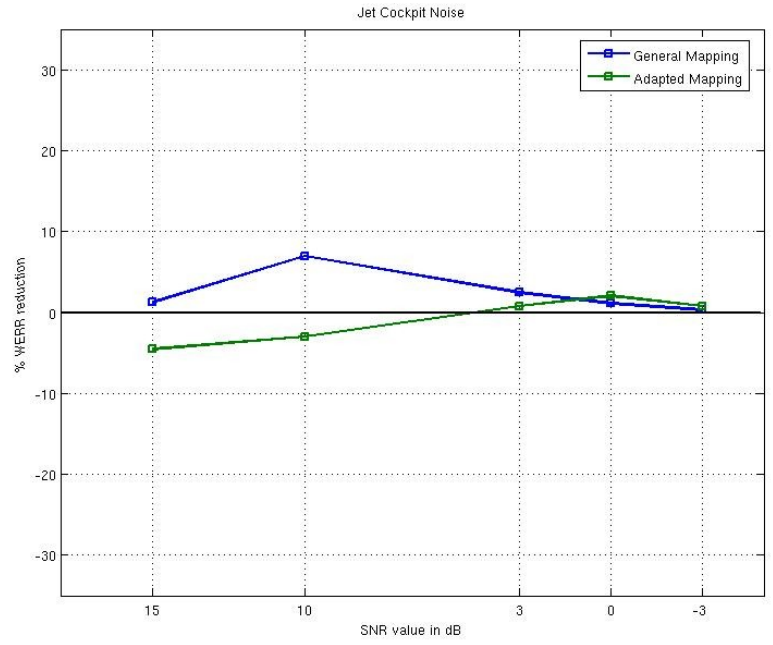


Figure 16(b): *Jet Cockpit noise results (Audio mapping)*

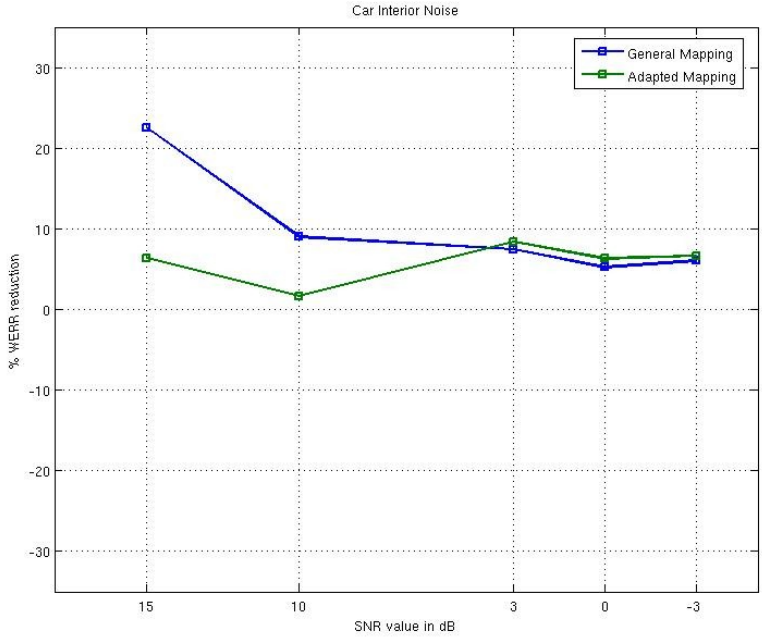


Figure 16(c): *Car Interior noise results (Audio mapping)*

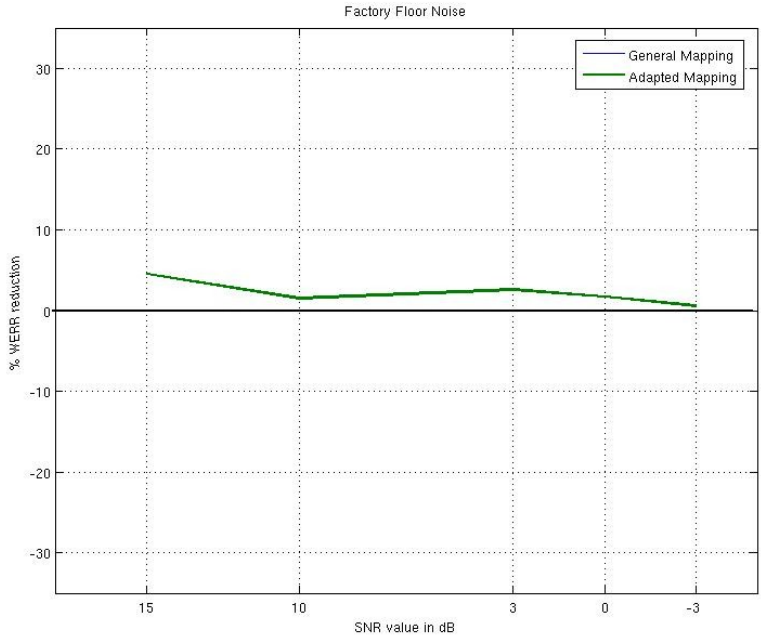


Figure 16(d): *Factory Floor noise results (Audio mapping)*

In table 12 the change in the audio stream weights computed by the three AV methods (best pre-computed fixed, instantaneous general - mean values, instantaneous noise type adapted - mean values) as the SNR changes is shown. We can see that the mean values of the time-variable weighting methods follow a downward trend as the SNR decreases, which shows adaptability of the method to decreasing audio stream reliability. Except in the Car interior noise that the stream means are stuck in the same values. This is a desirable behaviour in fact, as the car noise is present in low spectral

	Noise type	15 dB	10 dB	3 dB	0 dB	-3 dB
AV-Global Best Values	Babble Speech	0,88	0,86	0,86	0,91	0,83
AV-Instant General Mapping		0,84	0,82	0,79	0,78	0,78
AV-Instant Adapted Mapping		0,84	0,82	0,8	0,78	0,78
AV-Global Best Values	Jet Cockpit noise	0,82	0,79	0,69	0,58	0,62
AV-Instant General Mapping		0,82	0,79	0,76	0,76	0,77
AV-Instant Adapted Mapping		0,7	0,67	0,65	0,65	0,65
AV-Global Best Values	Car Interior noise	0,92	0,91	0,84	0,78	0,74
AV-Instant General Mapping		0,87	0,87	0,86	0,86	0,86
AV-Instant Adapted Mapping		0,83	0,83	0,82	0,82	0,82
AV-Global Best Values	Factory Floor noise	0,86	0,92	0,83	0,74	0,59
AV-Instant General Mapping		0,83	0,8	0,77	0,76	0,76
AV-Instant Adapted Mapping		0,83	0,8	0,77	0,76	0,76

Table 12: *Stream weight comparison for different AV methods (Audio mapping)*

region and does not affect the recognition process considerably as it is shown in the Accuracy table results. Another fact is that the weights of the instantaneous methods have a small region of variance and do not follow the best fixed weights in large inclines or declines. This happens because of the sigmoid characteristics which is chosen as a mapping function. The region of variance of the time-variant stream weight values can be increased by tweaking the mapping function, but if this would give better recognition results remains to be tested. In figure 17 the above stream weights are graphically shown.

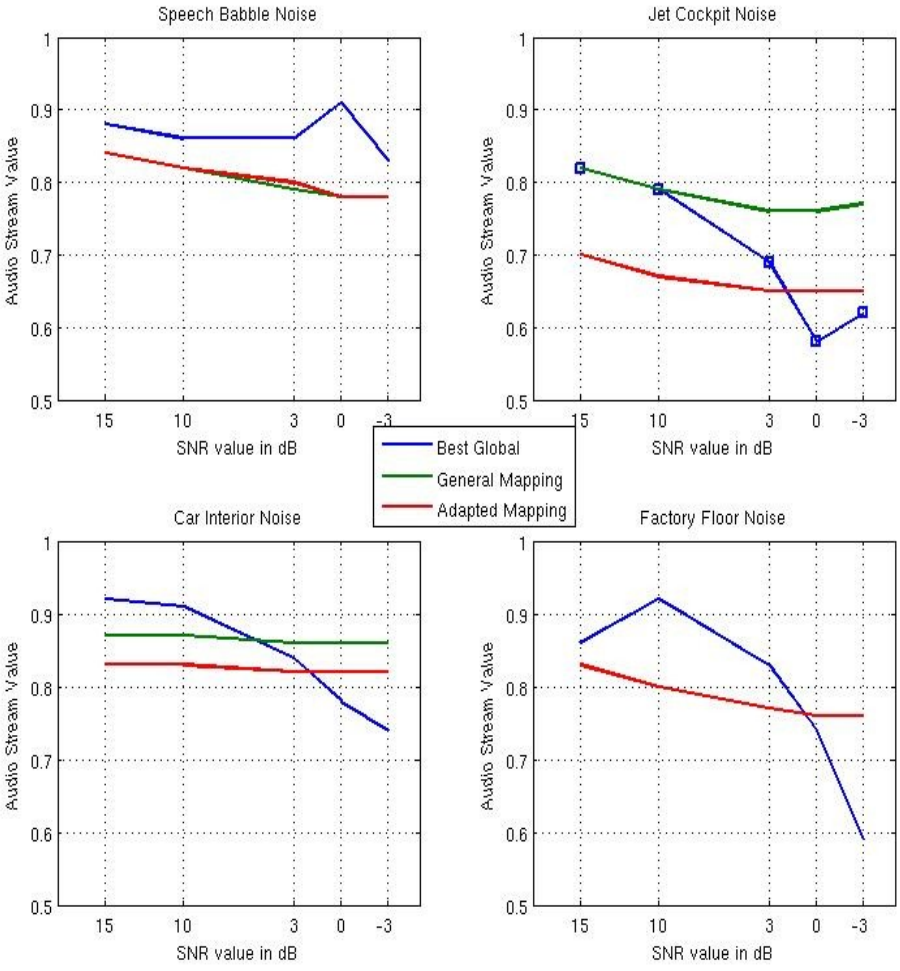


Figure 17: *Audio stream weight graphs for different AV Methods (Audio mapping)*

5.2 Audio and Video stream reliability mapping recognizer

The same results are now presented but for the setup that uses both stream reliability indicators to perform the computation of the stream weights. Careful inspection of the results shows that this method gives similar or inferior results to the previous method. This can be explained from the fact that the best mapping of two values is harder to accomplish than of one and possibly some more work can be done in the mapping parameter estimation of the method. Also because of the low accuracy, the single-stream visual classifier gives decision information that might be more confusing than helpful for the computation of the stream weights. Improvement in this field could overall improve this method. The recognition results in terms of accuracy are shown in

Speech Babble Noise

SNR Recognizer type	15 dB	10 dB	3 dB	0 dB	-3 dB
Audio-only	93,02	80,9	49,08	35,83	27,17
Video-only	28,35				
AV Global (0.8-0.2)	90,3	81,58	48,83	35,66	27,06
AV-Global Best Values	93,91	82,08	49,94	36,4	28,01
AV-Instant General Mapping	93,02	81,57	48,88	35,84	26,21
AV-Instant Adapted Mapping	93,24	82,36	48,99	36,74	27,46
AV-Instant-GM rel. WERR red.	28,04%	-0,05%	0,10%	0,28%	-1,17%
AV-Instant-AM rel. WERR red.	30,31%	4,23%	0,31%	1,68%	0,55%

Table 13: *Speech Babble accuracy results for Audio+Video reliability mapping method*

Jet Cockpit Noise

SNR Recognizer type	15 dB	10 dB	3 dB	0 dB	-3 dB
Audio-only	89,96	71,24	31,87	22,68	17,72
Video-only			28,35		
AV Global (0.8-0.2)	91,14	73,48	33,48	23,58	18,5
AV-Global Best Values	91,52	74,65	35,1	25,09	19,72
AV-Instant General Mapping	90,91	73,24	34,42	24,47	18,61
AV-Instant Adapted Mapping	89,18	70,19	32,92	24,41	19,39
AV-Instant-GM rel. WERR red.	-2,60%	-0,90%	1,41%	1,16%	0,13%
AV-Instant-AM rel. WERR red.	-22,12%	-12,41%	-0,84%	1,09%	1,09%

Table 14: *Jet Cockpit noise accuracy results for Audio+Video reliability mapping method*

Car Interior Noise

SNR Recognizer type	15 dB	10 dB	3 dB	0 dB	-3 dB
Audio-only	98,6	98,1	93,64	88,99	83,27
Video-only			28,35		
AV Global (0.8-0.2)	98,27	97,56	93,31	89,35	83,2
AV-Global Best Values	98,72	97,71	93,92	90,07	84,32
AV-Instant General Mapping	98,27	97,49	94,03	89,79	84,26
AV-Instant Adapted Mapping	98,21	97,54	93,98	90,07	84,37
AV-Instant-GM rel. WERR red.	0,00%	-2,87%	10,76%	4,13%	6,31%
AV-Instant-AM rel. WERR red.	-3,47%	-0,82%	10,01%	6,76%	6,96%

Table 15: *Car Interior noise accuracy results for Audio+Video reliability mapping method*

Factory Floor Noise

SNR Recognizer type	15 dB	10 dB	3 dB	0 dB	-3 dB
Audio-only	93,03	81,41	45,47	30,75	18,5
Video-only	28,35				
AV Global (0.8-0.2)	93,8	82,3	45,57	31,47	20,89
AV-Global Best Values	94,25	83,03	46,92	32,76	22,29
AV-Instant General Mapping	93,8	82,19	46,23	32,64	21,4
AV-Instant Adapted Mapping	94,36	82,47	46,01	30,85	20,73
AV-Instant-GM rel. WERR red.	0,00%	-0,62%	1,21%	1,71%	0,64%
AV-Instant-AM rel. WERR red.	9,03%	0,96%	0,81%	-0,90%	-0,20%

Table 16: *Factory floor noise accuracy results for Audio+Video reliability mapping method*

tables 13 to 16 and compared to the fixed weight to $\lambda_a = 0.8$ and $\lambda_v = 0.2$ static method. Also in figure 18 the percent relative WER reduction is plotted for the different environmental noise types and SNR.

In table 17 the change in audio stream weights computed by the three AV methods as the SNR changes is shown for this method. The stream weights are changing very slowly and this makes the method really inflexible to changing environmental conditions. Perhaps a better mapping parameterization may give the desired flexibility. In figure 19 the weight values are plotted together for comparison.

