FEATURE EXTRACTION OPTIMIZATION AND STREAM WEIGHT ESTIMATION IN AUDIO-VISUAL SPEECH RECOGNITION

RODOMAGOULAKIS ISIDOROS

A Thesis presented for the degree of
Electronic and Computer Engineer

Telecommunications Research Group
Department of Electronics and Computer Engineering
Technical University of Crete
Chania,Greece

October, 2008

Supervisor : Alexandros Potamianos

Committee member : Vasilios Digalakis

Committee member : Athanasios Liavas

Dedicated to

My Family, Friends, Teachers and Colleagues.

my dear Fotinoula

FEATURE EXTRACTION OPTIMIZATION AND STREAM WEIGHT ESTIMATION IN AUDIO-VISUAL SPEECH RECOGNITION

RODOMAGOULAKIS ISIDOROS

Submitted in partial fulfillment of the requirements for the degree of
Engineer in Technical University of Crete
October, 2008

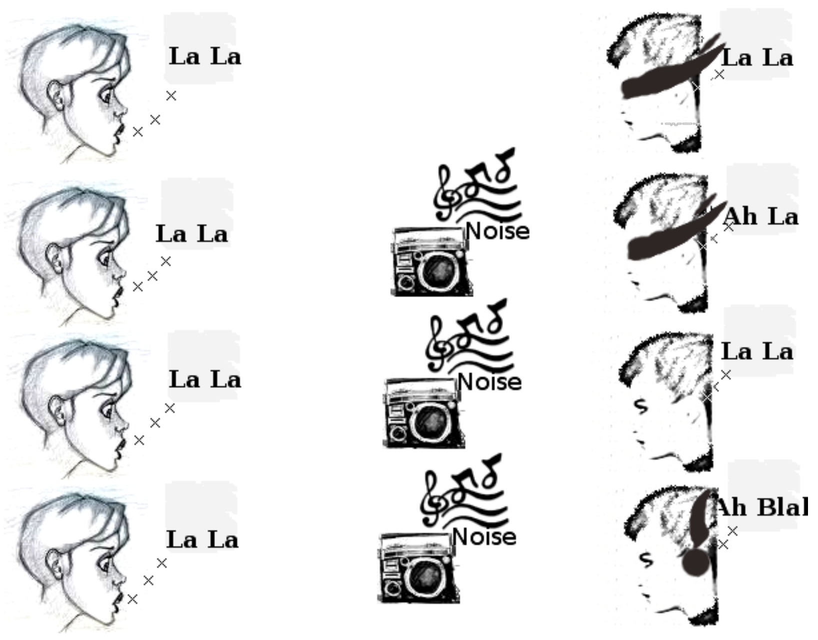


Figure 1: This is the main concept of Audio-Visual Speech Recognition.

Declaration

The work in this thesis is based on research carried out at the Telecommunications Laboratory of Technical University of Crete, Chania, Greece. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

Abstract

Automatic Speech Recognition (ASR) is an essential component in many Human-Computer Interaction systems. A variety of applications in the field of ASR have reached high performance levels but only for condition-controlled environments. Further development is necessary for better accuracy in real conditions, where environmental or other kind of noises exist. Audio signal features need to be enhanced with additional sources of complementary information to overcome problems due to large amounts of acoustic noise. Visual Information extracted from speaker's mouth region seems to be promising and appropriate for giving audio-only recognition a boost. Lip/Mouth detection and tracking combined with traditional Image Processing methods may offer a variety of solutions for the construction of the visual front-end schema. Furthermore, Audio and Visual stream fusion appears to be even more challenging and crucial for designing an efficient AV Recognizer.

In this project, we investigate some problems in the field of Audio-Visual Automatic Speech Recognition (AV-ASR) concerning visual feature extraction and audio-visual integration. Color-based detection and Template Matching strategies are used to detect and track the mouth region, which is considered as the Region of Interest (ROI) through sequential time frames. Subsequently, Discrete Cosine Transform (DCT) is used to transform pixel values in "compact", descriptive features. We prove that some factors related with ROI detection, like the inclination of the speaker's head as well as its size, affect the performance of both visual and audio-visual recognizers. In order to counter these effects, we investigate some methods for rotation correction and scaling normalization. The improved visual front-end schema yielded a Word Recognition Error (WER) reduction of 95% over a baseline implementation. On the other hand, we investigate a new approach for the unsupervised stream weight estimation which is based on K-means clustering. Stream weight behaviour and adaptability is tested under a variety of noises, in a

word or sentence level, for classification and recognition tasks. Finally, we compare the results for static and adaptive weighting to evaluate our weight estimation approach and moreover, we measure the improvements achieved by using an audio-visual recognizer instead of the traditional audio-only recognizer. All experiments were based in CUAVE AV database and furthermore, recognizers were built using HTK.

Acknowledgements

This project is the result of hard work and discipline. I would like to thank my professor and supervisor Alexandros Potamianos for his guidance and his usefull advices. Our partnership proved effective through the progress of this project. Also, Spiros Dimopoulos passed me his ideas and previous experience he had with this subject and moreover he helped me to overcome some critical problems appeared during the creative process. “Thank you” George Skoumas for offering your help during the presentation of this project. “Thank you” Spiros Meliopoulos and Nikos Georgantonis for the invaluable moments we had in the telecom lab and our helpful discussions. “Thank you” Fotini and George for offering your respectable opinion. Finally, I would like to thank my family for supporting me all these years.

Contents

List of Figures

List of Tables

Introduction

In recent years, the field of Audio Visual Speech Recognition (AVSR) has proved to be of significant interest for many researches, as the traditional Audio Automatic Speech recognition (ASR) systems seem to work only for relatively controlled environments. A major problem of ASR is robustness under channel and environmental noise. Many techniques have been investigated to improve the recognition under 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. However, for ASR to approach human levels of performance and for speech to become a truly pervasive user interface, we need novel, nontraditional approaches that have the potential of yielding dramatic ASR improvement. Lipreading, as an alternative source of information, consists such a different approach which is not affected by the acoustic environment and noise, and it possibly contains the greatest amount of complementary information to the acoustic signal. The additional information obtained from lipreading aid audio to recover from corruptions due to noise.

AV-ASR systems can outperform audio-only recognizers, particularly in environments where background noises and multiple speakers exist. Researchers have demonstrated the relationship between lipreading and human understanding and have produced performance increases with multi-modal systems. It is well known that humans have the ability to lipread: We combine audio and visual information in deciding what has been spoken, especially in noisy environments. One of the key properties of bimodal speech to emerge from such analysis is that of complementarity: Features that are the hardest to distinguish acoustically are the easiest to distinguish visually, and vice versa. The sensory integration of auditory and visual information in speech perception and the complementarity between these modalities shows clearly in experiments that independently vary auditory and visual information [8].

Various sets of visual features have been proposed over the last 20 years. In general, they can be grouped into three categories: High level lip contour (Shape) based features, low level appearance (pixel) based ones and finally a combination of both. These features are extracted from a sub-region of the lower-half, of the speaker's face or alternatively from the lip contour. Effects like head rotations and differences in scaling must be countered, as speakers tend to move in front of the camera. Such normalizations yield high discrimination among classes of interest.

Audio-visual fusion is an instance of the general classifier combination problem. In our case, two observation streams are available (audio and visual modalities) and provide information about hidden class labels, such as HMM states, or, at a higher level, word sequences. Each observation stream can be used alone to train single-modality statistical classifiers to recognize such classes. However, one hopes that combining the two streams will give rise to a bimodal classifier with superior performance to either single-modality ones. One of the main challenges in AV-ASR systems is the audio-visual information integration problem. The main issues in information integration are, (a) the class conditional dependence assumption made across streams, (b) the level (e.g. frame, phone, word) of integration, and (c) the kind (e.g. feature, partial likelihood, partial decision) of integration. Generally, speaking it is widely accepted that there are two kinds of integration: feature fusion and decision fusion. Feature fusion concatenates audio and visual features in a common feature space and captures any possible dependency between them. On the other hand, late integration, also known as decision fusion, takes into account the decisions made partially from each stream to conclude in a final decision. In this way, it is feasible to emphasize or deemphasize each stream's contribution to the final decision, making use of stream weights. The question of how these weights are going to be estimated is a strong motivation for many researches.

A brief description of how this thesis is organized follows:

Chapter 1 contains an introductory description about Audio-Visual Automatic Speech Recognition (AV-ASR).

Chapter 2 represents the visual front-end method we used to extract the visual features, including some improvements we made in a state-of-the-art visual front-end, such as rotation correction and scaling normalization of the obtained Region of Interest (ROI)

Chapter 3 enumerates methods for computing stream confidence and estimating stream weights including the K-means based method we applied in our weight estimation schema.

Chapter 4 presents the Audio-Visual Database we used for our experiments and analyzes in detail how we implemented HMM-based recognizers using HTK. Also describes how we apply K-means to compute stream weights for the tasks of recognition and classification.

Chapter 5 reports the project results, divided in two parts; These parts analyze the results obtained from Chapter 2 and Chapter 3 respectively, using figures and tables in order to be understood in which way visual information contributes to the overall improvement has been made.

Chapter 6 proposes some possible improvements can be made as well as some conclusions about this work.

Chapter 1

Audio Visual Automatic Speech Recognition

1.1 LipReading

It is well known that humans have the ability to lipread: We combine audio and visual information in deciding what has been spoken, especially in noisy environments. An example of *"catastrophic fusion"* is the so-called McGurk effect, where a spoken sound /ga/ is superimposed on the video of a person uttering /ba/. Most people perceive the speaker as uttering the sound /da/. In addition, the visual modality is well known to contain some complementary information to the audio modality. For example, using visual cues to decide whether a person said /ba/ rather than /ga/ can be easier than making the decision based on audio cues, which can sometimes be confusing. On the other hand, deciding between /ka/ and /ga/ is more reliably done from the audio than from the video channel. The above facts have recently motivated significant interest in the area of Audio-Visual Automatic Speech Recognition (AV-ASR), also known as automatic lipreading, or speechreading. Work in this field aims at improving automatic speech recognition by exploring the visual modality of the speaker's mouth region, in addition to the traditional audio modality. Not surprisingly, automatic speechreading has been shown to outperform audio-only ASR over a wide range of conditions. Such performance gains are particularly impressive in noisy environments, where traditional ASR performs poorly. Coupled with the diminishing cost of quality video capturing systems, this fact makes automatic speechreading tractable for achieving robust ASR in certain scenarios

1.2 How do we percept Audio and Visual Information?

In the case of visual speech, it is well known that not all phones are visually distinct, but rather they are clustered in so-called "visemes". An example of such categorization is depicted in Table (1.1), in which "viseme" classes can be categorized into 13 classes. The main criterion considered for this clustering is the place of articulation. However, some speech units are visually indistinguishable.

For example, /bi/ and /pi/ are percepted like being the same. Nevertheless, /pi/ and /bi/ are easily distinguished acoustically and that is because they have different voice-on-set times (VOT), which is the delay between the initial burst sound and the onset of vibration of the vocal folds. In this case, VOT for /pi/ is noticeably longer than in /bi/. Respectively, giving identical information for /mi/ and /ni/, acoustic features need to be enhanced by complementary visual information which play a catalytic role for taking the right decision in this case. /Mi/ and /ni/ have the same voicing and nasality (extra resonances from the nasal cavity), differing only in where the speaker closes the vocal tract with the lips, or tongue, or teeth.

One of the key properties of bimodal speech to emerge from such analysis is that of complementarity: Features that are the hardest to distinguish acoustically, are the easiest to distinguish visually, and vice versa. The sensory integration of auditory and visual information in speech perception and the complementarity between these modalities shows clearly in experiments that independently vary auditory and visual information [8]. McGurk is famous for his realization that in the most of the cases, when people hear /ba/ and see /ga/, they are confused and decide /da/. *McGurk effect* as it is called, is the most representative example of *"catastrophic fusion"*.

It has been shown that visible articulators, during an utterance, start and complete their trajectories asynchronously, exhibiting both forward and backward co-articulation with respect to the acoustic speech wave. Intuitively this makes a lot of sense, as visual articulators (i.e. lips, tongue, jaw) have to position themselves correctly before and after the start and end of an acoustic utterance. It has been also found in [9] that an audio lead of less than 80ms or lag of less than 140ms could not be detected during speech.

Silence	/sil/ , /sp/
Lip-rounding based vowels	/ao/ , /ah/ , /aa/ , /er/ , /oy/ , /aw/ , /hh/ /uw/ , /uh/ , /ow/ /ae/ , /eh/ , /ey/ , /ay/ /ih/ , /iy/ , /ax/
Alveolar-Semivowels	/l/, /el/, /r/, /y
Alveolar-Fricativies Alveolar Palato-Alveolar Bilabial Dental Labio-Dental Velar	/s/ , /z/ /t/, /d/, /n/, /en/ /sh/, /zh/, /ch/, /jh/ /p/, /b/, /m/ /th/, /dh/ /f/, /v/ /ng/, /k/, /g/, /w

Table 1.1: An example of “viseme” class categorization found in [7]

However, if the audio was delayed by more than 160ms it no longer contributed useful information, signifying the importance of some degree of asynchrony and synchrony in continuous audio-visual speech perception.

Various sets of visual features have been proposed over the last 20 years. In general ,they can be grouped into three categories: High level lip contour (óhape) based features, low level appearance (pixel) based ones and finally a combination of both.

1.3 Audio-Visual Automatic Speech Recognizer

The main difference between a traditional audio-only automatic speech recognition (ASR) system and an audio-visual ASR system is the participation of the visual information. However, it has been already researched a concept of an ASR system that uses information from multiple streams which consist of different audio spectral sub-bands [3]. Audio-visual approach is based in the same idea except that there are two streams, audio and visual. Each stream manipulates different kind of information. Acoustic waves and visual signals are combined after being transformed into “compact” and descriptive features.

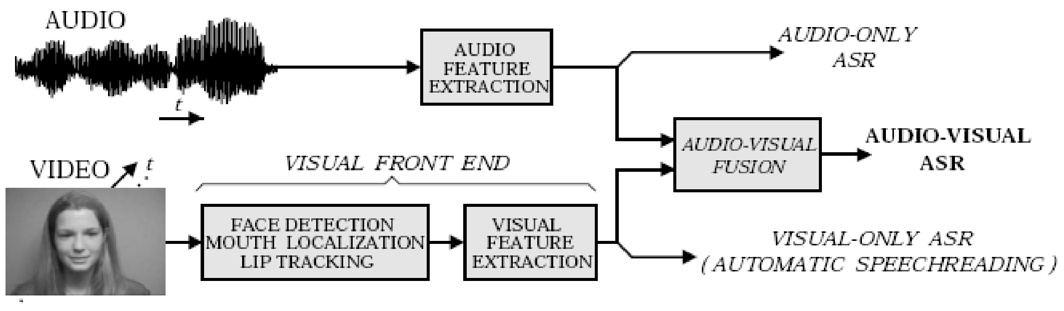

AUDIO
t
AUDIO FEATURE EXTRACTION
AUDIO-ONLY ASR
VIDEO
t
VISUAL FRONT END
AUDIO-VISUAL FUSION
AUDIO-VISUAL ASR
FACE DETECTION MOUTH LOCALIZATION LIP TRACKING
VISUAL FEATURE EXTRACTION
VISUAL-ONLY ASR (AUTOMATIC SPEECHREADING)

Figure 1.1: The main concept of an AV-ASR system

This process, so-called *feature extraction* is crucial for the efficiency of the recognizer. The more discriminative feature vectors are generated, the better results are obtained.

The combination of the two streams, known as audio-visual integration (*fusion*), belongs to the general problem of classification using multiple classifiers. In our case, HMM-based classifiers corresponding to the audio and visual streams respectively, are trained separately or like a joint model, assuming various levels of synchronization between their states. The final set of audio-visual multi-stream HMMs consists of phone (context-dependent or context-independent) models or alternatively word-level models. Their class-conditional observation likelihood is the product of the observation likelihoods of their single-stream components, raised to appropriate stream exponents that capture the reliability of each modality, or, equivalently, the confidence of each single-stream classifier.

An alternative integration method is based on the feature fusion, in which audio and visual feature vectors are concatenated on an early stage and then treated like belonging in a single feature space. Class dependency is assumed in contrast with decision fusion. The reduction of the feature space dimensions is feasible by applying Linear Discriminal Analysis (LDA) or alternatively Principal Component Analysis (PCA). A more sophisticated feature fusion schema is proposed in [13] for a Large Vocabulary Speech Recognition (LVSR) system. The remainder of this thesis describes in detail the parts of this system as well as the implementation we considered in order to perform experiments over the visual feature extraction and stream weight estimation.

Chapter 2

Feature Extraction

A main problem in the research field of audio as well as audio-visual ASR is feature extraction. If the extracted features from the audio signal or the speaker's images are carefully chosen, it is expected that the feature set will extract the relevant information in order to be performed an efficient recognition, classification. If classes of interest (such as phones in audio, or visems in lipreading) are well-discriminated and also feature space dimensions are reduced enough, then the feature set is considered to be appropriate for pattern recognition.

In the case of audio, mel-frequency cepstrum coefficients (MFCC) are often used in applications such as speech recognition, voice recognition, audio compression and music information retrieval. They are derived from a mel-frequence cepstrum (specrtum-of-spectrum) where the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. MFCCs are not very robust in presence of additive noise, and so researchers introduced visual information in speech recognition to overcome this drawback.

Various sets of visual features have been proposed over the last 20 years for this purpose. In general ,they can be grouped into three categories: High level lip contour (óhape) based features, low level appearance (pixel) based ones and finally a combination of both. In the first approach, the speaker's inner (and/or outer) lip contours are extracted from the image sequence. A parametric, or statistical lip contour model is then obtained and the model parameters are used as visual features. Alternatively, lip contour geometric features are used, such as mouth height and width. Appearance based features are extracted from a larger region around the mouth, including the jaws in some cases to exploit useful information coming from the articulations. A region of interest (ROI) is selected from speaker's face and then a transformation like Discrete Cosine Transform is used to transform pixel values into features. Alternatively, ROI projection into a reduced feature space using principal component analysis (PCA) is also used.

2.1 Visual Front-End

This section describes how visual features are extracted from each video frame and get processed to form a set of vectors, which exploit visual information and enhance audio features. ROI is detected and tracked through the frame sequence. A face detection module is used in the first place in order to estimate the face silhouette, using color based classification and image processing techniques. After the face region is determined, an other routine could be used to discriminate "lip pixels" from "non lip pixels" by applying the same kind of classification, and then, a larger region around the mouth could be selected and passed to an image-transformation algorithm such as DCT. Instead, in order to facilitate ROI tracking in every frame, a nose template is formed and used for nose tracking. Section 2.1.2 describes how ROI is detected applying template matching. As described in section 2.1.3, ROI's rotation correction and scale normalization are performed by employing eye detection to obtain useful information about the head inclination as well as the relative head size referring to an average size.

The following steps describe the process of the visual feature extraction and figure (2.1) depicts its schematic:

- Face Detection (every 10 frames) and nose template re-formation (every 10 frames).
- Template matching to track the nose (in every frame)
- Lip region estimation.
- Eye detection, estimation of the head inclination, scaling estimation.
- Image rotation and scaling correction
- ROI determination

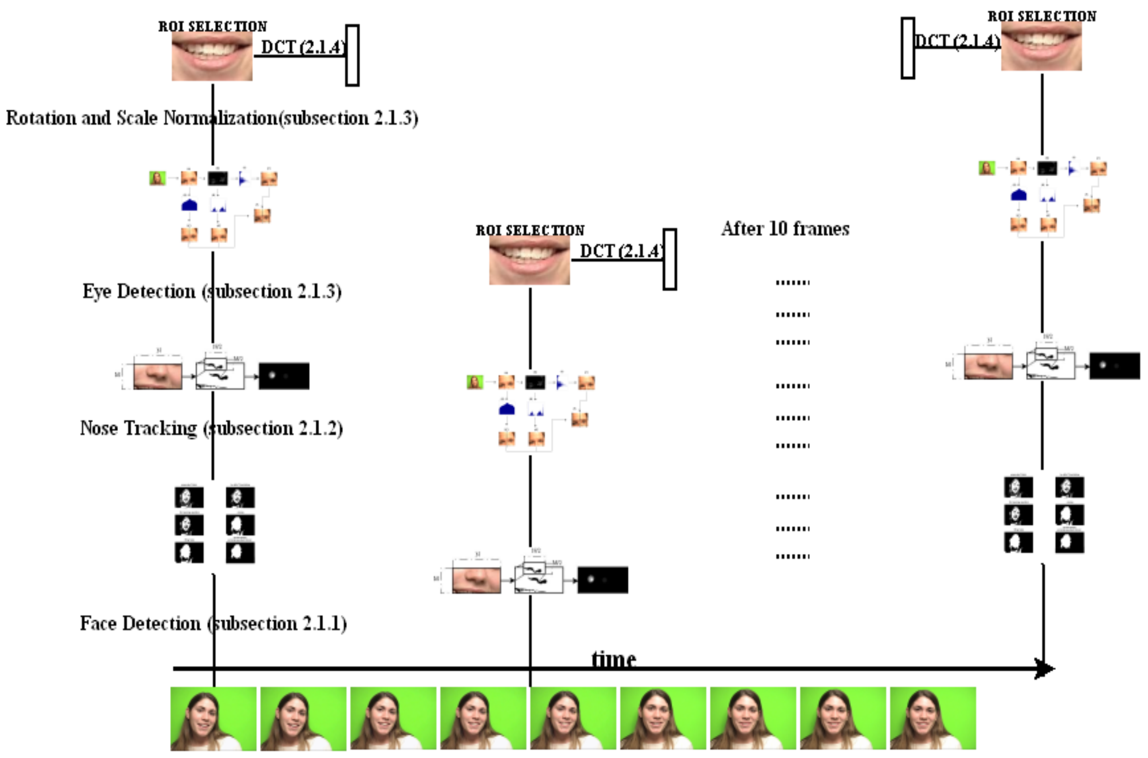

ROI SELECTION
DCT (2.1.4)
Rotation and Scale Normalization(subsection 2.1.3)
Eye Detection (subsection 2.1.3)
Nose Tracking (subsection 2.1.2)
Face Detection (subsection 2.1.1)
ROI SELECTION
DCT (2.1.4)
After 10 frames
ROI SELECTION
DCT (2.1.4)
time

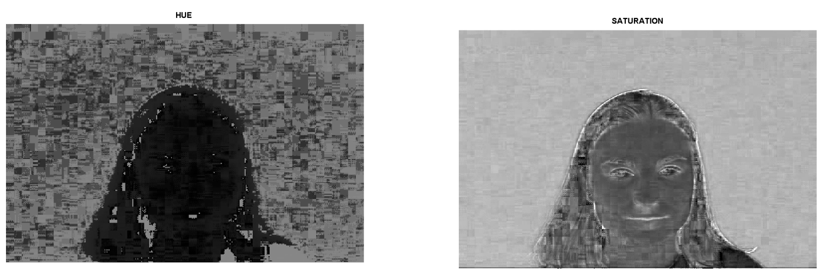
Figure 2.1: Visual Feature Extraction Schematic diagram

- DCT feature extraction

All the modules included in the schema above are analyzed in the next subsections.

2.1.1 Chroma-Based Face Detection

Face detection/localization and tracking consist one of the main components in many machine vision applications. Methods like template matching, optical flow calculation, chroma detection, shape-based techniques or a combination of the above are usually applied to localize and track a face. In this project, we use face detection in order to restrict our searching for the rest of the facial features on a specific area. Furthermore, nose tracking, as represented in section 2.1.2, may fail to track the nose in some cases due to



(a) Hue component of HSV model (b) Saturation of of the same image

Figure 2.2: Illustration of the discrimination between "skin" and "non-skin" pixels using HSV color model. Values in both grayscale images are normalized in [0 (black) , 1 (white)], with zero value corresponding to red hue for (a), and saturated colors for (b)

rapid changes or image noise. Having the knowledge of where the face is located approximately, the nose tracking algorithm is able to confirm the validity of the obtained result.