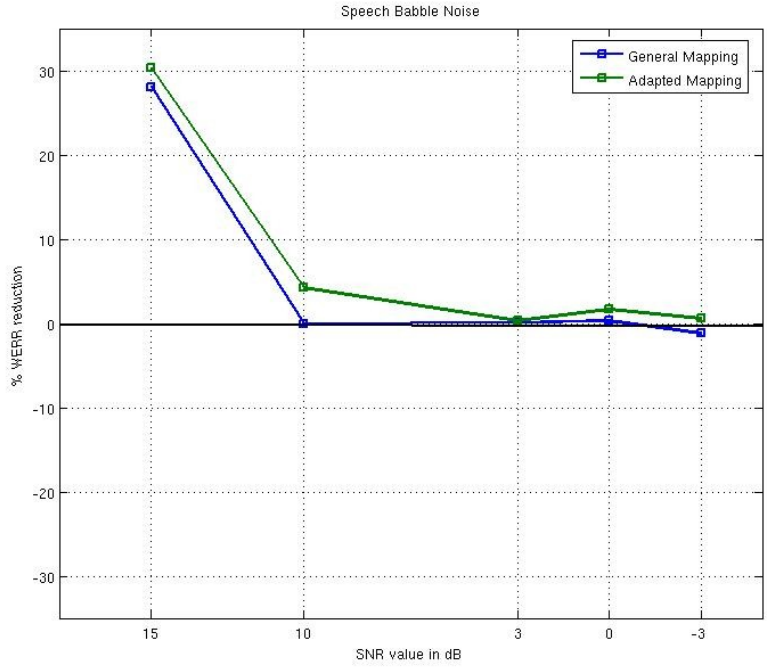


Figure 18(a): *Babble Speech noise results (Audio+Video mapping)*

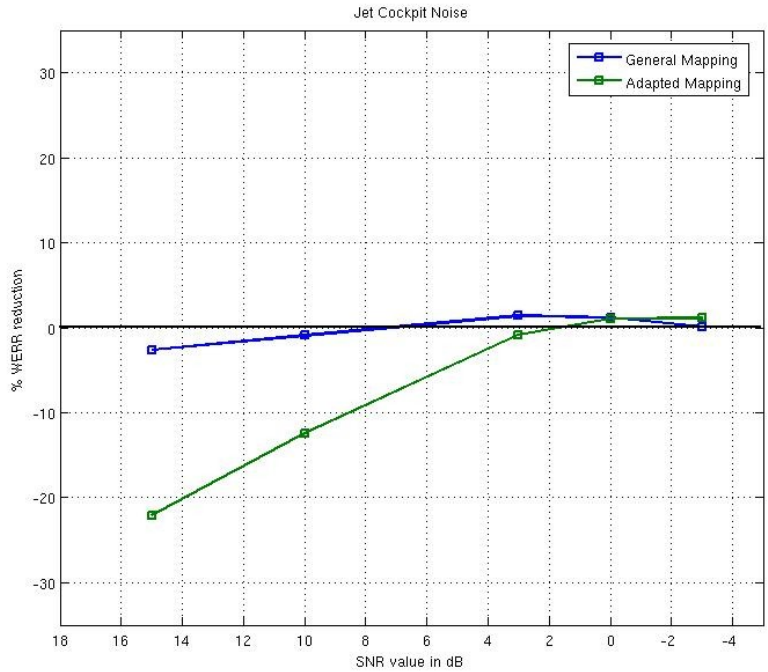


Figure 18(b): *Jet Cockpit noise results (Audio + Video mapping)*

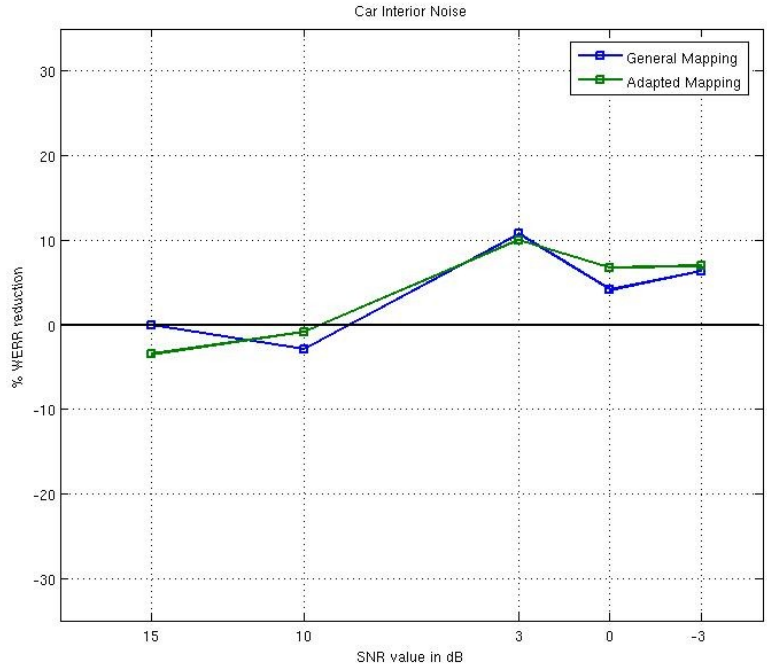


Figure 18(c): *Car Interior noise results (Audio + Video mapping)*

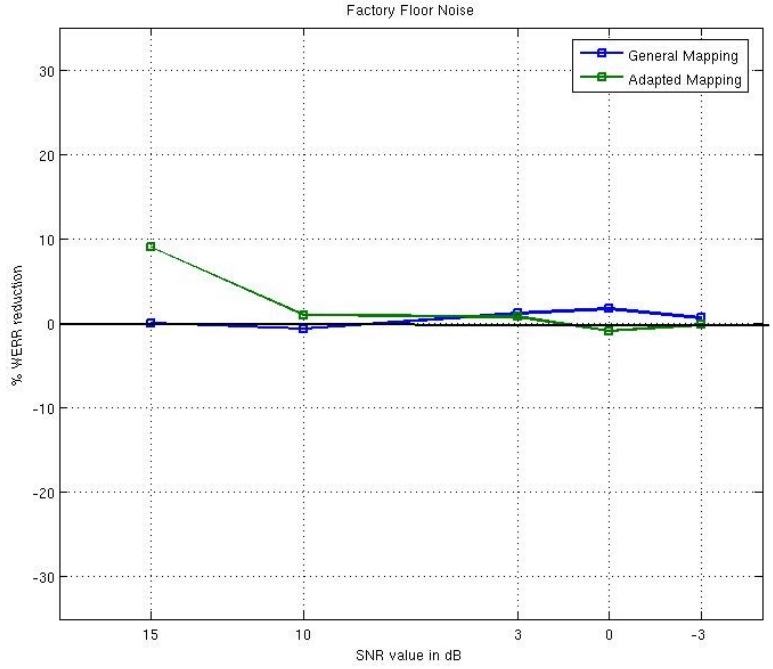


Figure 18(d): *Factory Floor noise results (Audio + Video mapping)*

	Noise type	15 dB	10 dB	3 dB	0 dB	-3 dB
AV-Global Best Values	Babble Speech	0,88	0,86	0,86	0,91	0,83
AV-Instant General Mapping		0,78	0,76	0,73	0,72	0,72
AV-Instant Adapted Mapping		0,91	0,9	0,89	0,88	0,88
AV-Global Best Values	Jet Cockpit noise	0,82	0,79	0,69	0,58	0,62
AV-Instant General Mapping		0,75	0,73	0,71	0,71	0,71
AV-Instant Adapted Mapping		0,67	0,64	0,61	0,61	0,62
AV-Global Best Values	Car Interior noise	0,92	0,91	0,84	0,78	0,74
AV-Instant General Mapping		0,81	0,81	0,8	0,8	0,8
AV-Instant Adapted Mapping		0,78	0,78	0,78	0,78	0,78
AV-Global Best Values	Factory Floor noise	0,86	0,92	0,83	0,74	0,59
AV-Instant General Mapping		0,76	0,74	0,71	0,71	0,7
AV-Instant Adapted Mapping		0,87	0,86	0,85	0,84	0,84

Table 17: *Stream weight comparison for different AV methods*

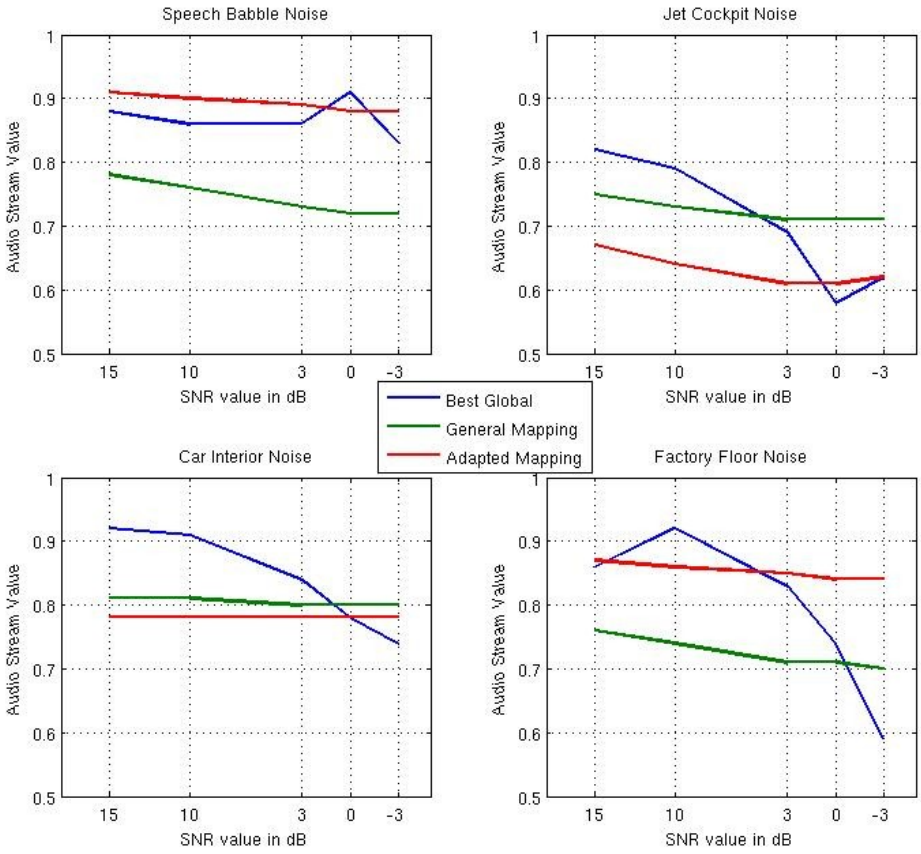


Figure 19: *Audio stream weight graphs for different AV Methods (Audio + Video mapping)*

Conclusions – Future Work CHAPTER 6

In this project we researched the Audio-Visual approach to speech recognition. The approach is based on the concept that audio and visual information of speech can be combined to develop a robust ASR system. The information can be extracted from Video data signals recorded during speech. The respective audio and visual signals can be regarded as information streams, so multiple information stream decision fusion methods can be used for the recognition. Different fusion methods were described. A stream weighting scheme was introduced during fusion to give appropriate biasing to each stream while making the final decision. The problem of stream weight computation was described and different methods based on a posteriori likelihoods of separate classifiers were given.

Then we created a recognition system setup to test the theoretical concepts we introduced above. First a method using the reliability indicator of audio stream to compute the stream weights was created. This method was tested using either a general all-noise-condition mapping or an adapted to noise condition mapping. Then a method using the reliability indicators of audio and video streams was created. Again a general all-noise-condition and an adapted to noise condition mapping were tested. The results were compared to the following recognizers: (a) Audio-only, (b) Video-only, (c) AV with fixed stream weights during recognition process and (d) AV with best pre-computed time-constant stream weights for each noise condition. The results showed that obvious improvement was achieved compared to (a) Audio-only recognizer and (c) AV with fixed stream weights during recognition process. Also the methods achieved similar

behaviour – sometimes superior - to the (d) AV with best pre-computed time-constant stream weights for each noise condition, both to the final Accuracy and actual stream weight values (mean values in the case of time variant).

The strong point of the method is the ability to estimate time-variant stream weights adapting to variable environmental conditions. The tests we conducted simulated different environmental noise conditions, but these were static during each recognition task (type of noise and SNR). Tests conducted under varying noise conditions should show in more detail the inherent abilities of the methods tested.

Also improving the visual stream recognition would give an overall boost to the accuracy of the AV-ASR system. Applying more sophisticated methods in pre-processing and ROI acquiring could help. Feature extraction and selection of the visual signal can be improved.

Additionally, some tweaking in the mapping of the reliabilities to the weights of streams could give some improvement. Also using alternative methods of computing the stream weights could be tested.

Finally a system could be set up, that would research the AV methods in a sub-word (phoneme-level) recognition process.

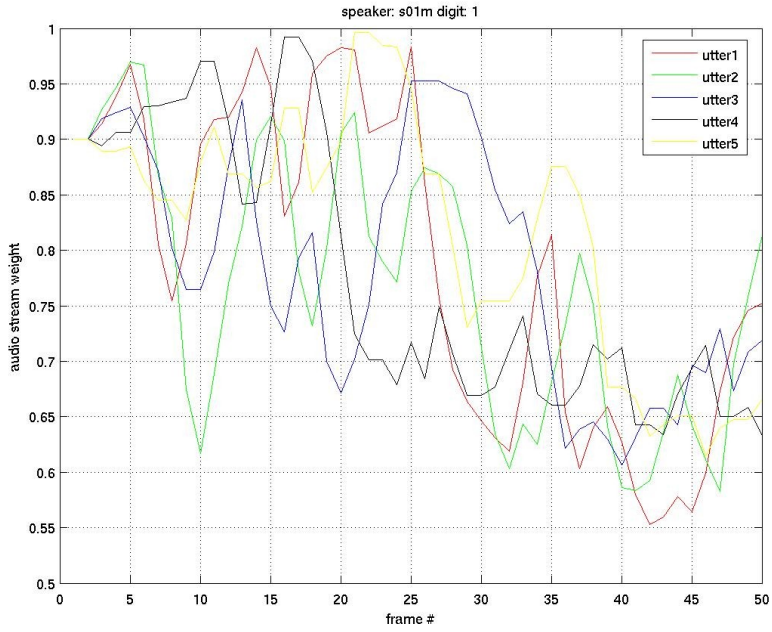
APPENDIX A

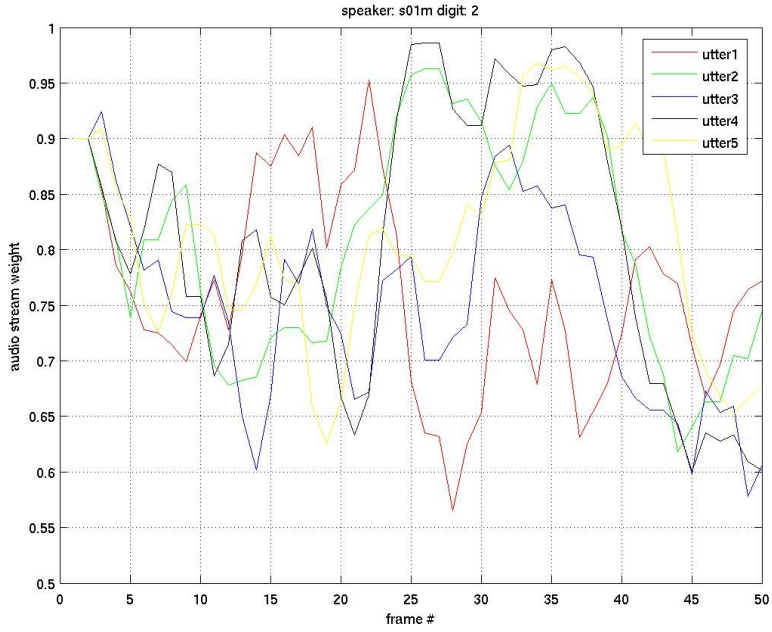
Instantaneous Weight Graphs for every Digit

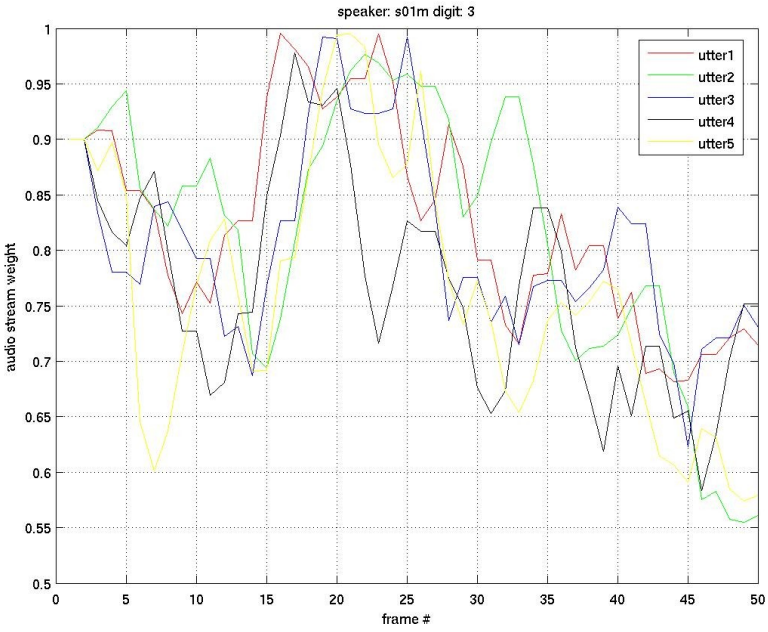
In this appendix we included the graphs of the stream weights that were computed using the method of audio reliability indicator general all-noise mapping to the stream weights. Especially the results of the recognition of speakers 1, 2, 4, 20, 33, 34 under 10 dB factory floor noise uttering 5 times the digits were plotted. For each speaker and each digit, a combined graph was created with the sample axis (x axis) normalized to value 50 with the 5 utterances.

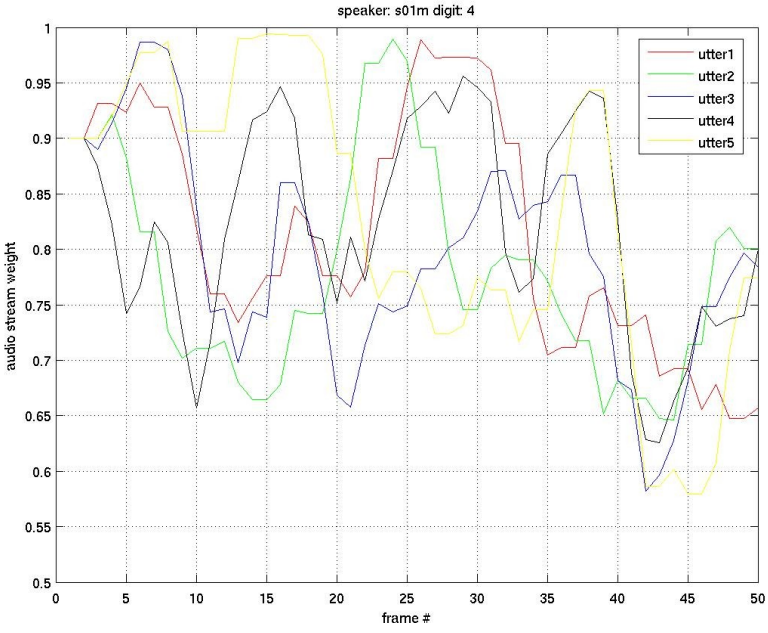
The results show that for the same digits, similar weight graphs can be identified. This makes us believe that there is some dependence of the reliabilities of streams on the specific digit that should be recognized. Some temporal regions in the digit utterance could include more information in audio stream and others less, thus making the video stream more reliable for that region. This can be confirmed when comparing the weights created by the same digit utterances but from different speakers.