We chose to implement a chroma-based detection because of the problem nature. CUAVE database [10], as described in section 4.1, was captured in a condition-controlled environment, without the presence of difficult light variations since a simple green background was used to facilitate face detection. To remove luminance component which is the same for every part of the image and consequently, it is meaningless to use it as an informative feature, we transformed red-green-blue (RGB) components of each frame to hue-saturation-value (hsv) space. In the next histograms, it is clear that skin's hue and saturation levels lie close to zero in contrast with the background. Skin segments contain non-saturated, red-chroma pixels. For this reason, face detection is simplified using hue and saturation to discriminate skin from "non-skin" pixels. This kind of discrimination that HSV model offers is depicted in Fig.(2.2).

Every pixel px_i is given a probability $P_{SKIN}\left(px_i\right) = P_{HUE}\left(px_i\right) \cdot P_{SAT}\left(px_i\right)$ of being a part of skin. P_{HUE} determines the Probability Density Function (pdf) of the hue component, of skin pixels. P_{SAT} determines the pdf of the saturation component respectively. Skin tone variations make the creation of a speaker independent pdf for hue and sat necessary. This is achieved by cropping all the skin segments within every speaker's face area and compute a pdf for each component. Fig.(2.3) depicts P_{HUE} and

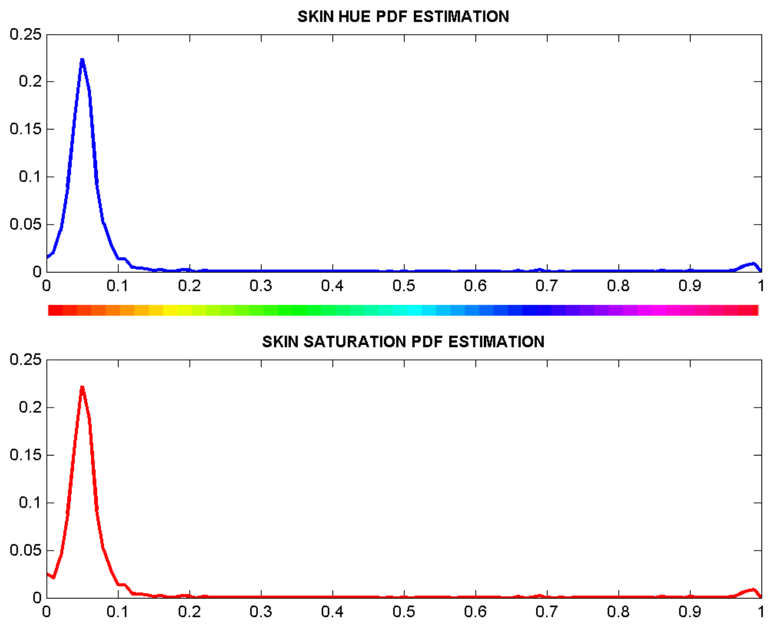


Figure 2.3: Estimated speaker independent PDFs for hue and saturation components of skin pixels

P_{SAT} respectively.

P_{SKIN} is considered as a grayscale image with pixel values $I \in [0..1]$ referring to probabilities. Pixels with small probabilities have to be rejected, so we apply a threshold which results to a black and white (bw) image. Usually, a minimum between two peaks in the histogram of a grayscale image is considered to be a good choice for a threshold. In order to perform the binarization of P_{SKIN}, we choose a threshold that satisfies $T = (I_1 + I_2)/2$, where I_1,I_2 are two of the most frequent values of P_{SKIN} and their difference is bigger than 0.3 ($I_1 - I_2 > 0.3$).

Fig.(2.4) illustrates the process of refining the binarization result using *opening* and *closing* morphological operations. Opening and closing are two operators from mathematical morphology. They are both derived from the fundamental operations of *erosion* and *dilation*. Like those operators, they are normally applied to binary images. Erosion, in general, causes objects to shrink. The amount and the way that they shrink depend upon the choice of the structuring element. The structuring element is to mathematical morphology what the convolution kernel is to linear filter theory. In our case, a disk with

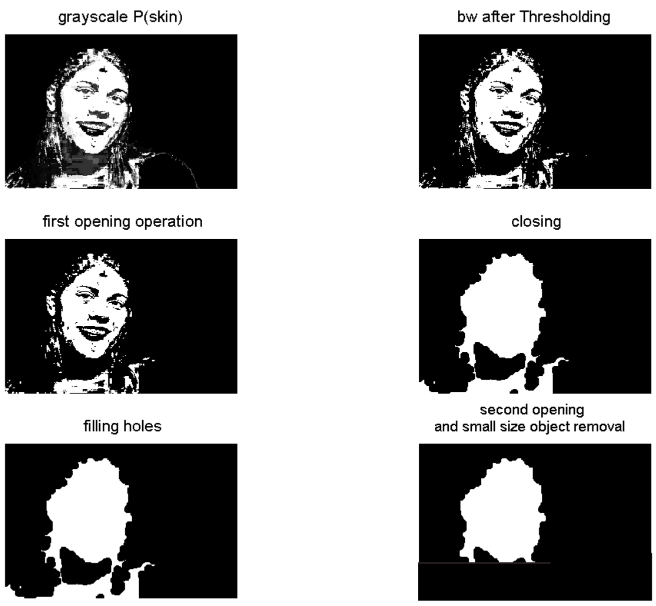


Figure 2.4: After the binarization of the grayscale image P_{SKIN}, a set of morphological operations are applied to form the final mask

radius 5 pixels is shifted over all the possible positions in the image and if any of the pixels contained in the disk area is set to 0, the output pixel, which is the center of the disk, is set to 0. On the other hand, dilation causes objects to dilate or grow in size by setting the pixel in the center of the disk to 1 if a pixel contained in the disk area is set to 1. The definition of a morphological opening of an image is an erosion followed by a dilation, using the same structuring element for both operations. The related operation, morphological closing of an image is the reverse: it consists of dilation followed by an erosion with the same structuring. We applied an opening operation in order to remove small objects from the binary image which considered as noise rather than face candidates. Subsequently, a closing operator is applied to fill the holes in the face region as Fig.(2.4) depicts.

Alternatively, *region growing* can be used to segment the grayscale image into regions with similar values, and then select as face one of the formed regions which satisfies some morphological criteria. For instance, knowing that the size of the face region is bigger than a threshold, it restricts all the candidate face regions to satisfy $CandidateRegionSize > T_1$. Furthermore, restrictions deal with the orientation of each candidate region or its

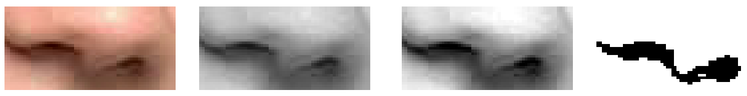

(a) Nose nostrils and bottom (b) Grayscale image (c) Histogram equalization (d) Black and white Template

Figure 2.5: A binary image of the nose region is created for template matching. Applying a threshold to the grayscale image results in (d)

centroid position can be represented as $T_2 < CandidateRegionOrientation < T_3$ and $CandidateRegionCentroid \in [x_1 \ldots x_2, y_1 \ldots y_2]$, where T_1, T_2 respond to some thresholds, and $[x_1 \ldots x_2, y_1 \ldots y_2]$ defines a region within the image.

We have to mention that in some cases, when P_{SKIN} did not indicate *"skin pixels"* correctly, face detection did not work. We took that as a bad estimation of the speaker independent pdfs, P_{HUE} and P_{SAT}. We solved this problem by estimating those pdfs separately for a speaker whose skin tones could not be described well by global pdfs.

2.1.2 Template Matching for Nose Tracking

Once mouth region is detected and ROI is obtained, it is necessary to repeat the same process for the next frames as well. This is called ROI tracking. The main difference from ROI detection is that tracking uses temporal correlation to locate ROI in a video sequence instead of detecting it in each frame independently. With temporal information, we can narrow down the search range significally. We have to track the mouth region in every possible change caused by speaker's nodding the head during articulation, or moving back and forth. There are several methods to achieve this. *Face tracking* employs *face localization* in a frame sequence to update face location using temporal correlation. In addition, *lip tracking* can be performed by using *Active-Appearance-Models (AAM)* or *lip contour estimation.* We choose to implement a *nose tracker* for detecting nose positions, and being able to extract the ROI using the nose tip as reference. It is important for the visual frond-end to handle tracking errors that result in noisy visual data and as a consequence in inaccurate recognizers.

The next properties of the nose region make *nose tracking* feasible by using *template matching*:

- There is a strong gradient change in the nose area because of the nostrils.
- Nose is located at the center of the face.
- No sudden changes occur while speaker's head is moving.
- Nose area is not significally deformed during articulation.

Our aim is to measure nose tip locations through senquencial frames. As Fig.(2.5) depicts, a black and white (bw) template image is formed by cropping the nostrils and also the bottom of the nose. Nostrils have a distinctive darkness from the facial skin and they can be detected easily by thresholding the grayscale image, and keeping the darkest pixels. *Thresholding* is a main issue for *image segmentation*. Processing of the intensity histogram is essential for the successful selection of a proper threshold which distinguishes nostrils from the rest parts of the nose. The same threshold will be used for thresholding the search area.

Template matching is applied by computing the *cross-correlation* measure between the template mask and all the possible template overlaps within the search area. The template position which results to a maximum cross-correlation, is considered as the match point. Algorithm 1 describes this concept.

$C\,(i,j)$ values indicate the similarity between the template, placed with its upper-left corner in $mp = (i,j)$, and the corresponding overlapped region of the search image. We apply a threshold to reject small values contained in C and after that, the centroid of the largest region of the remained pixels is picked up as the best match. Fig.(2.6) depicts how the template is shifted over the search image, and what kind of result is produced after applying a threshold at the cross-correlation image C. The nose tip is considered to lie at the center of the template image T. We are able to find ROI coordinates using this point as reference. Section 2.1.3 describes the rest details for *ROI extraction*, with respect to the nose tip tracking.

Some issues concerning the template matching initialization as well as the size selection of the search area, are under consideration. One way to initialize template matching is by applying a supervised template selection in the first frame of the sequence. On the other hand, we can use face detection to obtain the face silhouette, and then try to form the nose template by searching in the center of this area. We choose the first option to

Algorithm 1 Template matching using cross-correlation

- $T\,[i,j]$ is the *30 x 60* template
- $S[i,j]$ is a *60 x 120* search area (two times wider and taller than the template).
- $C[\delta_x,\delta_y] = \sum_{i=0}^{i=X-1}\sum_{j=0}^{j=Y-1} S[i,j]\cdot T[i+\delta_x, j+\delta_y]$,

where $C[\delta_x,\delta_y]$ is the cross correlation, measured between a (δ_x,δ_y) shifting of the template and the corresponding overlaped search region.

Alternatively:

$C = IDFT\,\{DFT\,(S)\cdot DFT\,(T)\}$, where DFT is the Discrete Fourier Transform of the images.

- $matchpoint = (k,l)$ if $C\,(k,l) = max\,(C)$
- $nosetip = matchpoint + (templatesize/2\,,\, templatesize/2)$

Note that

$T\,[i,j]$ is *zero-padded* to match with $S\,[i,j]$

ensure that template matching will be performed correctly. In addition, we adjust the size of the search area to be two times wider and taller than the template size. There are two main reasons for this choice. First, we know a priori that no rapid movements will occur during articulation, and afterwards, we tested template matching for various sizes and noticed that when we increased the search area size, various shades and objects are included. This is a kind of confusion for template matching which was not accurate for those cases. Note that we use the match point as the center of the search area in the next frame.