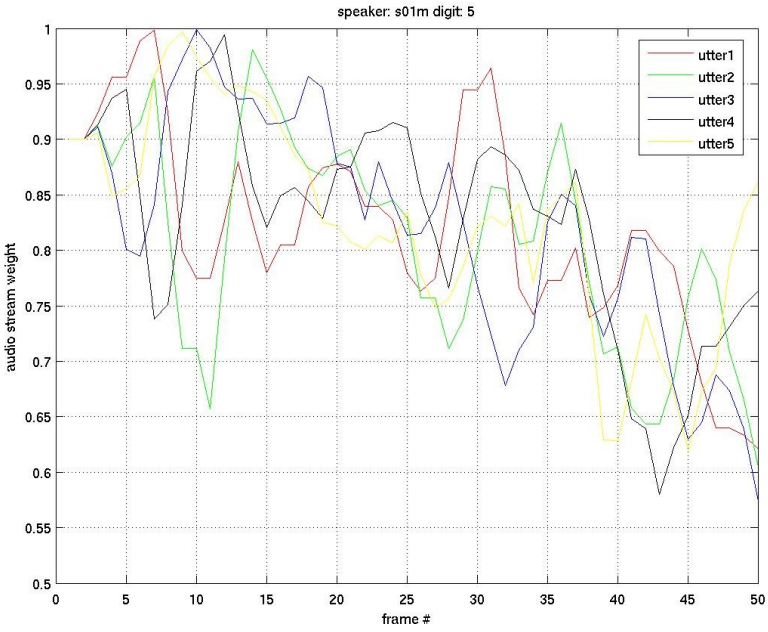
Speaker 1 (male)

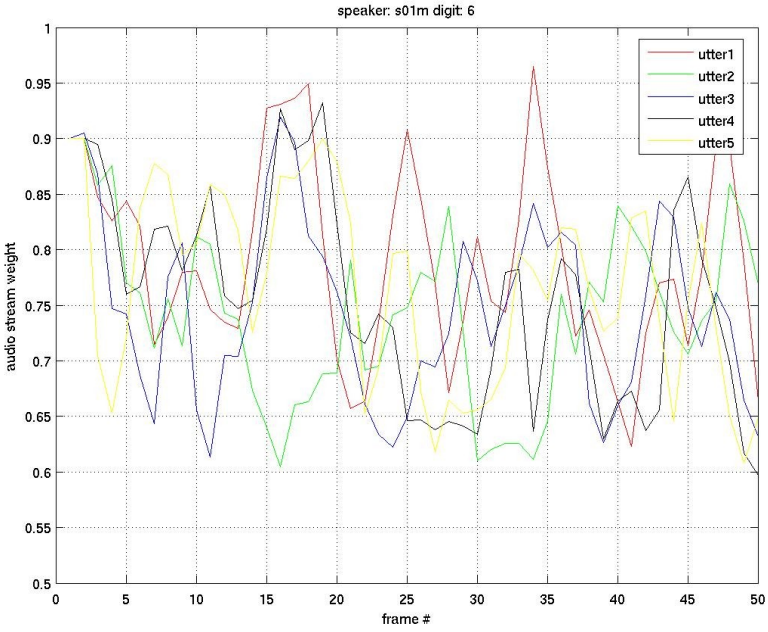


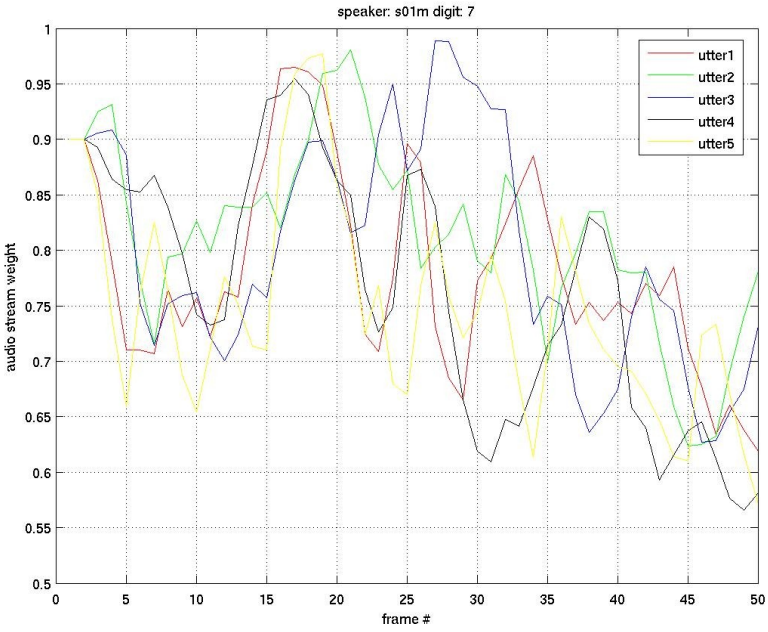


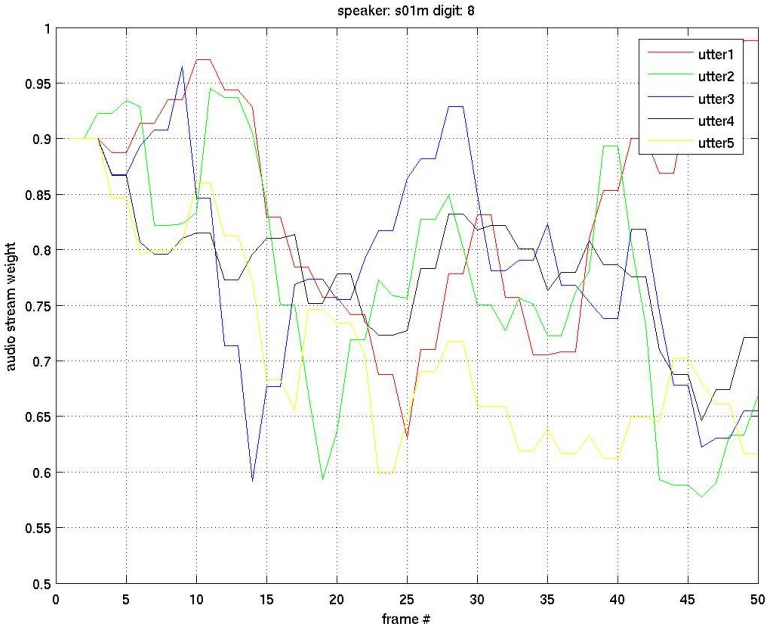

speaker: s01m digit: 3
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5

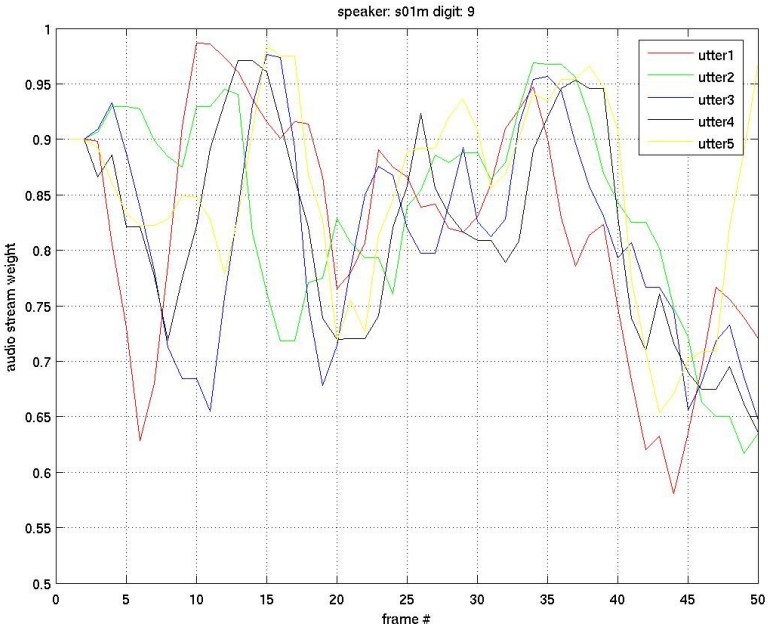

speaker: s01m digit: 4
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5


speaker: s01m digit: 5
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #

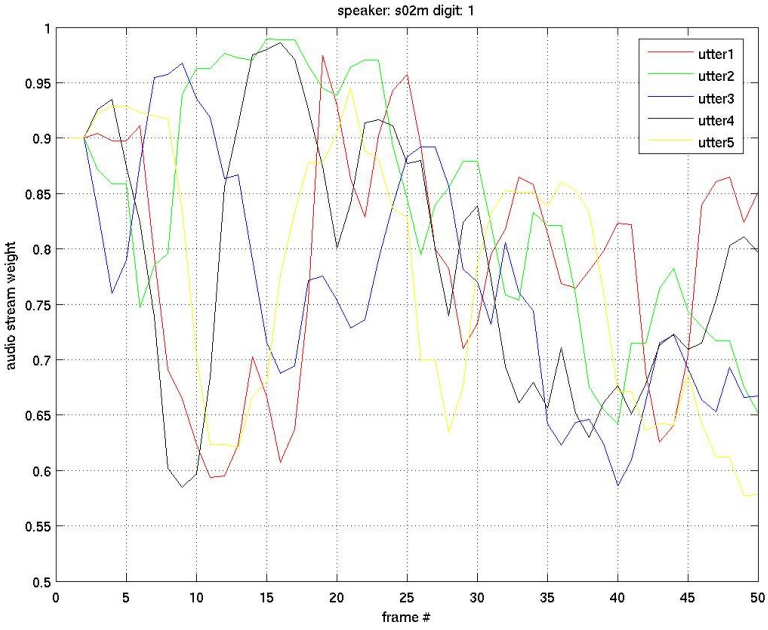

speaker: s01m digit: 6
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #

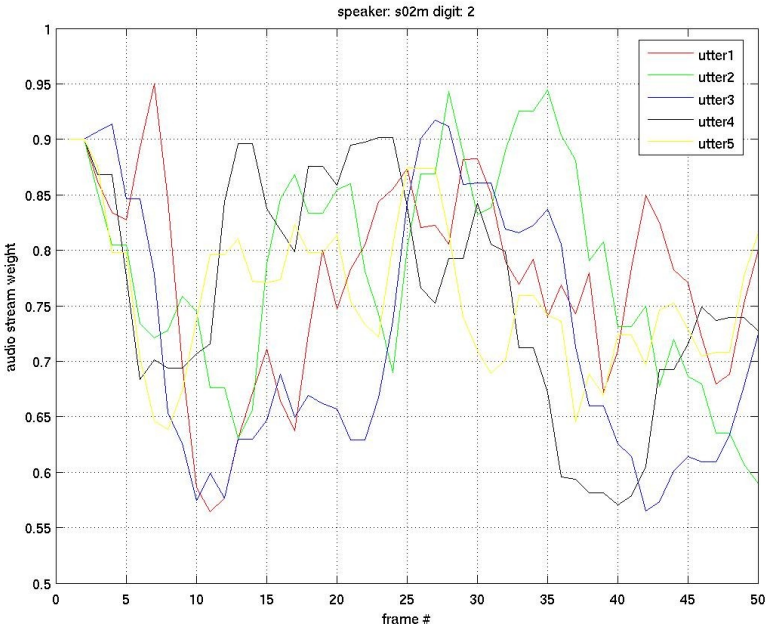

speaker: s01m digit: 7
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5

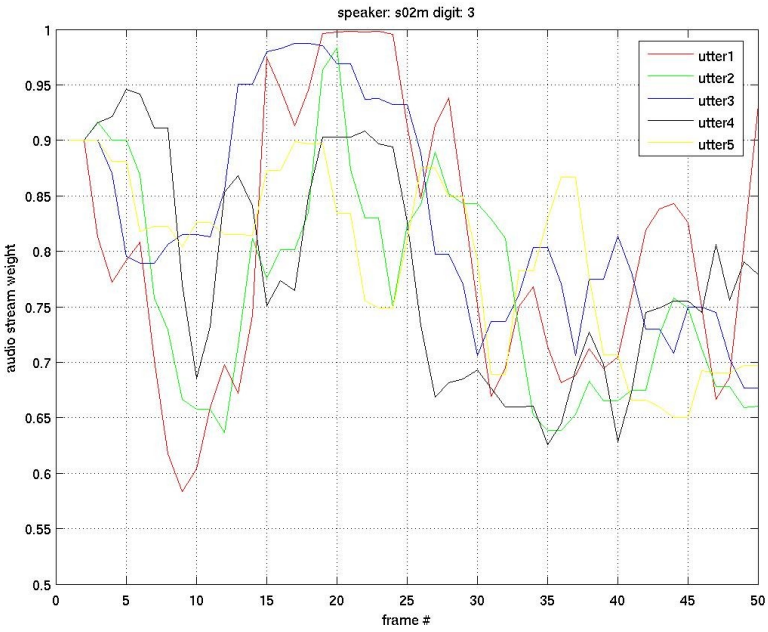

speaker: s01m digit: 8
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5

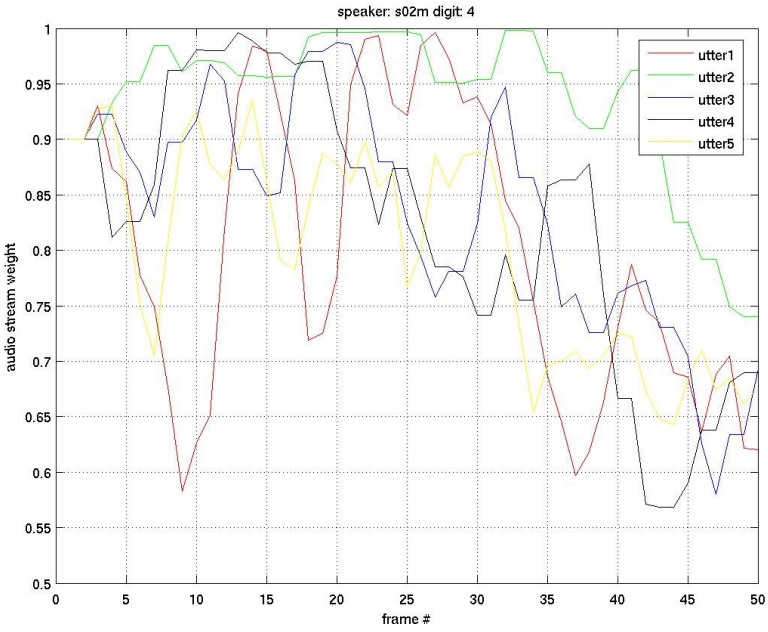


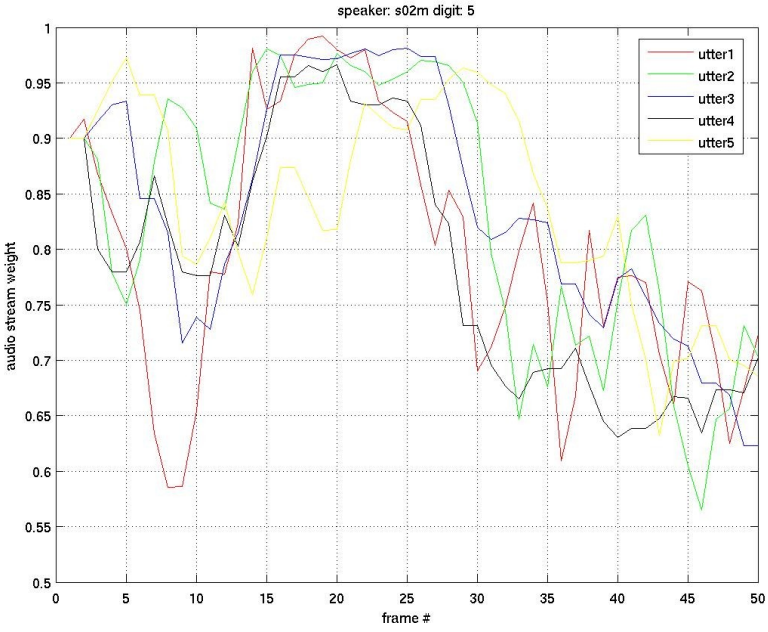
Speaker 2 (male)