Template matching is appropriate for tracking only for translations of the template. When rotations and size changes occur, it is essential to apply a *"greedy"* searching to achieve good results. In this project, we did not have to face such cases. We tested *affine transformations* [16] for some speakers with the tendency to slightly rotate their heads,

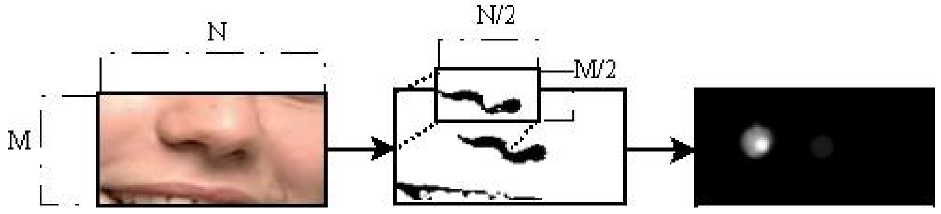


Figure 2.6: An illustration of the template matching process.

but the results were not satisfying. An other way to estimate the inclination of the head is by measuring the angle between the vertical axis of the picture and a line which connects the located eyes, as described in the next section. Finally, tracking result points were smoothed in order to reduce a possible mismatch error.

2.1.3 Rotation and Scale Normalization using Eye Detection

Speakers are expected to slightly rotate their heads during articulation due to natural movements. Also, scale changes occur when they move back-and-forth from the camera or their head size differs from the average size we assumed. It is essential to normalize these variations before extracting a region of interest which is placed under the nose tip we detected. This is achieved by performing face alignment. It involves spatially rotating and scaling the face image in order to extract representative DCT features. Given the location of both eyes, we are able to estimate the rotation angle and also the distance between the eyes, which consists a measure of the scale parameter.

As Fig.(2.7) depicts, we estimate the orientation of the face, which is defined as the formed angle θ between the horizontal axis of the image and line γ. Line γ, so-called eye-line, connects speaker's eyes. Furthermore, the $averageDistance/d$ ratio denotes the scale factor we use to perform *scaling normalization* . But first, we have to perform *eye localization*. Several approaches have been proposed for this task. To our knowledge, eye localization is considered to be a simple task if light conditions are controlled, like in our case. We restricted our search to the upper half of the detected face silhouette. The proposed method in [6] for grayscale images, combines vertical and horizontal projections of the gradient image in order to locate a rough region where the eyes could lie. Then,

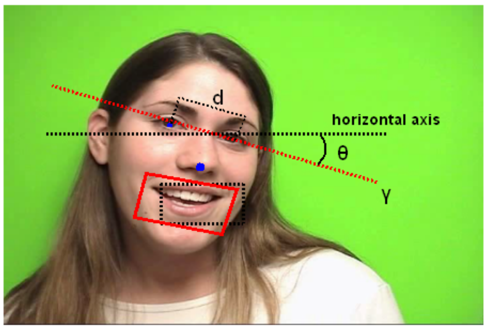


Figure 2.7: Rotation correction and Scale Normalization.

template matching is applied for the accurate detection of the iris centers.

Fig.(2.8) represents the process we follow to estimate the coordinates of two rectangular areas, where template matching will be applied. This method performs an hierarchical search starting with the upper-half (a) of the face mask which is obtained from face detection we described in section 2.1.1. Vertical and Horizontal projections in (d) and (c) of the gradient image (b) are computed in order to measure intensity changes that occur within the area. Each peak of the vertical projection indicates the vertical position of each eye respectively. This is due to the increased gradient in the eye region. The same effect appears for the horizontal projection. A peak is also formed that declares in which horizontal stripe the eyes could be located. Furthermore, the horizontal projection of the intensity image results in the histogram (e), which denotes a central vertical line (e') of the face lying between the eyes, where the intensity values are increased. Using the horizontal projection (c), we are able to restrict our search area within a horizontal zone (c'), and taking into account the vertical projection (d) we further reduce the search area in (d') by rejecting two vertical zones on the edges of the image. The final reduced area is a combination of all the above operations. The two rectangular regions formed in (f) are going to be scanned for template matching. It is worth mentioning that gradient image (b) was obtained after applying *Sobel Operators* for *edge detection*.

The same cross-correlation metric, as described in section 2.1.2, is used to find the best match. The first step for template matching is obviously considered to be template creation. However, the selected template, which consists of one usual eye image in this case, cannot be directly used for matching, because template size is not the same with that in the real image which is going to be examined. At this point, a simple solution for this

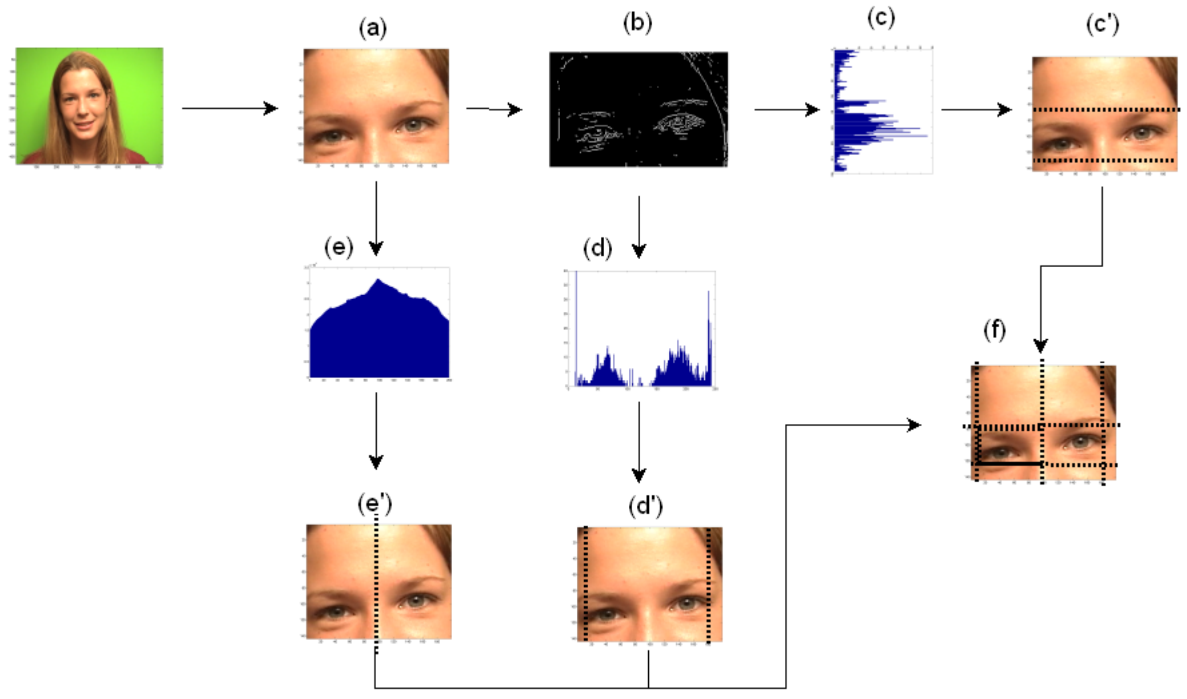


Figure 2.8: The main method we use in order to reject a significant part of the search area before applying template matching for iris detection.

problem is to perform the process of matching several times, and each time we will use the template with different size. But this method is very ineffective.

We have to decide which template size we should use for an efficient matching. Information about the geometric features of the face, concerning height and width parameters, is feasible to be extracted if we look at the vertical and horizontal projections of the image gradient. For example, gradient values of both sides at the vertical edges of face region near the cheeks, reach a local maxima in vertical projection histogram (d), which indicates the vertical borders of the face. Subsequently, we can measure the width of the face which is next used to normalize template images. This method achieved good results with limited success for speakers wearing glasses. In these cases, we used *MachinePerception-Toolbox* [2] to detect the eyes accurately.

Chapter 3

Audio-Visual Integration For Speech Recognition

3.1 Integration Methods

One of the main challenges in AV-ASR systems is the audio-visual information integration problem. The main issues in information integration are, (a) the class conditional dependence assumption made across streams, (b) the level (e.g. frame, phone, word) of integration, and (c) the kind (e.g. feature, partial likelihood, partial decision) of integration. Generally speaking, the following facts add complexity in combining audio with video for speech recognition.

- Audio and Video features have different dynamic ranges.
- Audio and Video features have different number of distinguishable classes. In other words, there are different number of phonemes than the number of visemes.
- Due to complexities involved in articulatory phenomena there is a time offset between audio and video signals.
- Video signal is usually sampled at a slower rate than the audio, and therefore, needs to be interpolated.

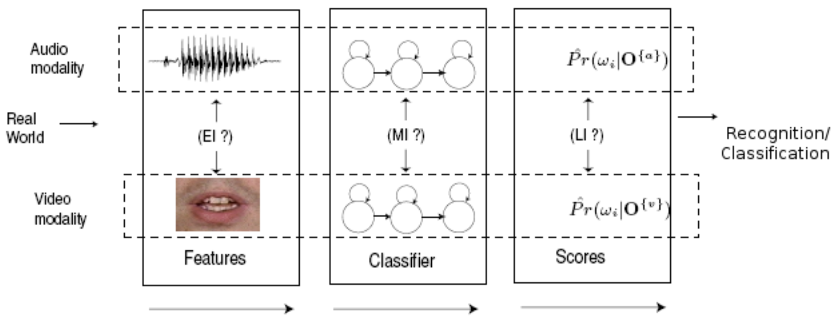

Audio modality
Real World
Video modality
$\hat{Pr}(\omega_i|\mathbf{O}^{\{a\}})$
(EI ?)
(MI ?)
(LI ?)
$\hat{Pr}(\omega_i|\mathbf{O}^{\{v\}})$
Features
Classifier
Scores
Recognition/ Classification

Figure 3.1: Early and Late Integration are widely acceptable in AV-ASR literature in contrast with Mid Integration which has unclear bounds with respect to LI. In some cases, LI and MI are considered to be same.

There is some confusion over terminology in AVSR concerning the different levels of possible integration . It is widely acceptable however, that the acoustic and visual speech modalities can be combined either at the feature (i.e. Early Integration) or the score level (i.e. Late Integration). However, strict definitions for EI and LI still remain unclear in AV-ASR literature. For example, in continuous audio-visual speech applications, LI is considered as combining scores at the sentence level in contrast with isolated word recognition where score combination for the same kind of integration takes place at the word unit level. On the other hand, EI schemes make use of a single feature space which represents both visual and audio information sources.

3.1.1 Early Integration

Early Integration (EI) or *Feature Fusion*, assumes class-conditional dependence between streams and frame synchronous information integration. Audio and visual features are computed from the acoustic and visual speech respectively and they are combined before the recognition experiment. Since the two set of features correspond to different feature spaces, they may differ in their characteristics . EI techniques for audio-visual speech are of benefit as they model the dependencies between acoustic and visual speech modalities directly. However, EI approaches suffer in two respects. First, if the acoustic or visual speech modalities are corrupted then the entire speech modality is corrupted due to classification occurring at such a low level. Second, there is an assumption that the acoustic and visual speech modalities are synchronized at the state level when HMM classifiers are being employed.

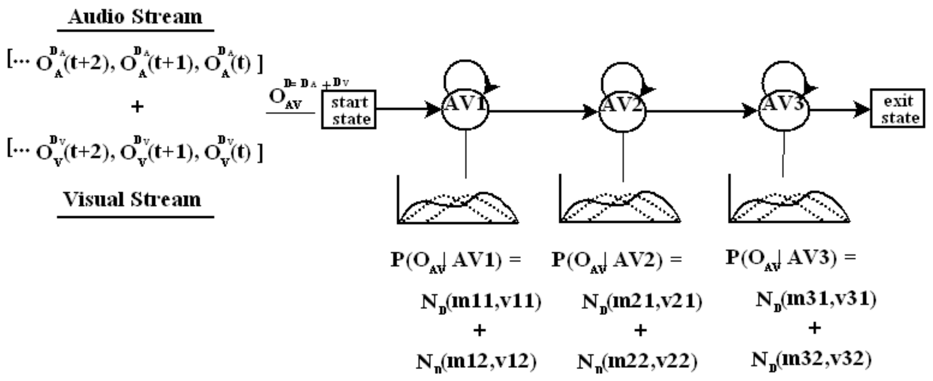
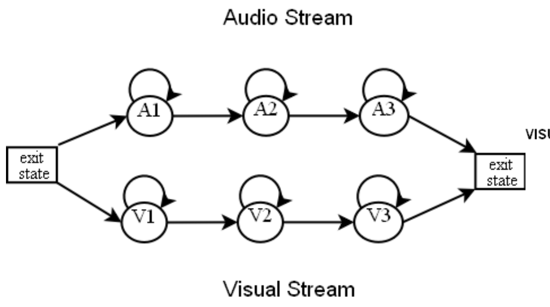

Audio Stream
[··· $O_A^{D_A}(t+2), O_A^{D_A}(t+1), O_A^{D_A}(t)$]
+
[··· $O_V^{D_V}(t+2), O_V^{D_V}(t+1), O_V^{D_V}(t)$]
Visual Stream
$O_{AV}^{D=D_A+D_V}$
start state
AV1
AV2
AV3
exit state
P(O_AV| AV1) = N_D(m11,v11) + N_D(m12,v12)
P(O_AV| AV2) = N_D(m21,v21) + N_D(m22,v22)
P(O_AV| AV3) = N_D(m31,v31) + N_D(m32,v32)

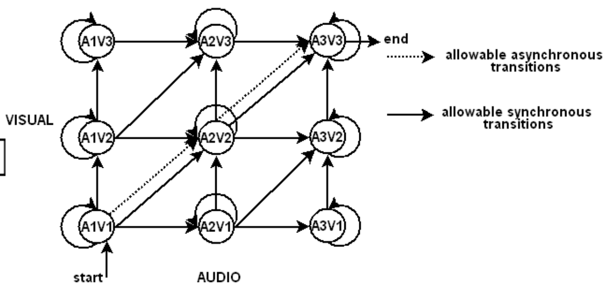
Figure 3.2: Early Integration Scheme for AV-ASR

It has been proved that acoustic and visual speech stimuli are not synchronous, at least at a feature based level because visible articulators, during an utterance, start and complete their trajectories asynchronously, exhibiting both forward and backward coarticulation with respect to the acoustic speech wave. Intuitively this makes a lot of sense, as visual articulators (i.e. lips, tongue, jaw) have to position themselves correctly before and after the start and end of an acoustic utterance. This time delay is known as the voice-onset-time (VOT), which is defined as the time delay between the burst sound, coming from the plosive part of a consonant, and the movement of the vocal folds for the voiced part of a voiced consonant or subsequent vowel. McGrath also found an audio lead of less than 80ms or lag of less than 140ms could not be detected during speech [9]. This kind of asynchrony cannot be modeled applying EI which assumes class-conditional dependence between streams.

Figure 3.2 demonstrates how EI can be applied. Suppose that $o^{(t)} = \left[o_A^{(t)}, o_V^{(t)}\right] \in R^D$is the concatenated observation vector obtained in time (t), where $D = D_A + D_V$. The class/state conditional observation probabilities of a sequence of such features is given by



(a) A multi-stream HMM



A1V3
A2V3
A3V3
end
allowable asynchronous transitions
allowable synchronous transitions
VISUAL
A1V2
A2V2
A3V2
A1V1
A2V1
A3V1
start
AUDIO

(b) A multi-stream HMM

Figure 3.3: A Multi-Stream HMM and its product. Lattice paths, starting from the down-left corner and ending at the up-right one, represent different levels of asynchrony may be captured. Diagonal path is for the state-synchronous HMM

$$P\left[o^{(t)} \mid c\right] \;=\; \sum_{j=1}^{J_c} w_{cj} \cdot N_D\left(o^{(t)};\, m_{cj}; v_{cj}\right),$$

where $c \in C$ is the class indicator, J_c is the number of Gaussian components of the mixture for class/state c, w_{cj} is the weight for j^{th} Gaussian component and finally m_{cj}, v_{cj} represent the mean value and the variance of the Gaussian component.

3.1.2 Late Integration

Late Integration (LI) or *Decision Fusion*, incorporates separate recognizers for audio and video channels and then combines the outputs of the two recognizers to get the final result. The final step of combining the two outputs is the most important step in this approach, as it has to deal with the issues of orthogonality between the two channels and the reliability of the channels. The level in which decisions can be fused, is a variable parameter that offers potential solutions. Many different levels have been proposed such as state level, phone level or word level. For automated isolated word applications LI strategies have reported superior results to EI for speech recognition tasks.

LI allows for the asynchronous classification of speech and can emphasize or deemphasize the importance of a modality in classification depending on the corruption present. The latter option is feasible by making use of stream weights in the state (class) conditional observation likelihood of the multi-stream HMM which is a weighted product of the partial stream observation likelihoods. Given the bimodal observation vector $o^{(t)}$, the state emision probability of the multi-stream HMM is ,

$$P\left[o^{(t)} \mid c\right] \;=\; \prod_{s\in\{A,V\}} \left[\sum_{j=1}^{J_c} w_{cj} * N_D\left(o_s{}^{(t)};\; m_{scj;}\, v_{scj}\right)\right]^{\lambda_{sct}} \tag{3.1}$$

,where $\lambda_{sct}$are the non-negative stream weights with $\lambda_{Act} + \lambda_{Vct} = 1$, which are also dependent to the modality s, the HMM state (class) c, and the time t. ***Instantaneous Stream weight estimation*** [4] makes use of stream reliability estimators for computing frame-level stream weights. In this work, we consider word or utterance level, modality dependent weights. The formula above is known as the *weighted product rule* which offers great facilities on emphasizing or deemphasizing the importance of a modality depending on the stream reliability. So far, it was assumed that audio and visual recognizers were state-synchronous. Let consider a multi-stream HMM which allows some degree of asynchrony between the two streams.

Figure 3.3 illustrates a multistream HMM and its product HMM in which different levels of synchronization between the two streams are possible.

A state synchronous decision fusion was considered for this project by combining each stream's likelihoods at a state level using the weighted product rule described above. HTK capabilities for multi-stream HMM were applied to train and evaluate our models. This process is described in section 4.4.1. The next section deals with stream weight estimation which is crucial for modeling stream reliability using the weighted product rule.

3.2 Stream Reliability Estimation

3.2.1 Entropy of A Posteriori Probabilities

This approach estimates stream confidence by computing the mean entropy over all the class-conditional a posteriori probabilities that occur in stream, for a specific frame in time. The computed entropy values are inversely proportional to the stream confidence. When all classes have almost the same probabilities then, entropy reaches its max value and subsequently the stream is considered to be totally unreliable. The same process is applied for both audio and visual streams and then, a function which combines their confidence measures is searched for the estimation of the final weights, which emphasize or deemphasize each stream contribution to the final decision. It has been shown that entropy measures should be avoided for time frames in which silence model states are among the 4 most probable states. The average entropy over N time frames and K number of speech recognition classes is

$$H = -\frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{K} P\left(c_{i,t} \mid o_{s,t}\right) \log P\left(c_{i,t} \mid o_{s,t}\right), \tag{3.2}$$

where $P\left(c_{i,t} \mid o_{s,t}\right)$ determines the a posteriori probability of the observed vector $o_{s,t}$, for class i, in time frame t. Various mappings have been proposed for relating entropy measures with stream weights. An efficient way to find a non linear mapping is proposed in [5]. A histogram of the last N entropy metrics is built to find which values occur more often. Then, a linear function which maps large entropy values to small weights and vice versa is adapted to the histogram in such way that its slope increases for frequent entropy values. An obvious drawback to using this kind of mapping is the loss of weight adaptiveness in each frame, due to computation of histogram, which needs a reasonable amount of entropy metrics before altering the mapping function.

3.2.2 N-best Log-Likelihood Difference/Dispersion

The distribution of the N-best log likelihoods in time t, consists a measure for the class discrimination in stream s, based on the observation. A reasonable choice for capturing such discrimination is either to compute the log likelihood difference average between the biggest log likelihood and the rest N-1 best values (see Eq. 3.3), or alternatively to find the differences between each possible likelihood couple of the N-best and then take an average (see Eq. 3.4). Suppose that $R_{s,t,n} = \log P\left(o_{s,t} \mid c_{s,t,n}\right)$ is the n^{th} best class-conditional log likelihood in stream s, in time frame t. Subsequently, $R_{s,t,1} = \log P\left(o_{s,t} \mid c_{s,t,1}\right)$ corresponds to the best log likelihood. The next two equations describe the reliability measures of N-best *Log-Likelihood Difference* and *Dispersion* respectively, for stream s, in time frame t.

$$Diff_{s,t} = \frac{1}{N-1}\sum_{n=2}^{N}\left(R_{s,t,1} - R_{s,t,n}\right) \tag{3.3}$$

$$Disp_{s,t} = \frac{2}{N(N-1)} \sum_{n=1}^{N} \sum_{n'=n+1}^{N} \left(R_{s,t,n} - R_{s,t,n'}\right) \tag{3.4}$$

where $N \geq 2$. For both (3.3), (3.4), "large" values indicate high stream confidence. A comparison of the three metrics above was made in [12], in which proved that eq.(3.4) gave the best results for a Large Vocabulary Speech Recognition (LVSR) task. Finally, eq.(3.3) was applied in [4] for the instantaneous stream weight estimation in AV-ASR, over the CUAVE database.

3.2.3 K-means Clustering Method

In the presence of modeling or estimation error in a multi-stream classification problem, stream weights can decrease total classification error. For a two-stream, two-class problem, the optimal weights have been shown to be inversely proportional to the single-stream pdf estimation error [11].

$$\frac{s_1}{s_2} = \frac{\sigma_{s2}^2}{\sigma_{s1}^2}, \tag{3.5}$$

where σ_s^2 defines the variance of the pdf estimation error for stream s. Subsequently, the optimal stream weights are inversely proportional to the variance of the pdf estimation error.

In the same work, it has been also proved that assuming equal pdf estimation error variances for the two streams, the next relation stands,

$$\frac{s_1}{s_2} \approx \frac{p(x_2 \mid c_1)}{p(x_1 \mid c_1)} \; for \; 0.5 \leq \frac{p(x_2 \mid c_1)}{p(x_1 \mid c_1)} \leq 1.5, \tag{3.6}$$

which means that, optimal stream weights are not a function of the estimation error if $\sigma_{s1} = \sigma_{s2}$ but, in a region of interest, as defined in (3.6), stream weights should be inversely proportional to the single-stream a posteriori probabilities $p(x_s \mid c_1)$, where x_s is a feature vector of stream s and c_1 stands for class 1. Moreover, the ration in eq.3.6 can be approximated by the single-stream classification error ration.

$$\frac{p(x_2 \mid c_1)}{p(x_1 \mid c_1)} \approx \frac{100 - WACC(\lambda_2, D_{test})}{100 - WACC(\lambda_1, D_{test})}, \tag{3.7}$$

where $100 - WACC(\lambda_s, D_{test})$ represents the stream s classification error for the testing set D_{test} using models $\lambda$and $WACC$ stands for "word accuracy".

A combination of equations (3.6) and (3.5) was tested for the stream weight estimation in the AV-ASR problem with various levels of audio stream corruption. The comparison between the optimal weights and the estimated ones showed that they were strongly correlated for Signal-to-Noise (SNR) values where the stream classification errors were almost the same, as defines the condition in Eq.(3.6). The correlation coefficient reached the value of 0.96. The following experimental formula was used for this test,

$$\frac{s_1}{s_2} \approx \frac{\sigma_V^2}{\sigma_A^2} \frac{100 - WACC\left(\lambda_v, D_{test}\right)}{100 - WACC\left(\lambda_A^{snr}, D_{test}\right)}. \tag{3.8}$$

There are two major issues we need to overcome if we want to apply the formula above for the unsupervised stream weight estimation in AVSR.