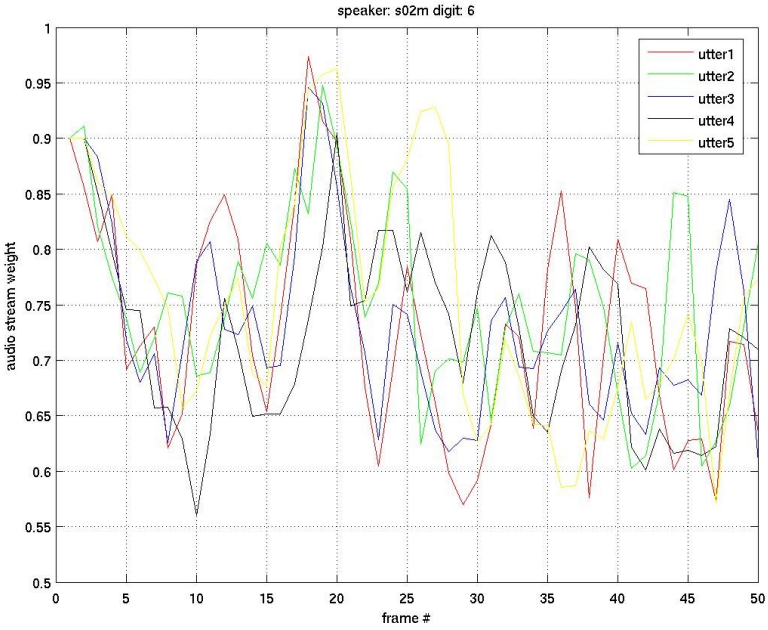


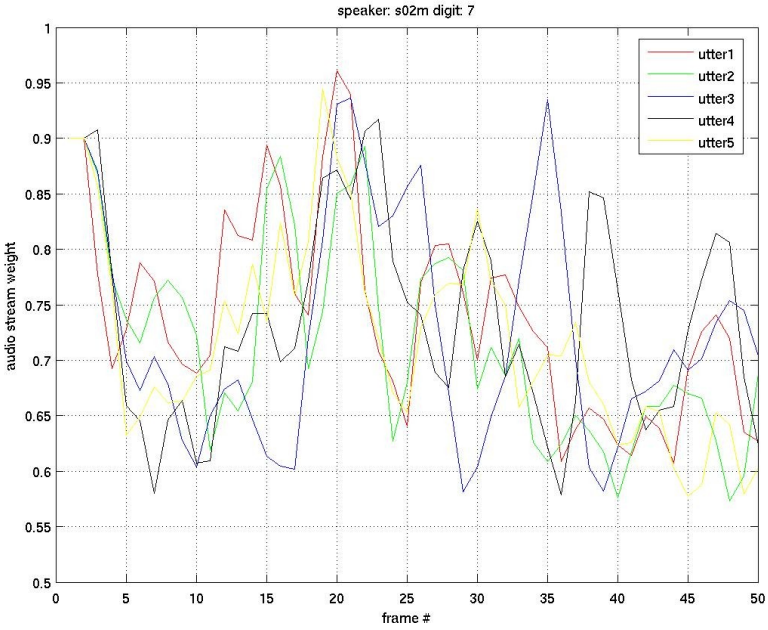

speaker: s02m digit: 2
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #

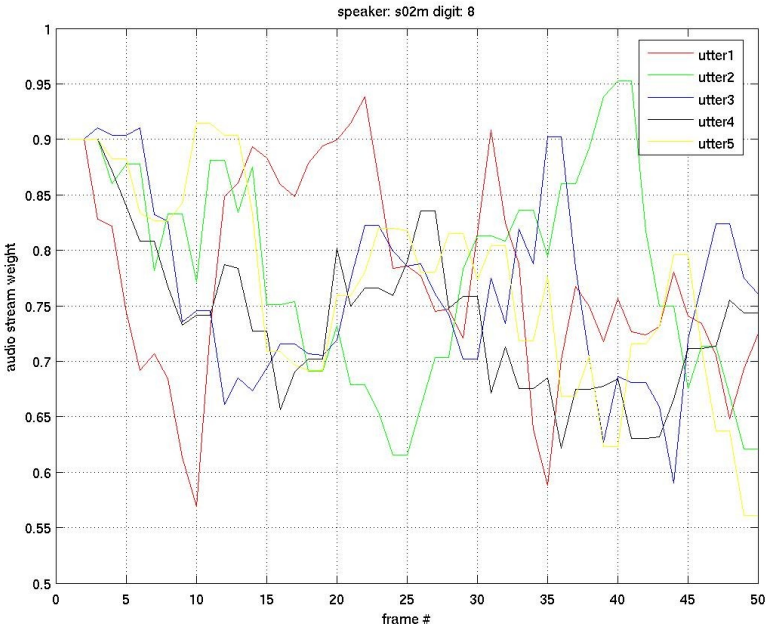

speaker: s02m digit: 3
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #

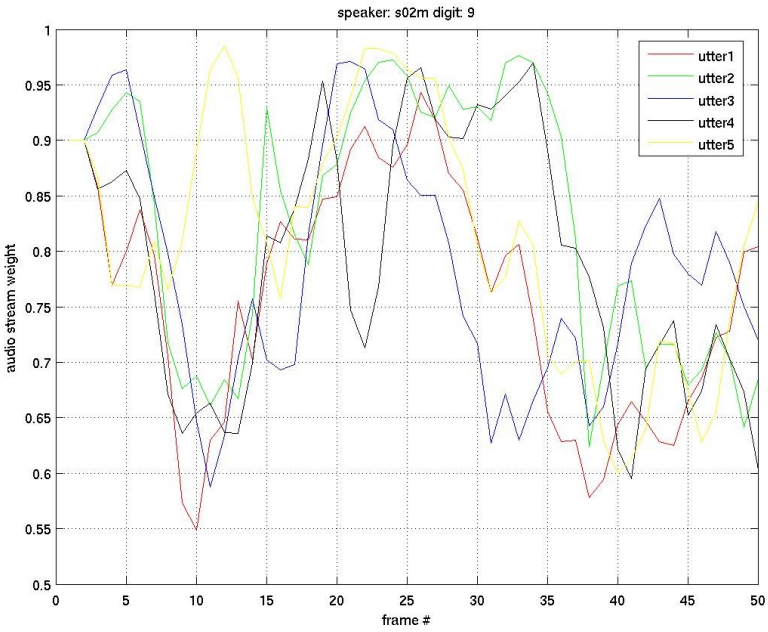

speaker: s02m digit: 4
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5


speaker: s02m digit: 5
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5

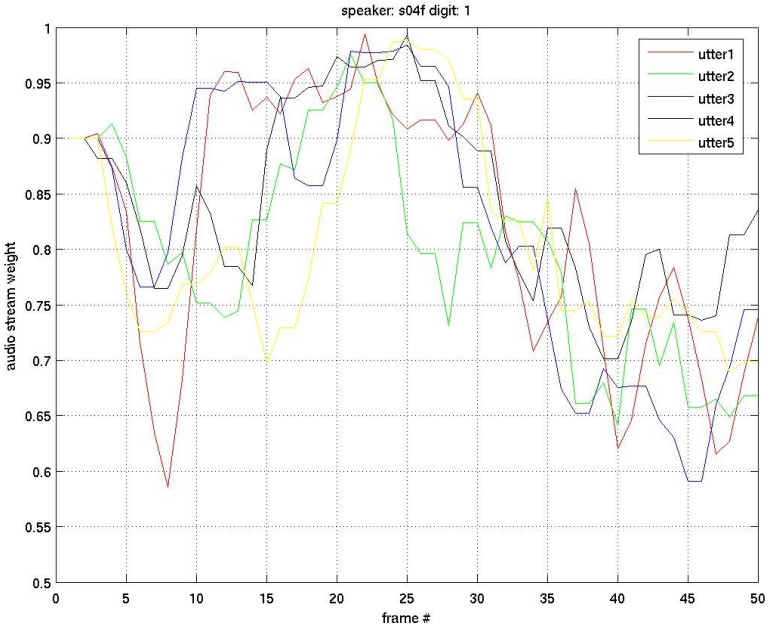

speaker: s02m digit: 6
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

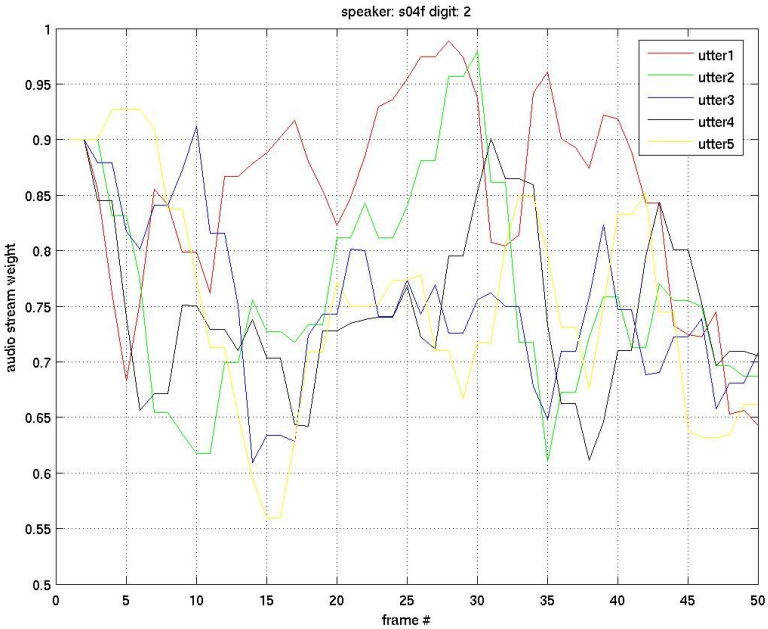

speaker: s02m digit: 7
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

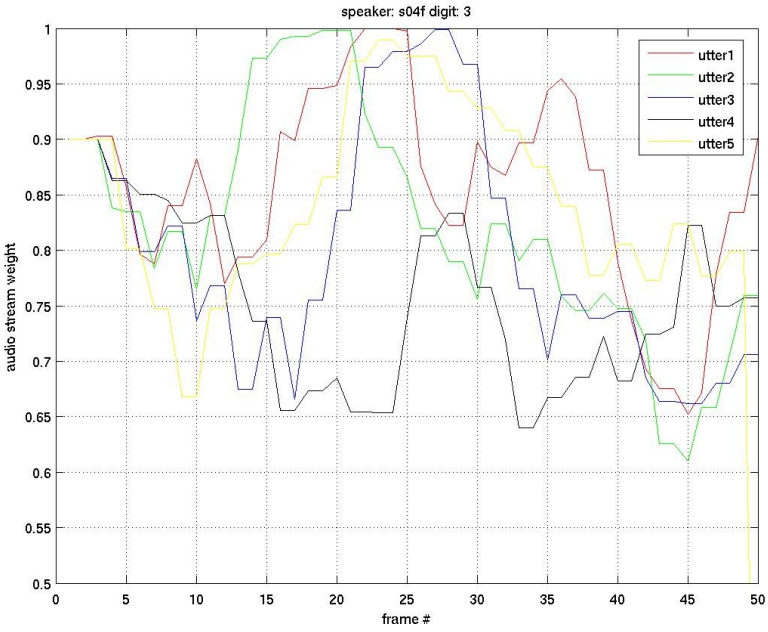

speaker: s02m digit: 8
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #

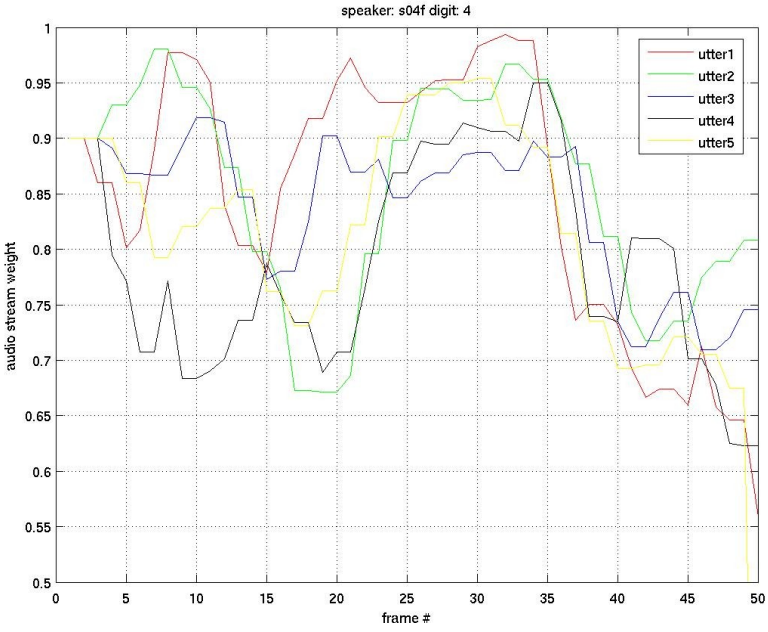

speaker: s02m digit: 9
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #

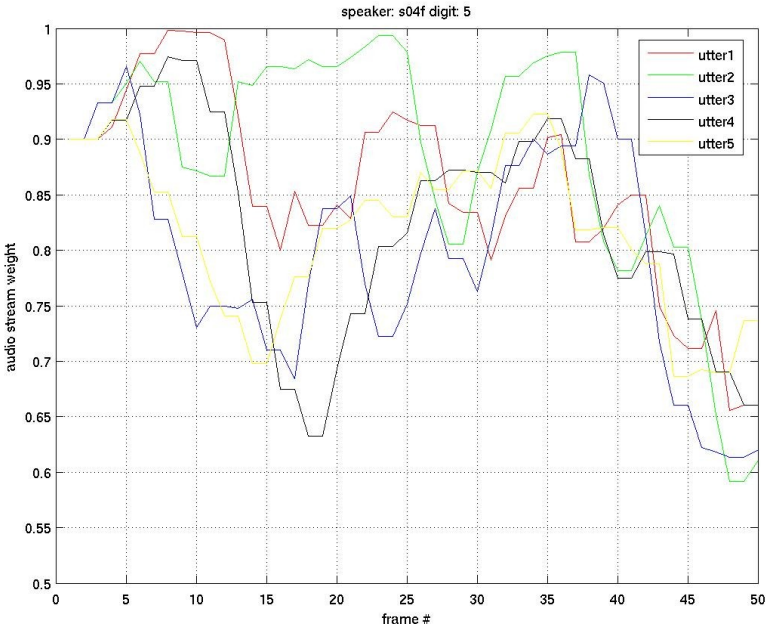
Speaker 4 (female)