1. Only the two-class classification problem has been examined so far, while we need a knowledge base for the multi-class problem

2. Knowledge of stream classification error for the test data is necessary but this consists a causality violence.

A solution for the both issues is given in [15], where the multi-class problem is reposed as many two-class problems which are defined as model vs "anti-model" problems and also, stream classification error is considered to be a non-linear function of the inter-to-intra class distance ratio. "Anti-models" are described as the background of real models. Training data are separated into two groups; one containing the training samples of the class of interest, and the other containing the rest of the training samples. Models and "anti-models" are built from the two training sets. A detailed description about how the concept of models and "anti-models" is used for the AV-ASR, is illustrated in Chapter 4.

To resolve the second issue, the single-stream classification error for the two-class problem, was treated as a function of $D = \mid \mu_1 - \mu_2 \mid / \sigma$ [1] supposing that $p\left(x \mid c_i\right)$ follow Gaussian distributions $N\left(\mu_i; \sigma^2\right)$. An approximation of D was made by applying *K-means clustering* and then using the inter- and intra-class distances to estimate the

[1]Actually, Bayes error is estimated as a function of $D = \mid \mu_1 - \mu_2 \mid / \sigma$ in a two-class classification problem

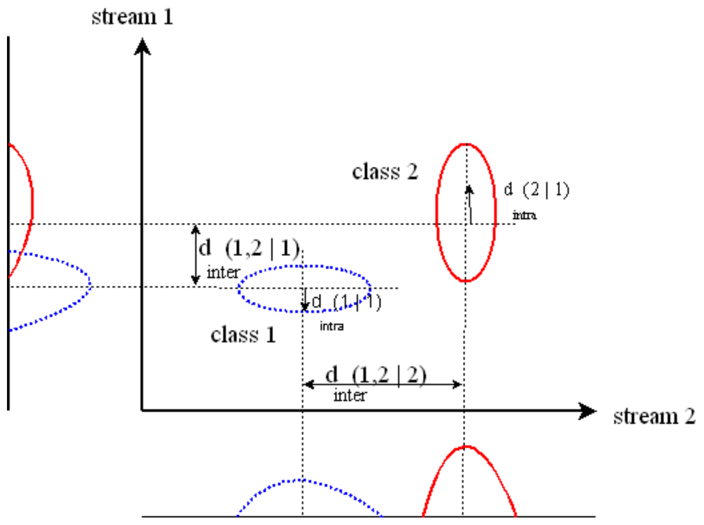


Figure 3.4: An illustration of inter-, intra-class distances. They seem to be inversely proportional and proportional respectively to the Bayes error.

quantities in the nominator and the denominator respectively. The *inter-class distance* $d_{inter}(i, j \mid s)$ is defined as the distance between the means (centroids) of classes i, j in stream s [2], while the *intra-class distance* $d_{intra}(i \mid s)$ is defined as the variability within a class i in stream s [3]. Figure (3.4) illustrates the meaning of inter-, intra-class distances. The relationship between the Bayes error and the inter- , intra-class distances are inversely proportional and proportional respectively. Overall, the next formula estimates the stream weight ratio for a two-class, two-stream classification problem.

$$\frac{s_1}{s_2} \approx a \cdot f\left(\frac{d_{inter}(1,2 \mid 1) / \left[d_{intra}(1 \mid 1) + d_{intra}(2 \mid 1)\right]}{d_{inter}(1,2 \mid 2) / \left[d_{intra}(1 \mid 2) + d_{intra}(2 \mid 2)\right]}\right) \tag{3.9}$$

We test the estimation accuracy which K-means approach can provide for a two-stream, two-class classification problem, with artificial data. Particularly, we assume a one-dimension feature space, where data are generated using $N(\mu_{s,c}; \sigma^2)$ normal distributions, for each class c, in stream s. Our aim is to investigate if K-means approach can correctly estimate the second part of Eq.(3.7). Firstly, data are generated in order to fulfil $\mu_{1,1} - \mu_{1,2} = \mu_{2,1} - \mu_{2,2}$ which indicates an equal bayes error for the two streams. Note that

[2] $d_{inter}(i, j \mid s) = \mid \overrightarrow{\mu_{s,i}} - \overrightarrow{\mu_{s,j}} \mid$

[3] $d_{intra}(i \mid s) = \frac{1}{N} \sum_{o=1}^{N} \left(\overrightarrow{x_{s,i,o}} - \overrightarrow{\mu_{s,i}}\right)^2$,where $\overrightarrow{x_{s,i,o}}$ is a feature vector of class i, stream s.

both classes for both streams have the same variance σ^2, which subsequently means that all single-stream classes have equal intra-class distances. So, according to (3.7), stream weights are expected to be equal ($\frac{s_1}{s_2} \approx 1$). After K-means is applied, it is expected to see that the formed clusters will have

$$d_{inter}\left(1,2 \mid 1\right) = d_{inter}\left(1,2 \mid 2\right)$$

and

$$d_{intra}\left(1 \mid 1\right) + d_{intra}\left(2 \mid 1\right) = d_{intra}\left(1 \mid 2\right) + d_{intra}\left(2 \mid 2\right).$$

If this happens, Eq.(3.9) successfully estimates the ratio of stream weights. If we assume that $\mu_{1,1} - \mu_{1,2} > \mu_{2,1} - \mu_{2,2}$, classification error in stream 1 is expected to be reduced and subsequently, its weight will be increased. Finally, we increase the class separation in stream 1 for various amounts, in order to measure the accuracy of the stream weight estimations. In Fig.(3.5), it is clear that the more stream classification errors differ, the less accurate estimations K-means provide. Moreover, when classification errors are close, with respect to the condition of Eq.(3.6) , it seems that K-means estimations reach a good level of similarity with the realistic classification error ratio. Finally, when applying a mapping to the estimates, of a type $c \cdot (estratio)^b$, the estimation curve (the dashed line in fig. 3.5.b) gets even more closer.

3.3 Weighting Schemes for Audio-Visual Fusion

In the previous sections, it was described in what ways it is possible to fuse audio with visual stream using HMM. It is widely accepted that employing weighted product rule (see Eq. 3.1) for combining partial decisions made from seperate audio and visual recognizers, is a good choice. Multi-stream HMM allow different levels of asynchrony between audio and visual streams. Phone-level or word-level boundaries are generally considered for fusing partial decisions. Stream Weight Estimation seems to be crucial

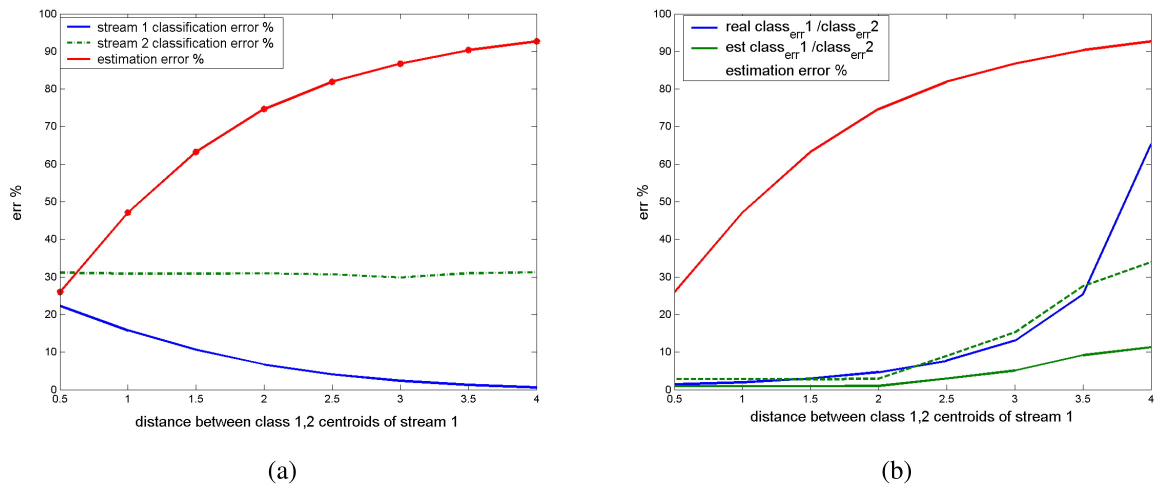


Figure 3.5: Simulation results using artificial data. Obviously, the estimations of $\frac{p(x_2|w_1)}{p(x_1|w_1)}$ are less accurate as the classification error difference between the two streams increases.

for emphasizing or deemphasizing a stream's membership in the final decision. Moreover, weight adaptation to time-varying noise conditions outperforms any static weighting schema. Stream Confidence indicators, as described in section 3.2, are based on data observations of audio-only or both streams. We chose a K-means based, unsupervised method to estimate stream weights for the classification and recognition task, which are described in sections 3.3.2 and 3.3.2 respectively. Most of the methods that make use of stream reliability indicators, need a mapping function to transform indicators into real weights. Section 3.3.3 represents a mapping schema for our method.

3.3.1 Unsupervised Stream Weight Computation for Digit Classification

Since audio-visual HMM recognizer has been set up and trained, as described in Chapter 4, CUAVE database isolated-digits of "task 1" are used to verify classifier's performance with respect to weight estimation. We repeat the same test for all training/testing set combinations of table (4.2). The Results obtained form these experiments are presented in Chapter 5. The remainder of this section describes the process we follow to estimate stream weights for a classification task.

The theoretical results in subsection 3.2.3 are not directly applicable to classification problems for two main reasons: (i) only results for the two-class classification problem are available, while in general the multi-class classification problem is of interest, and (ii) knowledge of class membership for each observation vector and/or the single-stream modeling/estimation error is required to compute the weights

To solve the first problem, we used the principle of anti-models in [15]. By creating models and anti-models, the multi-class classification problem is transformed to multiple two-class classification problems. To solve the second problem, we approximated in [15] the ratio in Eq.(3.6) by measurable quantities that can be computed in an unsupervised manner. Indeed, it is well known that, if $p\left(x \mid c_i\right)$ follow a Gaussian distributions $N\left(\mu_i; \sigma^2\right)$, the Bayes error is a function of $D = \mid \mu_1 - \mu_2 \mid / \sigma$, which can be computed in an unsupervised manner using the inter- and intra-class distances (see Fig.3.4). The intra-class distance is an estimate of the average class variance and the inter-class distance is the average distance between the means of each class. These distances are used to measure the separation and the overlapping between the class distributions, and they are directly dependent of the centroids of each class. The Gaussian-means of the (HMMs) model and anti-model are used to initialize the K-means algorithm (K=2) for each class to obtain the centroids and then compute the inter- and intra-distances (see Fig.3.4).

We consider ten different digit classes to construct HMMs with 8 states and a single Gaussian continuous density distribution per state. An important part of the training process is the generation of "anti-models". The class and anti-class models are both built in the training process using "clean" data. Class models for each stream are built following the traditional process. On the other hand, anti-class models are trained using all the data that do not belong in the corresponding class. For example, the model for digit one is created using all training data labeled as "one", while the anti-digit one is created using all the data not labeled as "one". At the end of this process, 20 models are obtained for each stream, ten models for the digits (0-9) and ten anti-digits all with the same number of parameters.

During the test phase, these class and anti-class models are used to initialize K-means classification. Specifically, the means of the Gaussian distribution in the class and anti-class models, are used as the initial K-means centroids. Given that a-priori it is not known to which class each utterance (digit) belongs, the features in each utterance are split into two classes (k=2) in ten different ways one for each digit and anti-digit model. In consequence, one k-means algorithm have to be performed for each couple model-antimodel in order to obtain d_{inter}, d_{intra} quantities. Stream weights can be obtained as the solution of the next linear system.

$$\frac{W_{aud}}{W_{vis}} \approx a \cdot f\left(\sum_K \frac{d_{inter}\left(m_k, am_k \mid s\right) / \left[d_{intra}\left(m_k \mid s\right) + d_{intra}\left(am_k \mid s\right)\right] \mid s = Audio}{d_{inter}\left(m_k, am_k \mid s\right) / \left[d_{intra}\left(m_k \mid s\right) + d_{intra}\left(am_k \mid s\right)\right] \mid s = Visual}\right), \tag{3.10}$$

$$W_{aud} + W_{vis} = 1$$

where $m_k,\ am_k$ are the mean vectors (centroids) of class K and d_{inter}, d_{intra} are the inter-, intra-class distances respectively, as they have been defined in section 3.2.3. The inter-, intra-class distances d_{inter}, d_{intra} are computed for each of the ten splits and the resulting inter- to intra-class ratio is averaged over the ten spilts. Note that stream weights are estimated in an utterance (digit) base. To take into account the difference between pdf estimation error in two streams, a constant a can be estimated on a held-out data set and used to improve the results. Additionally, $f(.)$ is a non-linear function that relates the metric obtained from k-means with the real bayes error. Further details can be found in section 3.3.3 about how a and $f(.)$ are efficiently computed .

3.3.2 Unsupervised Stream Weight Computation for a small Vocabulary Continuous Speech Recognition problem

In this Section, we extend the multi-class classification approach presented in the previous section to the recognition problem for a small Vocabulary AV database. Generally speaking, recognition is considered to be more difficult than classification but in this case, practically, weight estimation is actually easier, without the need of anti-models. That is due to the observation that larger separation between class distributions in a given stream increases its confidence computed by stream reliability estimation algorithms described in previous sections.

The same K-means approach is applied, as described in section 3.2.3. Intra-, inter-class distances are computed, as before, in order to measure class separation. The only difference is that K-means is performed only once, over all k classes, initialized with the precomputed means (centroids) of k classes obtained form “clean data”. The total inter-class distance $d_{inter}\left(s\right)$, in stream s, is computed by adding the inter-class distances over all the possible combinations of two classes,

$$d_{inter}\left(s\right) = \sum_{k=1}^{K} \sum_{l=k+1}^{K} d_{inter}\left(k, l \mid s\right), \tag{3.11}$$

where K is the total number of classes.

Overall, the stream weights are computed like Eq.(3.10) but using the total inter-class distance $d_{inter}\left(s\right)$ normalized by the sum of intra-class distances $d_{intra}\left(s\right) = \sum_{k=1}^{K} d_{intra}\left(k \mid s\right)$ in stream s.

$$\begin{aligned} &\frac{W_{aud}}{W_{vis}} \approx a * f\left(\frac{d_{inter}\left(s\right)/d_{intra}\left(s\right) \mid s = Audio}{d_{inter}\left(s\right)/d_{intra}\left(s\right) \mid s = Visual}\right), \\ &W_{aud} + W_{vis} = 1 \end{aligned} \tag{3.12}$$

Experiments are performed over the CUAVE AV database by separating speakers into training and testing sets which include 30 and 6 speakers respectively, each uttering 50 connected digits. Audio signal was corrupted by various additive noises (see Sec. 4.2) at various SNR levels. Feature extraction is performed as described in section 4.3 and finally, HMM are trained and tested using HTK capabilities (see Sec. 4.5). Recognition results for the round-robin the testing sets (see table 4.2) are represented in section 5.2. It has to be mentioned that the conceptual base for this experiment has produced in [14].

3.3.3 Stream Reliability to Stream Weight Mapping

A Non-linear function $f\left(.\right)$ is used in order to map estimated stream classification error ratio of equation (3.9) and (3.12) into stream weight ratio. A constant value a is also used for taking into account the class-conditional pdf estimation error difference between the two streams, as described in Eq.(3.7). Note that constant a is included in the mapping function as a parameter. Overall, four types of mapping are tested in order to find which one gives better results for different types of noise and SNR levels,

Note that in (Type II) and (Type IV), it is assumed that visual stream reliability remains constant for changing noise and SNR levels. Subsequently, this constant is considered as a mapping parameter embedded in a. To compute the parameters a , b or only a for each type of mapping, we perform a *fitting* between the estimated and optimal weight ratios

$$
\begin{array}{ll}
a \cdot \sqrt{\frac{AudReliabilty}{VisReliability}} & \text{(Type I)} \\
a \cdot \sqrt{AudReliability} & \text{(Type II)} \\
a \cdot \left(\frac{AudReliabilty}{VisReliability}\right)^{b} & \text{(Type III)} \\
a \cdot Audreliability^{b} & \text{(Type IV)}
\end{array}
$$

Table 3.1: The Mapping Types we tested

Adapted Mapping				
NOISE / TYPE OF MAPPING	$a \cdot \left(\frac{AudRel}{VisRel}\right)^{b}$	$a \cdot \sqrt{\frac{AudRel}{VisRel}}$	$a \cdot Audrel^{b}$	$a \cdot \sqrt{AudRel}$
babble	45.82	***22.68***	31.73	23.72
white	33.66	14.24	***10.41***	12.26
jet	34.56	14.14	***13.07***	13.56
volvo	21.56	7.11	***5.88***	5.31
factory	46.90	18.87	25.83	***18.5***
mean	*36.50*	*15.41*	*17.38*	***14.6***

Table 3.2: Weight Estimation Error % for various mappings. Bold Values correspond to the minimum estimation error and indicate the best type of mapping for each noise.

for the training data using those parametric mapping functions. After training each type of function, we test their performance over the testing data by comparing the recognition results. Weights are estimated for every sentence of ten digits but only a mean value is selected and applied to the synchronous multi-stream HMMs, in order to compare the results for different mapping functions.

Two different approaches are considered in order to compute the mapping parameters. In the first, we assume that each mapping is trained separately for a specific type of noise and data set in order to get adapted to it. This approach achieved better weight estimations but only if we have the knowledge for the type of noise. Table (3.2) represents the weight estimation error for each kind of mapping over five different noisy data. Mapping parameters were trained on a training set of 30 speakers and tested for six speakers, as mentioned before. The same process repeated for different partitions of the available data into training and test sets. The best average performance achieved by $a \cdot \sqrt{AudRel}$. In

Global Mapping				
NOISE / TYPE OF MAPPING	$a \cdot \left(\frac{AudRel}{VisRel}\right)^b$	$a \cdot \sqrt{\frac{AudRel}{VisRel}}$	$a \cdot Audrel^b$	$a \cdot \sqrt{AudRel}$
babble	38.83	***25.90***	33.42	27.44
white	38.51	12.2	14.09	***11.32***
jet	45.00	14.38	18.35	***13.82***
volvo	23.46	11.45	9.14	***8.19***
factory	42.85	***19.44***	25.22	20.20
mean	*37.73*	*16.67*	*20.04*	***16.19***

Table 3.3: Weight Estimation Error % for global mapping. The same types of mapping were trained to perform well for all noises.

the second approach, all the mapping parameters are trained using a global training set in order to perform the fitting between the reliability estimators and the stream weights. In this case, 7500 numerical data (30 speakers x 50 digits x 5 noises) are available in order to estimate the parameters which give the “best fit” for each kind of mapping function. The best fit is evaluated by using the least squares method and subsequently, the performance of each type of mapping is tested over a test set of six speakers. The weight estimation error for the “global mapping” is represented in table (3.3).

Chapter 4

Experiment Setup

In the following sections, it is described the process of creating, training and testing Audio-Visual HMM recognizers using HTK [1], in order to perform experiments for the unsupervised stream weight estimation in different noisy environments. Furthermore, a Visual recognizer is built for testing how informative features were extracted from the video frames using the feature optimization process which described in Chapter 2 . Finally, an audio recognizer is set up to be compared with the AV recognizer for the classification and recognition tasks.

4.1 CUAVE Database description

Clemson University Audio Visual Experiments (CUAVE) corpus is an audio-visual database that has been designed to meet some criteria. It is a speaker independent database of connected (or isolated) and continuous digit strings of high quality audio and video of a representative group of speakers. Recordings have been made in realistic conditions for testing robust audio-visual schemes. For example , speaker's movement is a physical parameter that exists in every real application and cannot be avoided. As table 4.1 depicts, the database is grouped in tasks. Every task is offered for a different kind of experiment.

There are 36 speakers with different kind of facial characteristics and skin tones (see fig.4.1). The fact that there is a variation in skin colours, makes our face detection method more robust. Task 1 includes stationary speakers that form a set of 36×50 connected digits. We will use Task 1 for our experiments in visual feature extraction as there is always a small movement or rotation of the speaker's head, which is desirable if we want

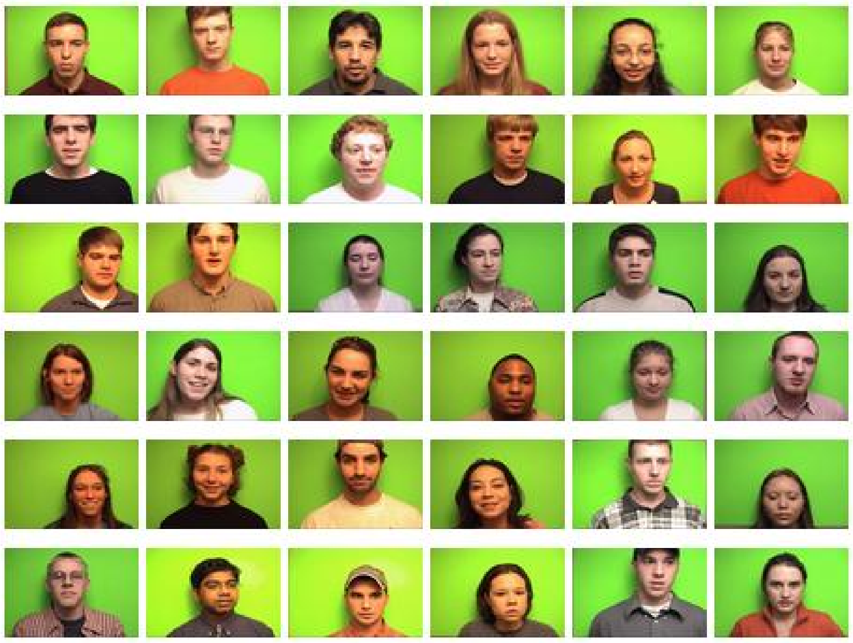


Figure 4.1: CUAVE includes speakers with different facial characteristics

to investigate rotation correction and scaling normalization.

Video was recorded at 720x480 resolution with 29.97 fps (NTSC) in full colour. It means that feature vectors for the visual stream are extracted in a rate of 30 vectors per second in contrast with audio features which are extracted in a rate of 100 vectors per second. So, visual feature vectors need to be interpolated in order to match with audio features. The sound was recorded in 16-bit stereo at 44 Khz. Also 16-bit mono wav files at 16 KHz are available to use in our experiments. A detailed description of the database can be found in [10].

4.2 Noise types

In order to test Audio-only and Audio-Visual recognizers in different noisy environ-

Part	Task	Movement	Numberof Digits	Mod
1.Individual	1	Still	50x36 speakers	Connected
	2	Moving	30x36 speakers	Connected
	3	Profile	20x36 speakers	Connected
	4	Still	30x36 speakers	Continuous
	5	Moving	30x36 speakers	Continuous
2.Pairs	6	Still	(30x2)x20 pairs	Continuous

Table 4.1: CUAVE Task Description

	Speakers
Testing set 1	s01m, s02m, s04f, s20f, s33m, s34f
Testing set 2	s03m, s05f, s06f, s19f, s21f, s32m
Testing set 3	s09m, s10m, s17m, s27m, s35m, s36f
Testing set 4	s07m, s12m, s14m, s23f, s28f, s29m
Testing set 5	s08m, s11f, s16f, s22m, s27m, s31m
Testing set 6	s03m, s15f, s18f, s24m, s25f, s30f

Table 4.2: Separation of the initial data set to testing sets for different independent expirements. For each testing set, models are trained with the rest of the data.

ments, we simulate this audio degradation by injecting different kinds of noise in the ”clean” audio. The noise audio files were acquired from the Signal Processing Information Base (SPIB) repository (http://spib.rice.edu/spib.html) and were downsampled from 20KHz to 16KHz which is the sampling rate of ”clean” audio. We used five types of noise:

Speech Babble is the source of 100 people speaking in a canteen. The room radius is over two meters, therefore, individual voices are slightly audible. The sound level during the recording was 88 db.