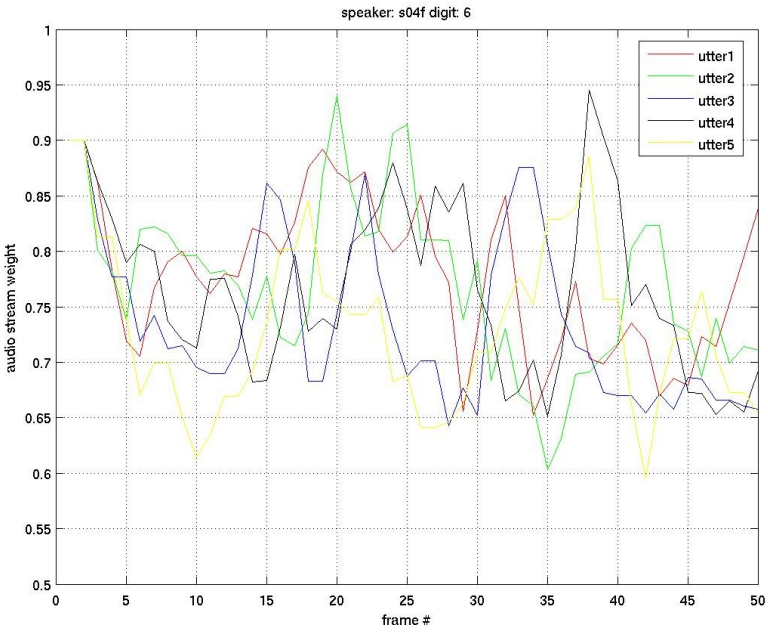


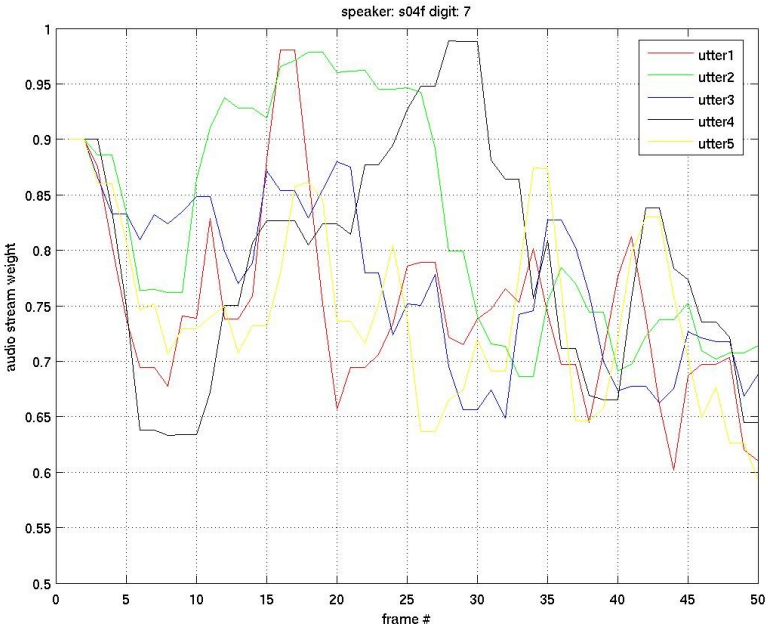


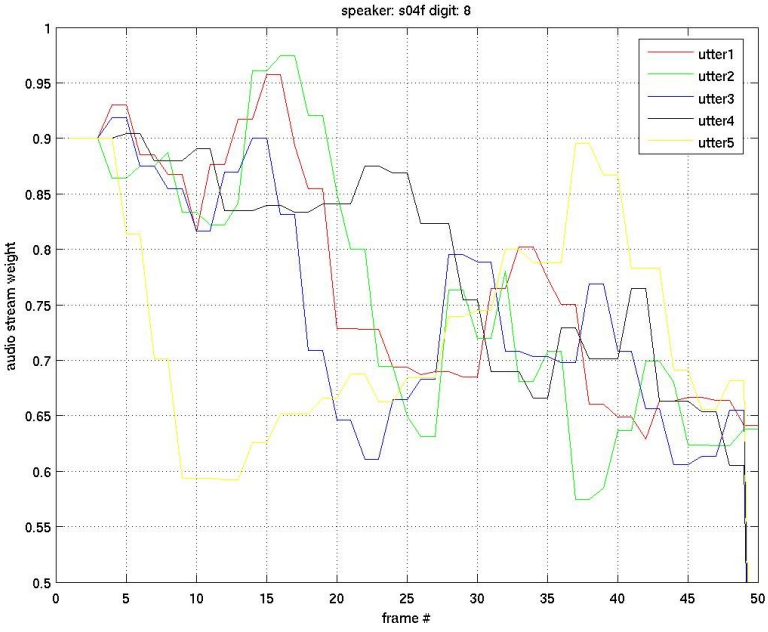

speaker: s04f digit: 3
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5

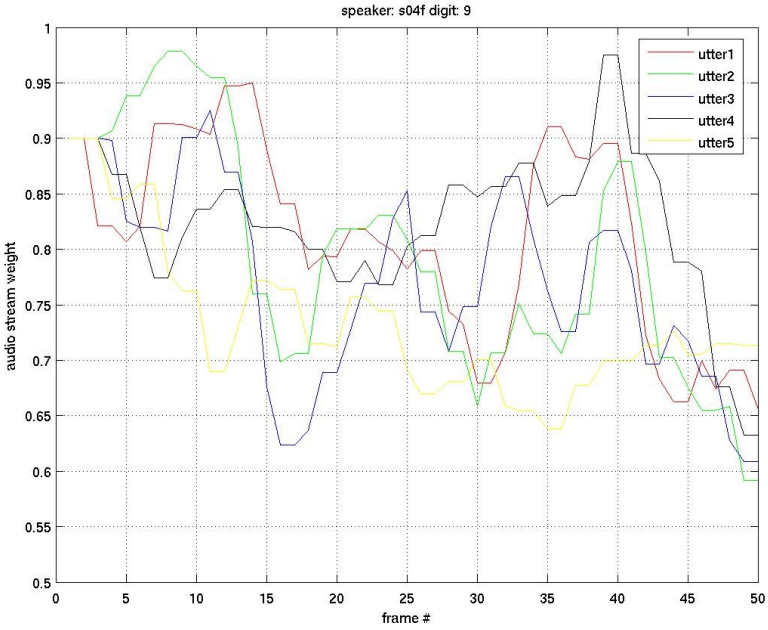

speaker: s04f digit: 4
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5


speaker: s04f digit: 5
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5

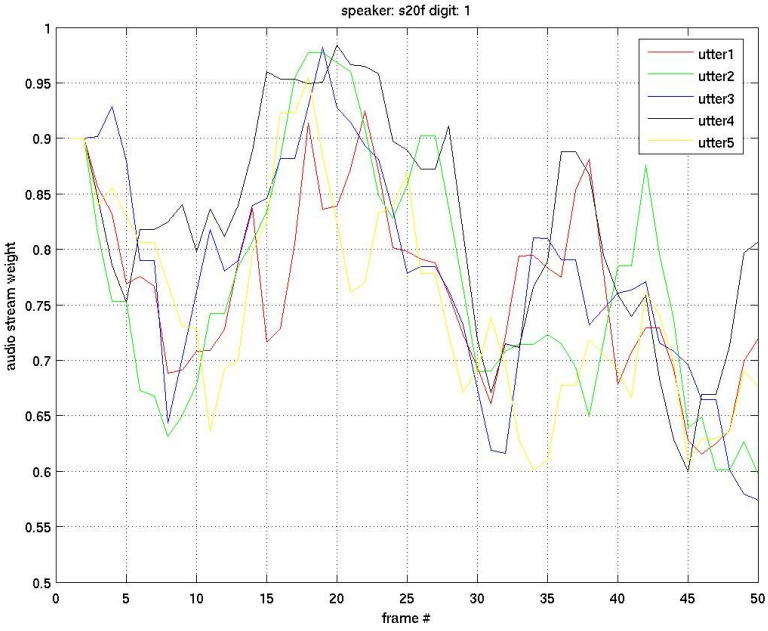

speaker: s04f digit: 6
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5

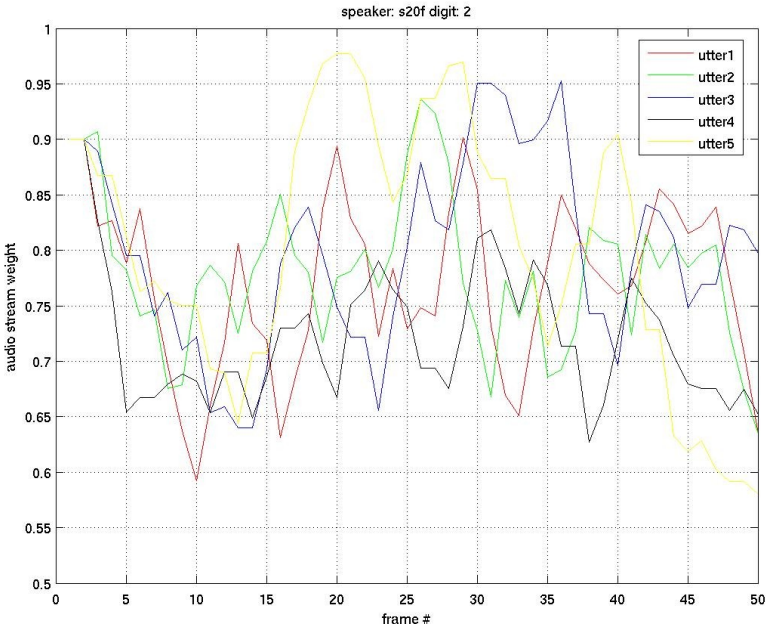

speaker: s04f digit: 7
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5

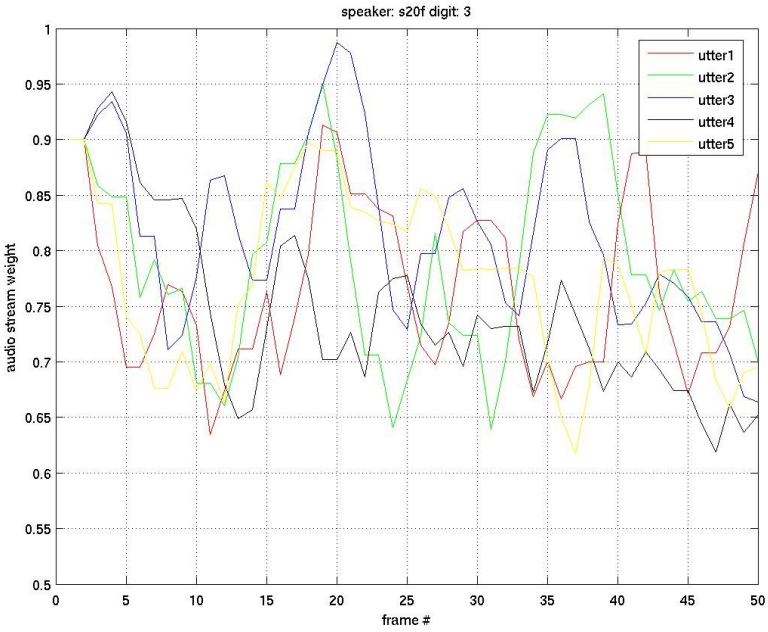

speaker: s04f digit: 8
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5

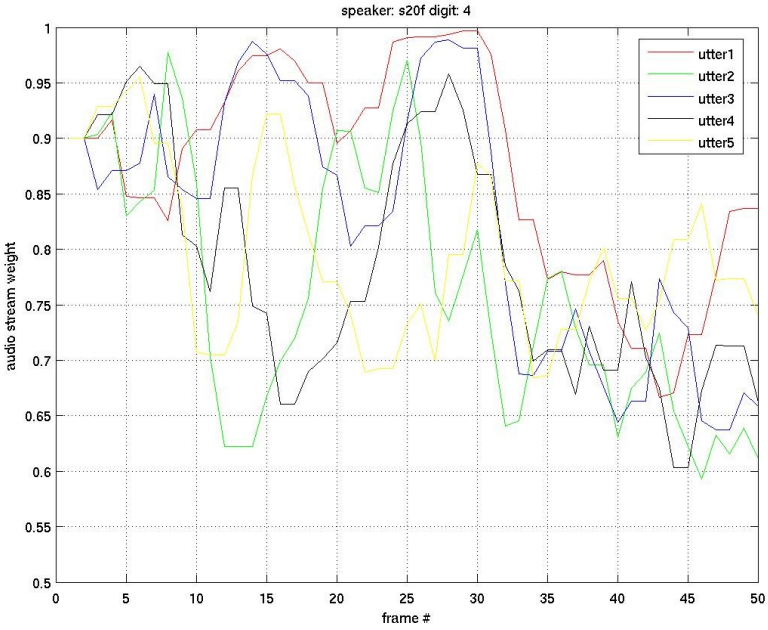


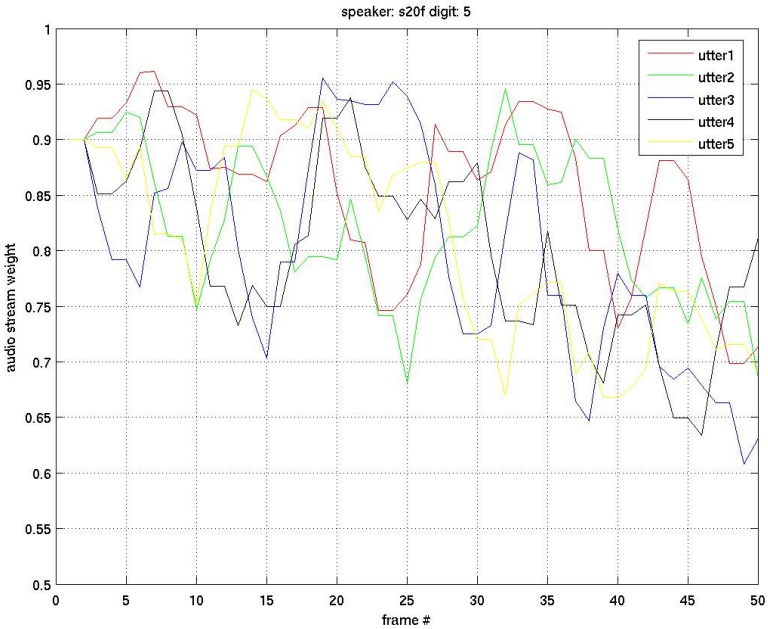
Speaker 20 (female)

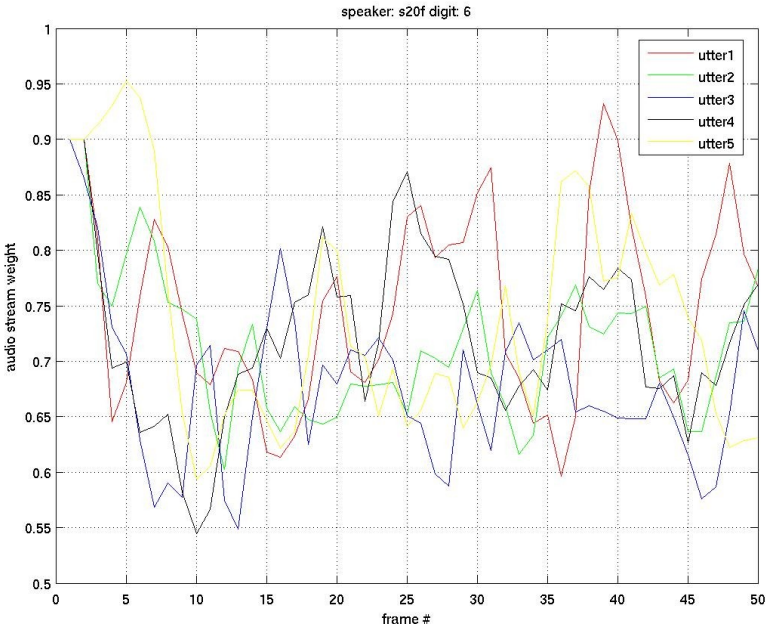


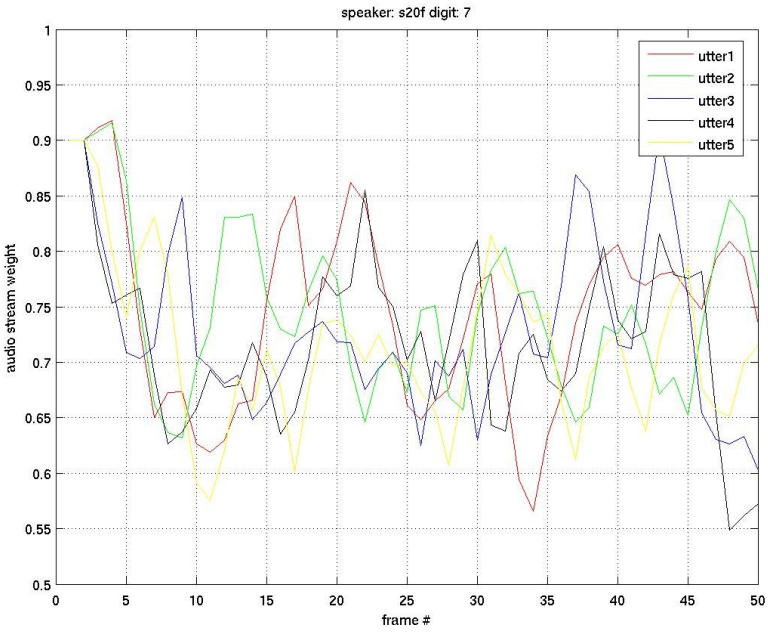

speaker: s20f digit: 2
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #

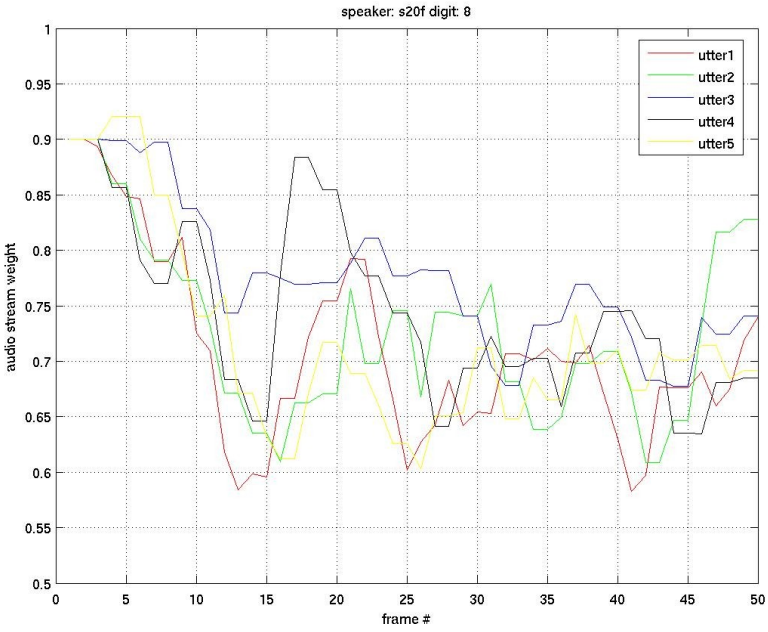

speaker: s20f digit: 3
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #

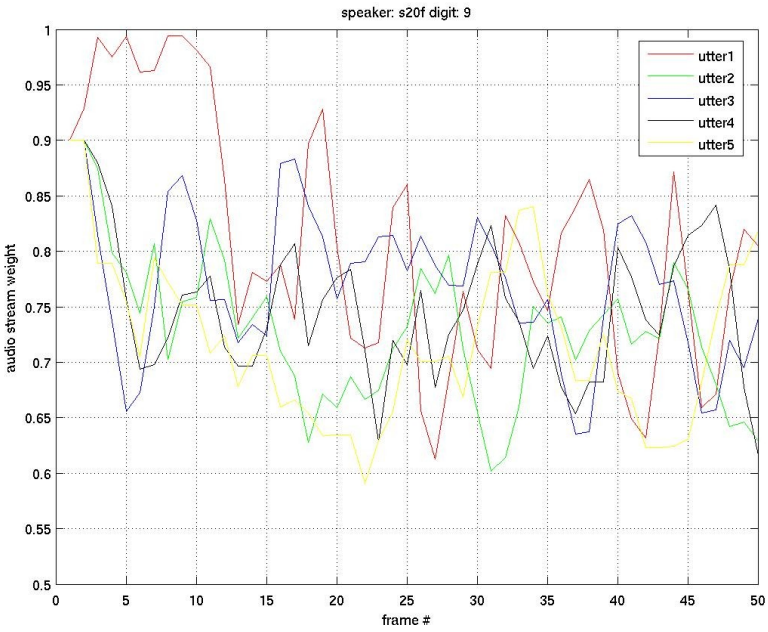

speaker: s20f digit: 4
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50


speaker: s20f digit: 5
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

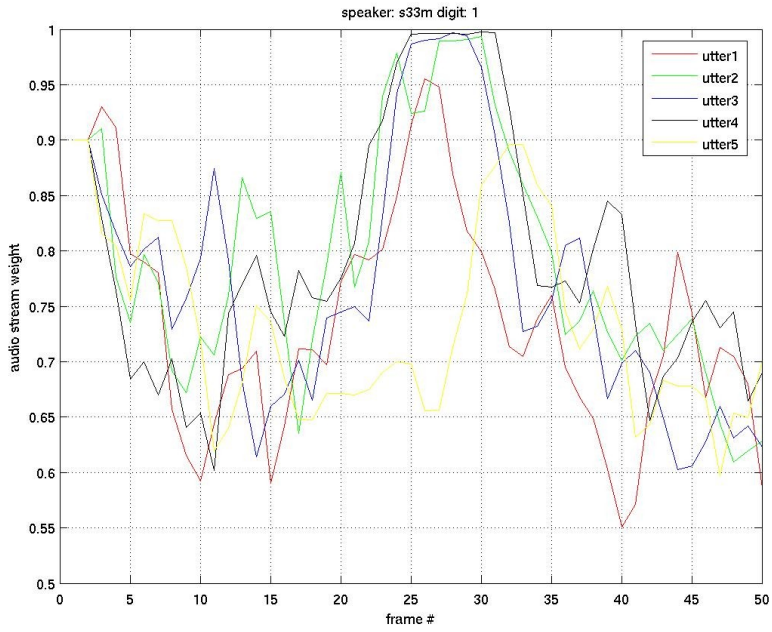

speaker: s20f digit: 6
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

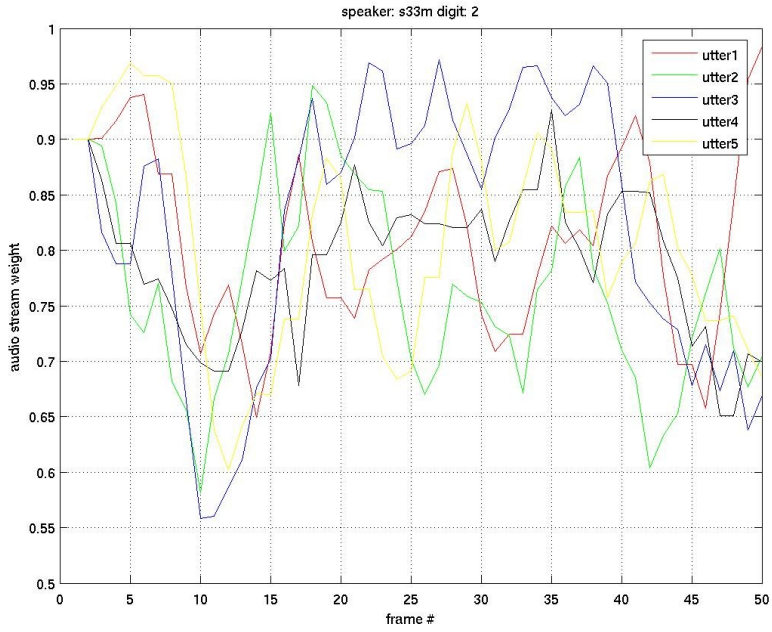

speaker: s20f digit: 7
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

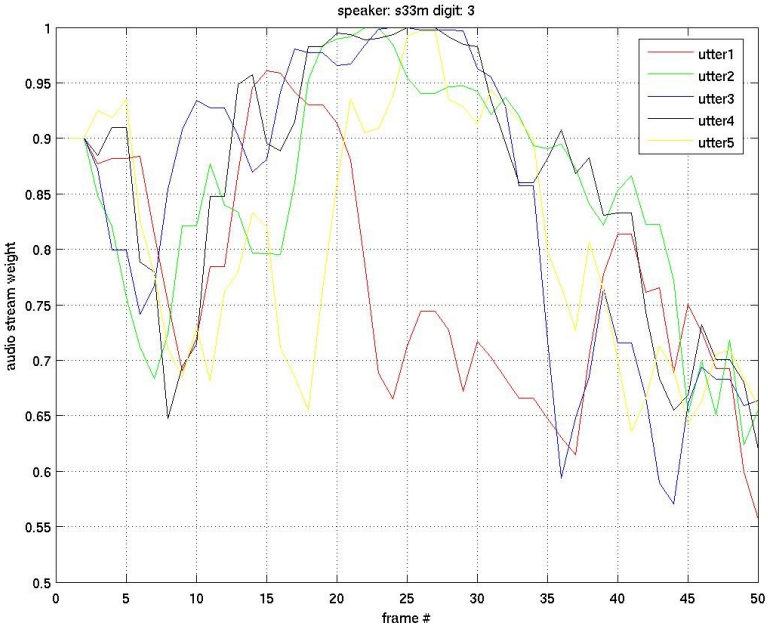

speaker: s20f digit: 8
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5

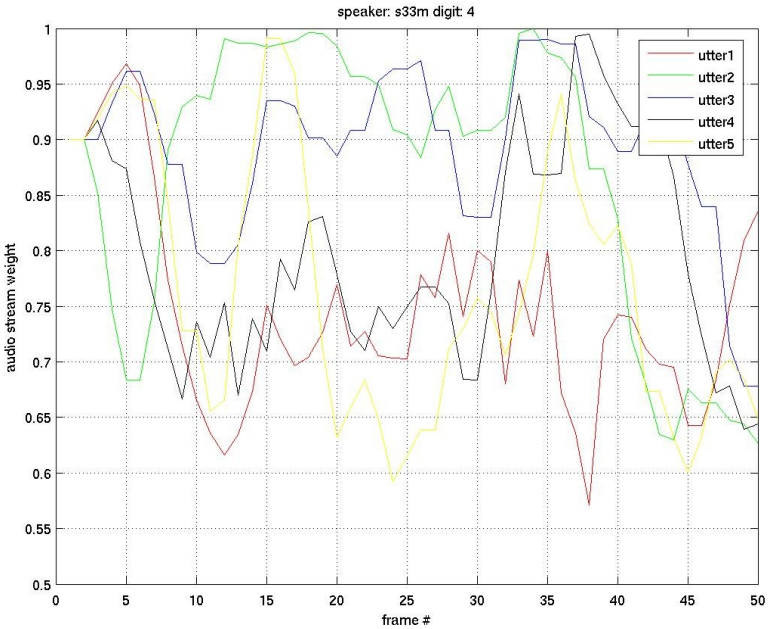

speaker: s20f digit: 9
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5

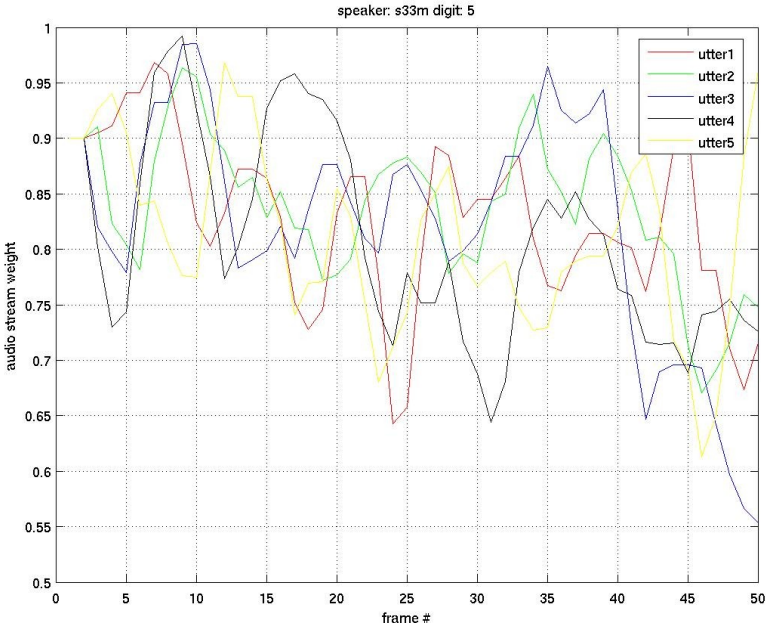
Speaker 33 (male)

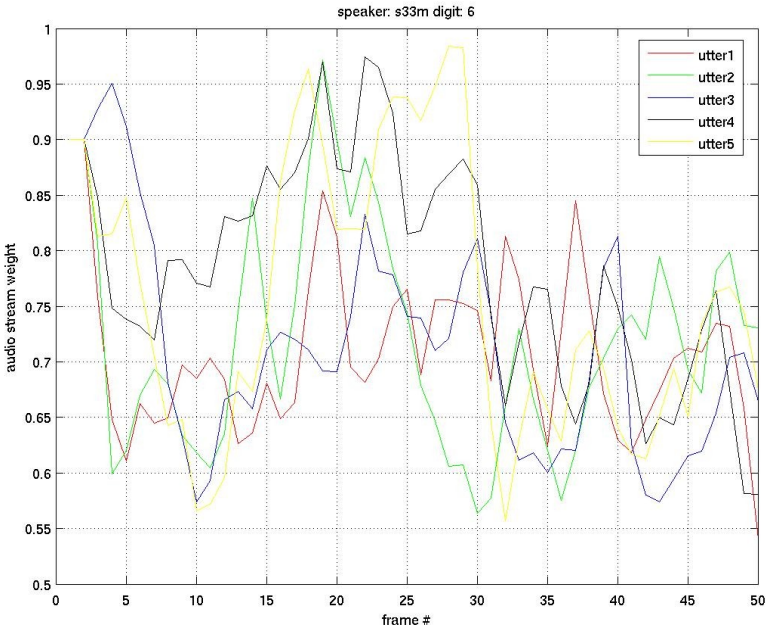


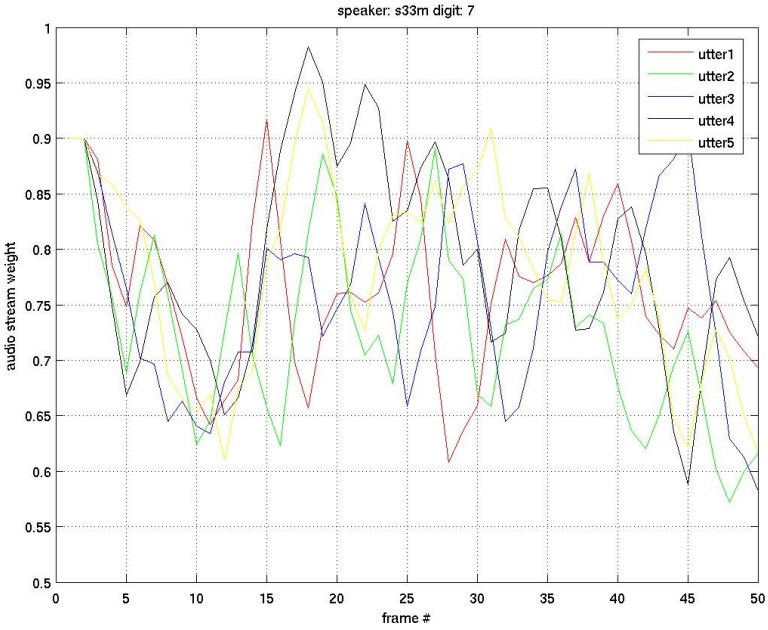


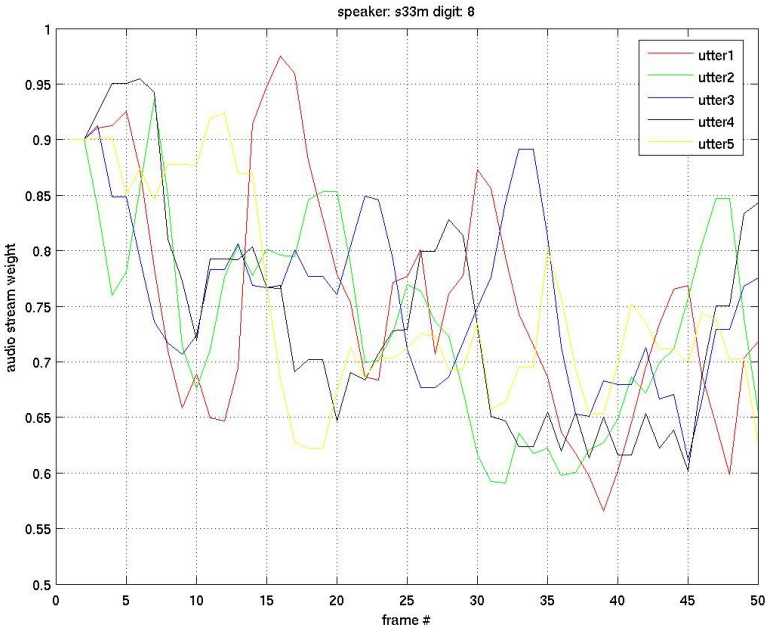

speaker: s33m digit: 3
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

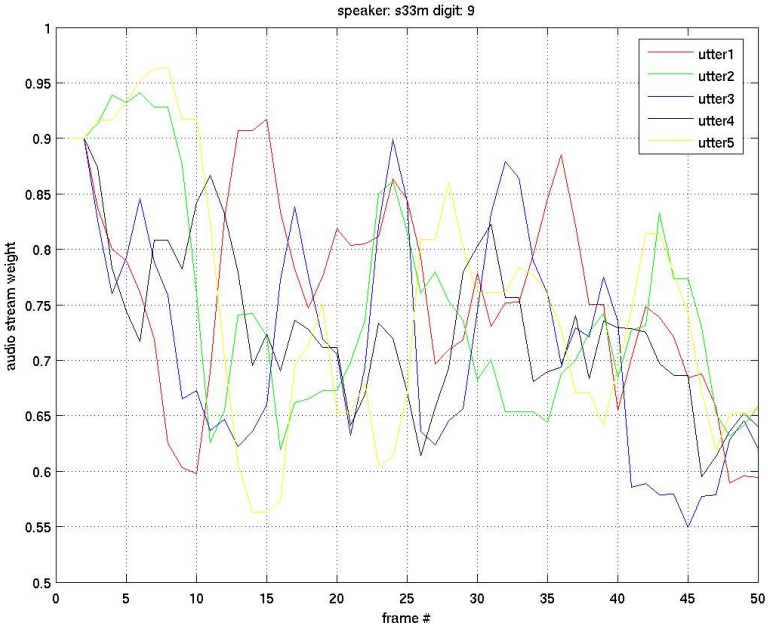

speaker: s33m digit: 4
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50


speaker: s33m digit: 5
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

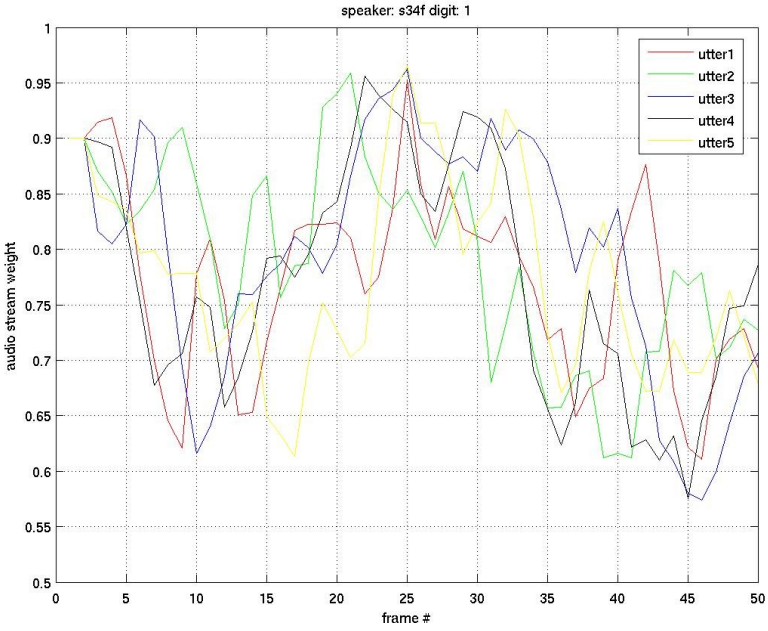

speaker: s33m digit: 6
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

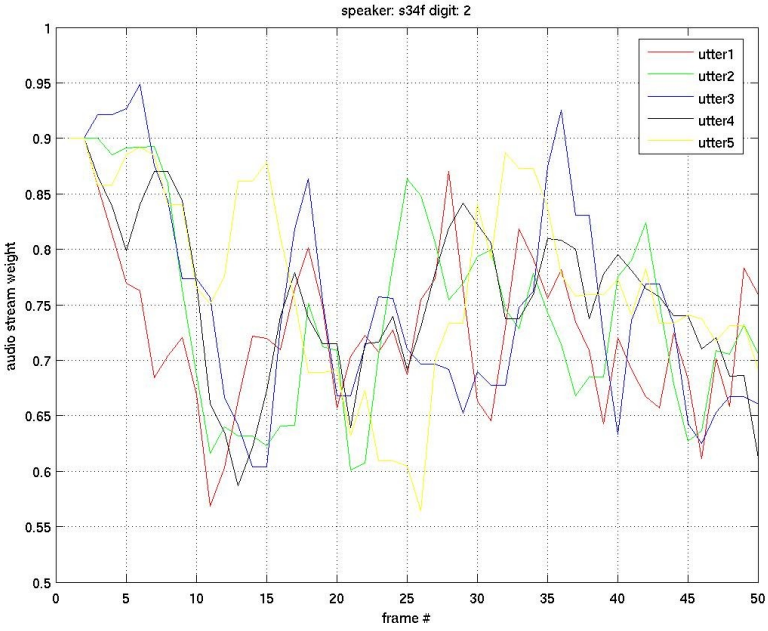

speaker: s33m digit: 7
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

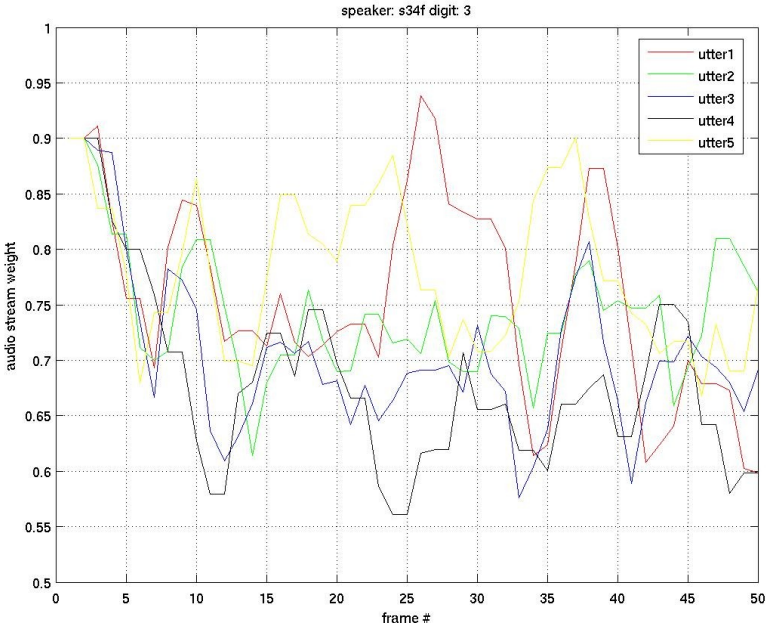

speaker: s33m digit: 8
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

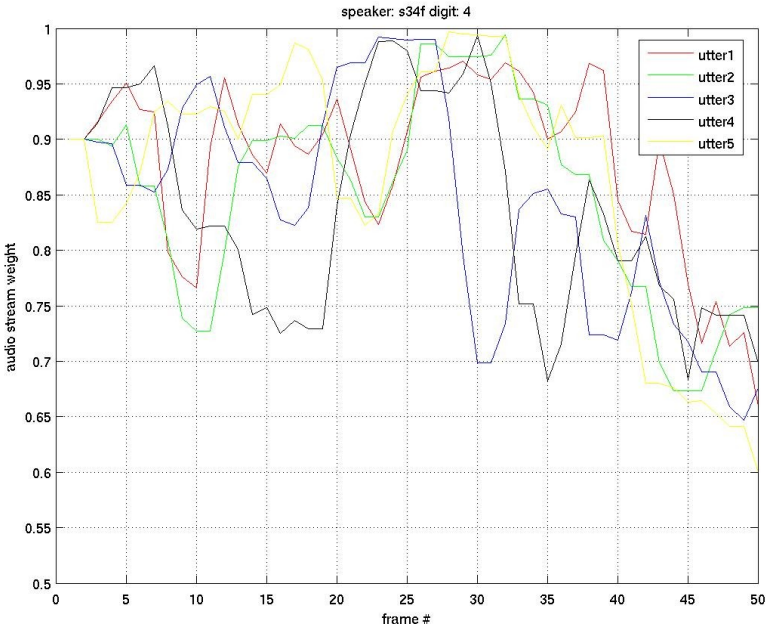


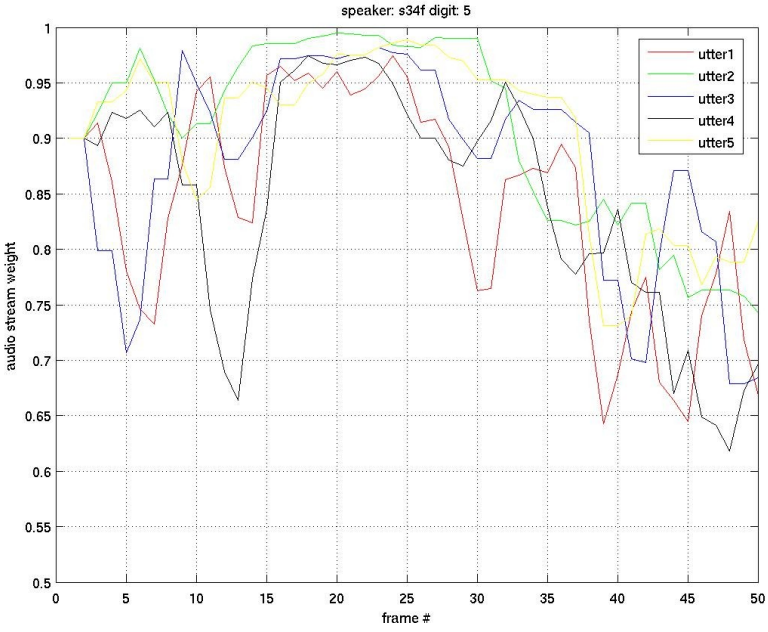
Speaker 34 (female)

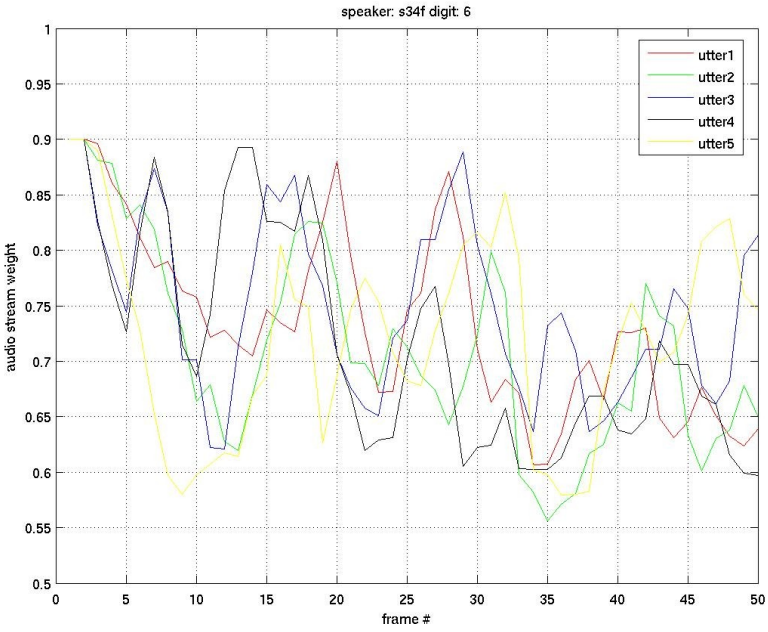


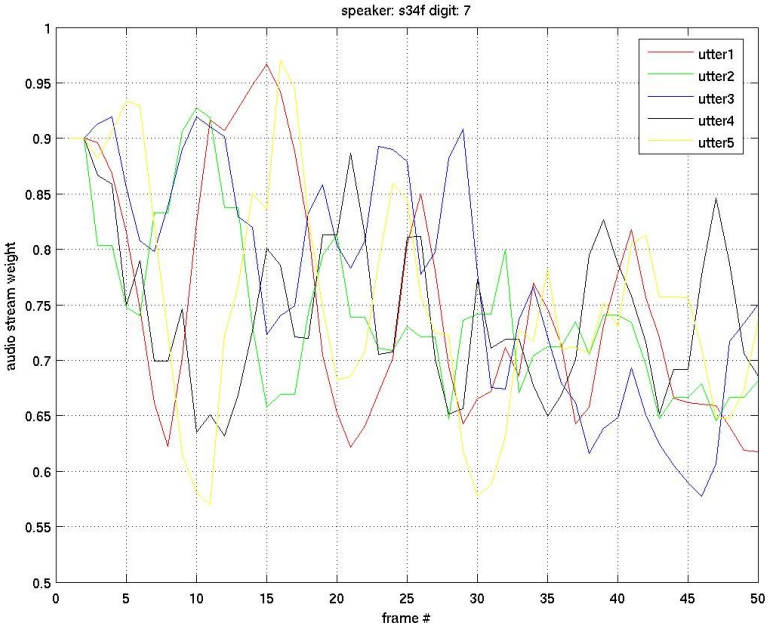

speaker: s34f digit: 2
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

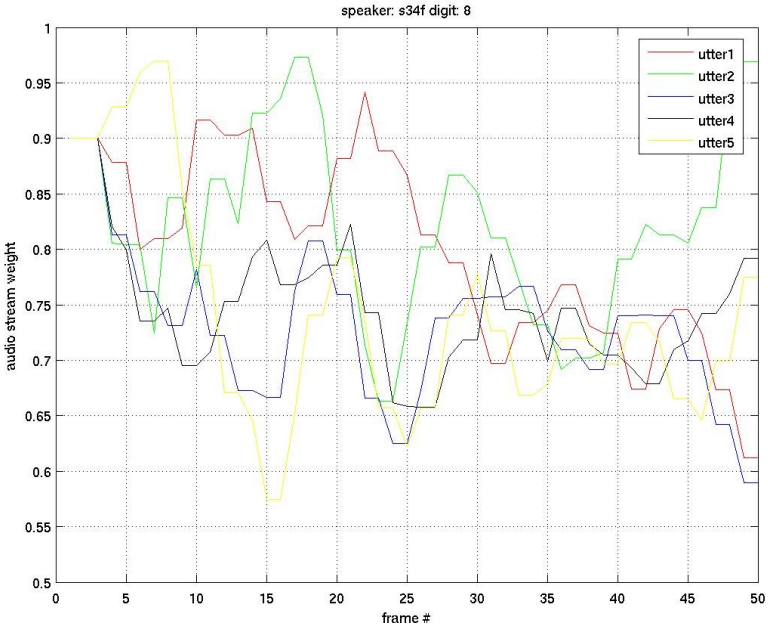

speaker: s34f digit: 3
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

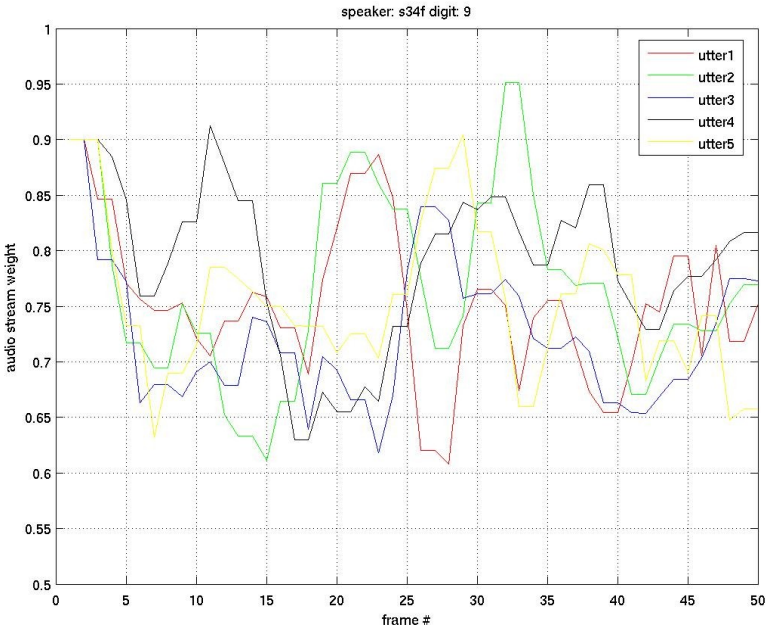

speaker: s34f digit: 4
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5


speaker: s34f digit: 5
audio stream weight
frame #
utter1
utter2
utter3
utter4
utter5


speaker: s34f digit: 6
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #


speaker: s34f digit: 7
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #


speaker: s34f digit: 8
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50


speaker: s34f digit: 9
utter1
utter2
utter3
utter4
utter5
audio stream weight
frame #
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
5
10
15
20
25
30
35
40
45
50

APPENDIX B

HTK source code modifications

B.1 Instantaneous stream weight support

To support the time variant stream weights as described in 2.2 and 2.4, two files were modified: Hvite.c and Hrec.c.

Hvite.c source code modifications:

First we declared some global variables that would be useful for the implementation

```
static char *wgt_datFN;              /* Weights file */
static Observation wgt_obs;          /* weight observation */
static MemHeap bufHeap_WGT;
```

Then inside the Initialise() function we make the appropriate initialisation for the stream weights file

```
wgt_obs=MakeObservation(&gstack,hset.swidth,hset.pkind,
                        hset.hsKind==DISCRETEHS,eSep);

CreateHeap(&bufHeap_WGT,"Input Buffer heap
          WGT",MSTAK,1,0.0,50000,50000);
```

Then the function ProcessFile() is substituted by the modified function ProcessFileWGT() which is as follows

```
/* ProcessFile: process given file. If fn=NULL then direct audio */
Boolean ProcessFileWGT(char *fn, Network *net, int utterNum, LogDouble
currGenBeam, Boolean restartable)
{
  FILE *file;
  ParmBuf pbuf;
  ParmBuf wgt_pbuf; /* Spiros */
  BufferInfo pbinfo;
```

```
   NetNode *d;
   Lattice *lat;
   LArc *arc,*cur;
   LNode *node;
   Transcription *trans;
   MLink m;
   LogFloat lmlk,aclk;
   int s,j,tact,nFrames;
   LatFormat form;
   char *p,lfn[255],buf1[80],buf2[80],thisFN[MAXSTRLEN];
   Boolean enableOutput = TRUE, isPipe;

   if (fn!=NULL)
      strcpy(thisFN,fn);
   else if (fn==NULL && saveAudioOut)
      CounterFN(roPrefix,roSuffix,++roCounter,4,thisFN);
   else
      enableOutput = FALSE;

   if((pbuf =
OpenBuffer(&bufHeap,fn,50,dfmt,TRI_UNDEF,TRI_UNDEF))==NULL)
      HError(3250,"ProcessFileWGT: Config parameters invalid");

   /* Check pbuf same as hset */
   GetBufferInfo(pbuf,&pbinfo);
   if (pbinfo.tgtPK!=hset.pkind)
      HError(3231,"ProcessFileWGT: Incompatible sample kind %s vs %s",
             ParmKind2Str(pbinfo.tgtPK,buf1),
             ParmKind2Str(hset.pkind,buf2));
   if (pbinfo.a != NULL && replay)  AttachReplayBuf(pbinfo.a, (int)
(3*(1.0E+07/pbinfo.srcSampRate)));

/* Spiros */
       wgt_datFN = GetStrArg();
       if (trace&T_TOP) {
            printf("File: %s\n",wgt_datFN);
              }

   if((wgt_pbuf =
OpenBuffer(&bufHeap_WGT,wgt_datFN,50,dfmt,TRI_UNDEF,TRI_UNDEF))=
=NULL)
      HError(3250,"ProcessFileWGT: Config parameters invalid");

/* Spiros */

   StartRecognition(vri,net,lmScale,wordPen,prScale);
   SetPruningLevels(vri,maxActive,currGenBeam,wordBeam,nBeam,tmBea
m);

   tact=0;nFrames=0;
   StartBuffer(pbuf);
   StartBuffer(wgt_pbuf); /* Spiros*/
   while(BufferStatus(pbuf)!=PB_CLEARED) {
      ReadAsBuffer(pbuf,&obs);
```

```
    ReadAsBuffer(wgt_pbuf,&wgt_obs);  /* Spiros*/
    if (trace&T_OBS) {
            PrintObservation(nFrames,&obs,13);
                PrintObservation(nFrames,&wgt_obs,13); /* Spiros*/
                }

    if (hset.hsKind==DISCRETEHS){
      for (s=1; s<=hset.swidth[0]; s++){
        if( (obs.vq[s] < 1) || (obs.vq[s] > maxMixInS[s]))
          HError(3250,"ProcessFile: Discrete data value [ %d ] out of range
in stream [ %d ] in file %s",obs.vq[s],s,fn);
      }
    }

/*    ProcessObservation(vri,&obs,-1,xfInfo.inXForm);*/
    ProcessObservationWGT(vri,&obs,&wgt_obs,-1,xfInfo.inXForm); /*
Spiros*/

    if (trace & T_FRS) {
      for (d=vri->genMaxNode,j=0;j<30;d=d->links[0].node,j++)
        if (d->type==n_word) break;
      if (d->type==n_word){
        if (d->info.pron==NULL) p=":bound:";
        else p=d->info.pron->word->wordName->name;
      }
      else p=":external:";
      m=FindMacroStruct(&hset,'h',vri->genMaxNode->info.hmm);
      printf("Optimum @%-4d HMM: %s (%s)  %d %5.3f\n",
            vri->frame,m->id->name,p,
            vri->nact,vri->genMaxTok.like/vri->frame);
      fflush(stdout);
    }
    nFrames++;
    tact+=vri->nact;
  }
  lat=CompleteRecognition(vri,pbinfo.tgtSampRate/10000000.0,&ansHea
p);

  if (lat==NULL) {
    if ((trace & T_TOP) && fn != NULL){
      if (restartable)
        printf("No tokens survived to final node of network at beam
%.1f\n", currGenBeam);
      else
        printf("No tokens survived to final node of network\n");
      fflush(stdout);
    } else if (fn==NULL){
      printf("Sorry [%d frames]?\n",nFrames);fflush(stdout);
    }
    if (pbinfo.a != NULL && replay)  ReplayAudio(pbinfo);
    CloseBuffer(pbuf);
    CloseBuffer(wgt_pbuf); /* Spiros*/
    return FALSE;
  }
```

```
  if (vri->noTokenSurvived && restartable)
    return FALSE;

  if (vri->noTokenSurvived && trace & T_TOP) {
    printf("No tokens survived to final node of network\n");
    printf("  Output most likely partial hypothesis within network\n");
    fflush(stdout);
  }

  lat->utterance=thisFN;
  lat->net=wdNetFn;
  lat->vocab=dictFn;

  if (trace & T_TOP || fn==NULL) {
    node=NULL;
    for (j=0;j<lat->nn;j++) {
      node=lat->lnodes+j;
      if (node->pred==NULL) break;
      node=NULL;
    }
    aclk=lmlk=0.0;
    while(node!=NULL) {
      for (arc=NULL,cur=node->foll;cur!=NULL;cur=cur->farc) arc=cur;
      if (arc==NULL) break;
      if (arc->end->word!=NULL)
        printf("%s ",arc->end->word->wordName->name);
      aclk+=arc->aclike+arc->prlike*lat->prscale;
      lmlk+=arc->lmlike*lat->lmscale+lat->wdpenalty;
      node=arc->end;
    }
    printf(" ==  [%d frames] %.4f [Ac=%.1f LM=%.1f]
(Act=%.1f)\n",nFrames,
           (aclk+lmlk)/nFrames, aclk,lmlk,(float)tact/nFrames);
    fflush(stdout);
  }
  if (pbinfo.a != NULL && replay)  ReplayAudio(pbinfo);

  /* accumulate stats for online unsupervised adaptation
    only if a token survived */
  if ((lat != NULL) &&  (!vri->noTokenSurvived) && ((update > 0) ||
(xfInfo.useOutXForm)))
    DoOnlineAdaptation(lat, pbuf, nFrames);

  if (enableOutput){
    if (nToks>1 && latExt!=NULL) {
      MakeFN(thisFN,labDir,latExt,lfn);
      if ((file=FOpen(lfn,NetOFilter,&isPipe))==NULL)
        HError(3211,"ProcessFile: Could not open file %s for lattice
output",lfn);
      if (latForm==NULL)
        form=HLAT_DEFAULT;
      else {
        for (p=latForm,form=0;*p!=0;p++) {
```

```
            switch (*p) {
            case 'A': form|=HLAT_ALABS; break;
            case 'B': form|=HLAT_LBIN; break;
            case 't': form|=HLAT_TIMES; break;
            case 'v': form|=HLAT_PRON; break;
            case 'a': form|=HLAT_ACLIKE; break;
            case 'l': form|=HLAT_LMLIKE; break;
            case 'd': form|=HLAT_ALIGN; break;
            case 'm': form|=HLAT_ALDUR; break;
            case 'n': form|=HLAT_ALLIKE; break;
            case 'r': form|=HLAT_PRLIKE; break;
            }
          }
        }
        if(WriteLattice(lat,file,form)<SUCCESS)
          HError(3214,"ProcessFile: WriteLattice failed");

        FClose(file,isPipe);
      }

      trans=TranscriptionFromLattice(&ansHeap,lat,nTrans);

      if (labForm!=NULL)
        FormatTranscription(trans,pbinfo.tgtSampRate,states,models,
                            strchr(labForm,'X')!=NULL,
                            strchr(labForm,'N')!=NULL,strchr(labForm,'S')!=NULL,
                            strchr(labForm,'C')!=NULL,strchr(labForm,'T')!=NULL,
                            strchr(labForm,'W')!=NULL,strchr(labForm,'M')!=NULL);

      MakeFN(thisFN,labDir,labExt,lfn);
      /* if(LSave(lfn,trans,ofmt)<SUCCESS)
        HError(3214,"ProcessFile: Cannot save file %s", lfn); */
      LSave(lfn,trans,ofmt);
      Dispose(&ansHeap,trans);
    }
    Dispose(&ansHeap,lat);
    CloseBuffer(pbuf);
    CloseBuffer(wgt_pbuf); /* Spiros*/
    if (trace & T_MMU){
      printf("Memory State after utter %d\n",utterNum);
      PrintAllHeapStats();
    }

    return !vri->noTokenSurvived;
}
```

Now in function DoRecognition() we also modify the call to ProcessFile() to ProcessFileWGT()

```
ProcessFileWGT(datFN,net,n++,genBeam,FALSE);
```

HRec.c source code modifications:

Some modification to Hrec.c were also necessary.

CPOutP() function was substituted with CPOutPWGT() as follows

```
/* Version of POutP that caches outp values with frame id */
static LogFloat cPOutPWGT(PSetInfo *psi,Observation *obs,Observation
*wgt_obs,StateInfo *si,int id)
{
  PreComp *pre;
  LogFloat outp;
  StreamElem *se;
  Vector w;
  Vector wgt_v; /* Spiros */
  int s,S;

 /* printf("Hello from cPoutP\n");*/

  if (si->sIdx>0 && si->sIdx<=pri->psi->nsp)
    pre=pri->psi->sPre+si->sIdx;
  else pre=NULL;

#ifdef SANITY
  if (pre==NULL)
    HError(8520,"cPOutP: State has no PreComp attached");
#endif

  wgt_v=wgt_obs->fv[1]; /*Spiros*/

  if (pre->id!=id) { /* bodged at the moment - fix !! */
    if ((FALSE && psi->mixShared==FALSE) || (psi->hset->hsKind ==
DISCRETEHS)) {
      outp=POutP(psi->hset,obs,si);
    }
    else {
      S=obs->swidth[0];
      if (S==1 && si->weights==NULL){
        outp=cSOutP(psi->hset,1,obs,si->pdf+1,id);
      }
      else {
      /* Spiros - Edw tha prepei na oristei sto w ta weights pou yparxoun
sto observation */
        outp=0.0;
        se=si->pdf+1;
        w=si->weights;
        for (s=1;s<=S;s++,se++){
          outp+=wgt_v[s]*cSOutP(psi->hset,s,obs,se,id);
        }
        /**********************************************************************
**********/
      }
    }
    pre->outp=outp;
    pre->id=id;
```

```
  }
  return(pre->outp);
}
```

Also StepHMM1() function was substituted with modified version StepHMM1WGT().

```
static void StepHMM1WGT(NetNode *node,Observation * wgt_obs) /* Model
internal propagation NBEST */
{
  NetInst *inst;
  HMMDef *hmm;
  Token tok,max;
  TokenSet *res,cmp,*cur;
  Align *align;
  int i,j,k,N,endi;
  LogFloat outp;
  Matrix trP;
  short **seIndex;

  inst=node->inst;
  max=null_token;

  hmm=node->info.hmm;
  N=hmm->numStates;
  trP=hmm->transP;
  seIndex=pri->psi->seIndexes[hmm->tIdx];

  for (j=2,res=pri->psi->sBuf+2;j<N;j++,res++) {  /* Emitting states first
*/
    i=seIndex[j][0];
    endi=seIndex[j][1];
    cur=inst->state+i-1;

    res->tok=cur->tok; res->n=cur->n;
    for (k=0;k<cur->n;k++) res->set[k]=cur->set[k];

    res->tok.like+=trP[i][j];

    for (i++,cur++;i<=endi;i++,cur++) {
      cmp.tok=cur->tok;
      cmp.tok.like+=trP[i][j];
      if (res->n==0) {
        if (cmp.tok.like > res->tok.like)
          res->tok=cmp.tok;
      }
      else
        TokSetMerge(res,&cmp.tok,cur);
    }
    if (res->tok.like>pri->genThresh) { /* State pruning */
```

```
    /* Spiros - Perasma tou weight observation stin synartisi */
/*        outp=cPOutP(pri->psi,pri->obs,hmm->svec[j].info,pri->id);*/
       outp=cPOutPWGT(pri->psi,pri->obs,wgt_obs,hmm->svec[j].info,pri-
>id);
       /*******************************************************/

       res->tok.like+=outp;

       if (res->tok.like>max.like)
         max=res->tok;
       if (pri->states) {
         if (res->tok.align==NULL?TRUE:
             res->tok.align->state!=j || res->tok.align->node!=node) {
           align=NewNRefAlign(node,j,
                              res->tok.like-outp-res->tok.lm*pri->scale,
                              pri->frame-1,res->tok.align);
           res->tok.align=align;
         }
       }
    }
    else {
       res->tok=null_token;
       res->n=((pri->nToks>1)?1:0);
    }
  }

  /* Null entry state ready for external propagation */
  /*  And copy tokens from buffer to instance */
  for (i=1,res=pri->psi->sBuf+1,cur=inst->state;
       i<N;i++,res++,cur++) {
    cur->n=res->n; cur->tok=res->tok;
    for (k=0;k<res->n;k++) cur->set[k]=res->set[k];
  }

  /* Set up pruning limits */
  if (max.like>pri->genMaxTok.like) {
    pri->genMaxTok=max;
    pri->genMaxNode=node;
  }
  inst->max=max.like;

  i=seIndex[N][0]; /* Exit state (ignoring tee trP) */
  endi=seIndex[N][1];

  res=inst->exit;
  cur=inst->state+i-1;

  res->n=cur->n;
  res->tok=cur->tok;
  for (k=0;k<cur->n;k++) res->set[k]=cur->set[k];

  res->tok.like+=trP[i][N];

  for (i++,cur++;i<=endi;i++,cur++) {
```

```
    cmp.tok=cur->tok;
    cmp.tok.like+=trP[i][N];

    if (res->n==0) {
      if (cmp.tok.like > res->tok.like)
        res->tok=cmp.tok;
    }
    else
      TokSetMerge(res,&cmp.tok,cur);
  }
  if (res->tok.like>LSMALL){
    tok.like=res->tok.like+inst->wdlk;
    if (tok.like > pri->wordMaxTok.like) {
      pri->wordMaxTok=tok;
      pri->wordMaxNode=node;
    }
    if (!node_tr0(node) && pri->models) {
      align=NewNRefAlign(node,-1,
                         res->tok.like-res->tok.lm*pri->scale,
                         pri->frame,res->tok.align);
      res->tok.align=align;
    }
  } else {
    inst->exit->tok=null_token;
    inst->exit->n=((pri->nToks>1)?1:0);
  }
}
```

ProcessObservation() function which is called by HVite is substituted with modified version ProcessObservationWGT()

```
void ProcessObservationWGT(VRecInfo *vri,Observation *obs,Observation *
wgt_obs,int id, AdaptXForm *xform)
{
  NetInst *inst,*next;
  int j;
  float thresh;

/*   kostis_weight = obs; /*Spiros*/

  pri=vri->pri;
  inXForm = xform; /* sepcifies the transform to use for this observation */
  if (pri==NULL)
    HError(8570,"ProcessObservationWGT: Visible recognition info not
initialised");
  if (pri->net==NULL)
    HError(8570,"ProcessObservationWGT: Recognition not started");

  pri->psi->sBuf[1].n=((pri->nToks>1)?1:0); /* Needed every observation
*/
  pri->frame++;
```

```
  pri->obs=obs;
  if (id<0) pri->id=(pri->prid<<20)+pri->frame;
  else pri->id=id;

  if (obs->swidth[0]!=pri->psi->hset->swidth[0])
    HError(8571,"ProcessObservationWGT: incompatible number of streams
(%d vs %d)",
        obs->swidth[0],pri->psi->hset->swidth[0]);
  if (pri->psi->mixShared)
    for (j=1;j<=obs->swidth[0];j++)
      if (VectorSize(obs->fv[j])!=pri->psi->hset->swidth[j])
        HError(8571,"ProcessObservatioWGT: incompatible stream widths
for %d (%d vs %d)",
            j,VectorSize(obs->fv[j]),pri->psi->hset->swidth[j]);


  /* Max model pruning is done initially in a separate pass */

  if (vri->maxBeam>0 && pri->nact>vri->maxBeam) {
    if (pri->nact>pri->qsn) {
      if (pri->qsn>0)
        Dispose(&vri->heap,pri->qsa);
      pri->qsn=(pri->nact*3)/2;
      pri->qsa=(LogFloat*) New(&vri->heap,pri->qsn*sizeof(LogFloat));
    }
    for (inst=pri->head.link,j=0;inst!=NULL;inst=inst->link,j++)
      pri->qsa[j]=inst->max;
    if (j>=vri->maxBeam) {
      qcksrtM(pri->qsa,0,j-1,vri->maxBeam);
      thresh=pri->qsa[vri->maxBeam];
      if (thresh>LSMALL)
        for (inst=pri->head.link;inst->link!=NULL;inst=next) {
          next=inst->link;
          if (inst->max<thresh)
            DetachInst(inst->node);
        }
    }
  }
  if (pri->psi->hset->hsKind==TIEDHS)
    PrecomputeTMix(pri->psi->hset,obs,vri->tmBeam,0);
   /* PrecomputeTMixWGT(pri->psi->hset,obs,wgt_obs,vri->tmBeam,0); /*
Spiros*/
  /* Pass 1 must calculate top of all beams - inc word end !! */
  pri->genMaxTok=pri->wordMaxTok=null_token;
  pri->genMaxNode=pri->wordMaxNode=NULL;
  for (inst=pri->head.link,j=0;inst!=NULL;inst=inst->link,j++)
    if (inst->node)
/*      StepInst1(inst->node);*/
      StepInst1WGT(inst->node,wgt_obs); /*Spiros*/

  /* Not changing beam width for max model pruning */

  pri->wordThresh=pri->wordMaxTok.like-vri->wordBeam;
  if (pri->wordThresh<LSMALL) pri->wordThresh=LSMALL;
```

```
  pri->genThresh=pri->genMaxTok.like-vri->genBeam;
  if (pri->genThresh<LSMALL) pri->genThresh=LSMALL;
  if (pri->nToks>1) {
    pri->nThresh=pri->genMaxTok.like-vri->nBeam;
    if (pri->nThresh<LSMALL/2) pri->nThresh=LSMALL/2;
  }

  /* Pass 2 Performs external token propagation and pruning */
  for (inst=pri->head.link,j=0;inst!=NULL && inst-
>node!=NULL;inst=next,j++)
    if (inst->max<pri->genThresh) {
      next=inst->link;
      DetachInst(inst->node);
    }
    else {
      pri->nxtInst=inst;
      StepInst2(inst->node);
      next=pri->nxtInst->link;
    }

  if ((pri->npth-pri->cpth) > vri->pCollThresh ||
     (pri->nalign-pri->calign) > vri->aCollThresh)
    CollectPaths();

  pri->tact+=pri->nact;

  vri->frame=pri->frame;
  vri->nact=pri->nact;
  vri->genMaxNode=pri->genMaxNode;
  vri->wordMaxNode=pri->wordMaxNode;
  vri->genMaxTok=pri->genMaxTok;
  vri->wordMaxTok=pri->wordMaxTok;
}
```

Finally function StepInst1() was substituted with modified version StepInst1WGT()

```
static void StepInst1WGT(NetNode *node, Observation * wgt_obs) /* First
pass of token propagation (Internal) */
{
  if (node_hmm(node))
    StepHMM1WGT(node,wgt_obs);   /* Advance tokens within HMM
instance t => t-1 */
                /* Entry tokens valid for t-1, do states 2..N */
  else
    StepWord1(node);
  node->inst->pxd=FALSE;
}
```

B.2 Log-Likelihood in every time frame support

As described in 3.4, we had to make some tweaking in the HTK in order to report the log-likelihoods $P(\boldsymbol{o}_{s,t}|c)$ in every time frame.

We inserted some code in HRec.c and the output had the following fields:

Time frame	HMM Name	State	Likelihood

The coloured fields denote the class. Then we ranked the output data for each time frame according to likelihood values and took the N-best for the computation of the reliability indicators.

HRec.c source code modifications

Only HRec.c file was needed to be modified. Some additional source code was included to support the required functionality. Inside function StepHMM1() and from line 655 to 659 we have the following code

```
if (res->tok.like>pri->genThresh) { /* State pruning */
        outp=cPOutP(pri->psi,pri->obs,hmm->svec[j].info,pri->id);
        hmm_name=HMMPhysName(pri->psi->hset,hmm);
        res->tok.like+=outp;
             printf("%f %d %d %s\n",outp,j-1,pri->frame,hmm_name);
/*Spiros: 1:Probab 2:State 3:Frame 4:Model */
```

This allows HTK tools to output the required fields, as shown above, to the standard output and then used appropriately by our implementation.

Bibliography

[1] G. Potamianos, C. Neti, G. Gravier, A. Garg and A.W. Senior "Recent Advances in the Automatic Recognition of Audiovisual Speech", in Proceedings of the IEEE Vol. 91 No. 9 September 2003

[2] Z. Malafouris, "Automatic speech transcriptions of Greek broadcast news ", Diploma Thesis, Technical University of Crete, June 2007

[3] A. Rogozan, "Discriminative Learning of Visual Data for Audiovisual Speech Recognition", International Journal on Artificial Intelligenece Tools Vol. 8 No. 1 pages 43-52, March 1999

[4] E. Patterson, S. Gurbuz, Z. Tufekci, J. Gowdy, "Moving-Talker, Speaker-Independent Feature Study, and Baseline Results Using the CUAVE Multimodal Speech Corpus", EURASIP Journal on Applied Signal Processing 2002:11, 1189-1201.

[5] HTK - Hidden Markov Model Toolkit - Speech Recognition toolkit. (http://htk.eng.cam.ac.uk/)

[6] C. Cibelushi, F. Deravi, J. Mason, "A Review of Speech-Based Bimodal Recognition", IEEE Transactions on Multmedia, Vol. 4, No. 1, March 2002

[7] H. Bourlard, S. Dupont, "A NEW ASR APPROCH BASED ON INDEPENDENT PROCESSING AND RECOMBINATION OF PARTIAL FREQUENCY BANDS", Philadelphia, USA, October 1996, ICSLP, Vol. 1

[8] G. Gravier, G. Potamianos, C. Neti, “Asynchrony modeling for audio-visual speech recognition”, Proc. Humal Language Technology Conference, San Diego, California, March 24-27, 2002

[9] M. Heckmann, F. Berthommier, K. Kroschel, “Noise Adaptive Stream Weighting in Audio-Visual Speech Recognition”, EUROASIP, Journal of Applied Signal Processing, Vol. 1, pp. 1260-1273, November 2002

[10] A. Garg, G. Potamianos, C. Neti, and T.S. Huang, “Frame-dependent multi-stream reliability indicators for audio-visual speech recognition”, Proc. Int. Conf. Acoust. Speech Signal Process., vol. I, pp. 24-27, Hong Kong, Apr. 2003.

[11] A. Potamianos, E. Sanchez-Soto, K. Daoudi, “Stream Weight Computation for Multi-Stream Classifiers”, Proc. Of Intl. Conf, Acoustic, Speech and Signal Proc., Toulouse, France, May 2006

[12] E. Sanchez-Soto, A. Potamianos, K. Daoudi, “Unsupervised Stream Weight Estimation using Anti-Models”, Intl. Conf, Acoustic, Speech and Signal Proc., Honolulu, Hawaii, USA, April 15-20, 2007

[13] K. Dermatas “Soft feature decoding for speech recognition over wireless channels”, Diploma Thesis, Chania, Technical University of Crete, 2005