White Noise was acquired by sampling high-quality analog noise generator (Wandel & Goltermann). Exhibits equal energy per Hz bandwidth.

Jet Cockpit Noise 1 was acquired from the interior of a Buccaneer Jet moving in a speed of 190 Knots, and an altitude of 10000 feets, with airbrakes out. The sound level during the recording process was 109 dB.

Car Interior Noise was recorded while driving a Volvo 340 at the speed of 120 Km/h, in 4^{th}gear, on an asphalt road, in rainy conditions.

Factory Floor Noise 1 was recorded near a plate-cutting and electrical-welding equipment.

These types of noise constitute different scenarios were AV-ASR could be applied. It is worth mentioning that SNR was considered as the speech to noise ratio taking into account their global energies. Different values of SNR were obtained by adjusting noise global energy before it was added to the audio signal. Alternatively, signal to noise ratio can be adjusted locally, in segments.

4.3 Features

A subset of the CUAVE audio-visual database [10] was employed for these experiments as described in section 4.1. The process of feature extraction for audio and visual data, has been already analyzed in Chapter 2. We use the standard Mel-Frequency Cepstrum Coefficients (MFCCs) as audio features, computed for frames with duration 25ms, extracted every 10ms. The acoustic vectors of dimension $d_A = 39$ consist of 12-dimensional MFCCs, energy, and their first and second order derivatives. Respectively, visual features were extracted from the scale, rotation normalized mouth region after down-sampling and performing 2-D Discrete Cosine Transform (DCT). The first 35 most ”energetic” DCT coefficients were kept resulting in a feature vector of dimension $d_V = 105$ including the first and second order derivatives. Both audio and visual features are mean normalized by subtracting their mean value.

4.4 HMM recognizers

Speech recognition systems generally assume that the speech signal is a realisation of some message encoded as a sequence of one or more symbols. To perform the reverse operation of recognising the underlying symbol sequence given a spoken utterance, the continuous speech waveform is first converted to a sequence of feature vectors. The role of the recogniser is to apply a mapping between sequences of speech vectors and the wanted underlying symbol sequences.

Two problems make this very difficult. Firstly, the mapping from symbols to speech is not clear since different underlying symbols can give rise to similar speech sounds. Furthermore, there are large variations in the realised speech waveform due to speaker variability, mood, environment, etc. Secondly, the boundaries between symbols cannot be identified explicitly from the speech waveform. Hence, it is not possible to treat the speech waveform as a sequence of concatenated static patterns.

4.4.1 HMM creation and training

Three types of recognizers are built (Audio-only, Visual-only, Audio-Visual) as mentioned before depending on which kind of modality we use in order to perform classification/recognition. Context-independent whole-digit models with 11 states per digit and a single Gaussian continuous density distribution per state are defined using HTK [1] capabilities. The first and last state of each model is used for HTK purposes. Ten models are created and trained knowing that data are separated into ten classes (digits). In addition, a model for silence is built with 3 only states. The training procedure begins after data preparation (data conversion into HTK format) and prototype HMM definition (topology description of the models are going to be trained). HMM definitions are stored as simple text files. Note that all training and testing data labels are provided. In addition, a dictionary and a task-grammar are necessary for HTK in order to have knowledge for the utterance structure during recognition. In our case, the dictionary consists of the ten digits and the word lattice describing the task-grammar is built with *HParse.* We provide HTK with the knowledge that sentences which are going to be recognized include a sequence of digits interleaved by silence segments. Note that audio and audio-visual recognizers are trained with ”clean” audio data. On the other hand, visual features stay unaffected by noise.

Each of the required HMMs is generated individually.

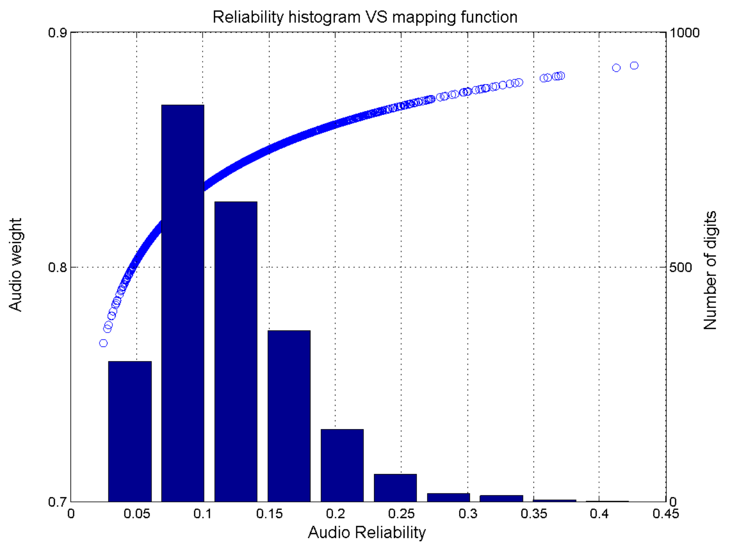
- *HInit* is used to provide an initial set of single-model parameter estimates. It reads in all of the labeled training data and cuts out all of the examples of the required digit. It then iteratively computes an initial set of parameter values using a segmental k-means procedure. On the first cycle, the training data is uniformly segmented, each model state is matched with the corresponding data segments and then means and variances are estimated. On the second and successive cycles, the uniform segmentation is replaced by Viterbi alignment.

- The initial parameter values computed by HInit are then further re-estimated by *HRest*. Again, the fully labelled training data is used but this time the segmental k-means procedure is replaced by the *Baum-Welch* re-estimation procedure.

HTK provides some capabilities for multi-stream HMMs but unfortunately those are limited. Only state-synchronous models with joinly estimated transition probabilities can be built. It is important to be mentioned that *HRest* does not take into account stream weights during training and so, the training result is suboptimal. Furthermore, state-level weight changeability is not supported. Each model uses a single set of stream weights for all the included states. We decided to train AV recognizer models using equal stream weights ($W_A = W_V = 0.5$). Finally, HTK implementation for multi-stream models manipulates a single vector for both streams.

4.5 Evaluation of Weight Estimation

After AV recognizer is set up and trained, its performance is compared with respect to the audio-only recognizer, for five different noisy environments we simulated by injecting noise in the audio data (as described in sec. 4.2). Audio is degraded for eight different levels of SNR (-10,-3,0,3,6,9,12,20). We split features into digits and sentences in order to perform weight estimation using K-means and then evaluate the results for classification and recognition respectively. Note that both results may be biased due to the implementation of K-means we use. It is well known that K-means is sensitive to the initial centroids which is provided and it is possible to give different results depending on the initialization. However, we use MATLAB's (6.5 edition) K-means impementation to estimate stream weights, following different processes for classification and recognition as described in sections 3.3.2, 3.3.2 respectively.



A mean value of the metric inside the parenthesis of Eq.(3.10) is computed for all the digits in the training set and then an optimal mapping function $f(.)$ is searched among the 4 types as depicted in table (3.1). This is achieved by selecting a function type which best estimates the optimal weights for the testing set. Note that the parameters of each function type are trained by fitting using the training data. Optimal weights for each type of noise and SNR value have been computed by hand. Only a pair of stream weights is estimated for each combination of noise and SNR. In other words, it is assumed that weights remain constant for every model during recognition/classification.

Chapter 5

Experimental Results

The result set contained in this chapter is divided in two parts. Firstly, we investigate what kind of improvements have been made after each stage of the visual front-end optimization process and then, we compare the performance of audio-only and audio-visual recognizer focusing on the efficiency of unsupervised weight estimation. Figures and tables are included for better understanding the behaviour of each recognizer (Audio-only, Visual, Audio-Visual) in various environments. Moreover, it is interesting to see the word error rate (WER) reduction achieved by the AV recognizer implementation with estimated stream weights in contrast with the baseline implementation (equal stream weights).

5.1 Visual feature extraction optimization

To avoid confusion, different notations are used for each visual front-end implementation. In *"Baseline-DCT"* visual feature extraction, it is applied a simple method for ROI tracking in which, a manually selected mouth region from the first frame is shifted in multiple directions within the second frame, in order to be matched using cross-correlation as a similarity measure. When searching stops, the best matched region is used as a template for the next frame and so on. 2D-DCT features are then extracted from the decimated ROI and the (0,0) coefficient is ignored to obtain, at the end, a feature dimension equal to 35 over the first 6×6 first DCT coefficients. These feature vectors are re-sampled to 100 FPS in such a way that they can be combined with the audio stream.

File	*%Cor*	*%Acc*	*Cor*	*Del*	*Sub*	*Ins*	*N*
s01m.rec	32.00	(32.00)	16	26	8	0	50
s02m.rec	40.00	(24.00)	20	10	20	8	50
s04f.rec	12.00	(12.00)	6	40	4	0	50
s20f.rec	18.00	(18.00)	9	36	5	0	50
s33m.rec	44.00	(42.00)	22	20	8	1	50
s34f.rec	26.00	(26.00)	13	32	5	0	50
———————Overall Results———————-							
WORD:	28.27	(25.67)	86	164	50	9	300

Table 5.1: Baseline-DCT recognition results for testing set 1

File	*%Cor*	*%Acc*	*Cor*	*Del*	*Sub*	*Ins*	*N*
s01m.rec	70.00	(58.00)	35	22	14	6	50
s02m.rec	78.00	(-8.00)	39	3	11	43	50
s04f.rec	44.00	(32.00)	22	30	12	6	50
s20f.rec	30.00	(22.00)	16	42	24	4	50
s33m.rec	80.00	(42.00)	40	33	10	19	50
s34f.rec	76.00	(64.00)	38	33	10	6	50
———————Overall Results———————-							
WORD:	63.33	(35.33)	190	29	81	84	300

Table 5.2: Nt-DCT recognition results for testing set 1, insertion penalty = -150

An alternative ROI tracking approach, also based on template matching, is applied in our front-end schema, as described in Chapter 2. Knowing that the mouth region is sensitive to intensity changes through time frames due to mouth-motion and thus template matching is expected to be inefficient, we avoided using the mouth region as a template and therefore, we applied nose tracking in order to track the ROI (see section 2.1.2). This nose-tracking-based implementation of the visual front-end is tested here, denoted as *"Nt-DCT"* , as well as its versions with rotation correction and scaling normalization, referred as *"Rc-DCT"* and *"Sn-DCT"* respectively.

HMM-based visual recognizers were trained and then tested for every combination

File	*%Cor*	*%Acc*	*Cor*	*Del*	*Sub*	*Ins*	*N*
s01m.rec	62.00	(58.00)	31	8	11	2	50
s02m.rec	66.00	(20.00)	33	1	16	23	50
s04f.rec	36.00	(34.00)	18	22	10	1	50
s20f.rec	26.00	(22.00)	13	26	11	2	50
s33m.rec	62.00	(58.00)	31	3	16	2	50
s34f.rec	66.00	(64.00)	33	7	10	1	50
————————Overall Results————————							
WORD:	53.00	(42.67)	159	67	74	31	300

Table 5.3: Nt-DCT recognition results for testing set 1, insertion penalty = -250

File	*%Cor*	*%Acc*	*Cor*	*Del*	*Sub*	*Ins*	*N*
s20f.rec	44.00	(42.00)	21	20	9	1	50

Table 5.4: Rc-DCT recognition results for speaker s20f, insertion penalty = -250

of training, testing sets as depicted in table (4.2). Table (5.1) reports the recognition accuracies obtained for the *"Baseline-DCT"* front-end, for testing set 1, using HVite of HTK for decoding and HResults to format the results. An HTK parameter called “word insertion penalty” is a fixed value added to each token when it transits from the end of one word to the start of the next, during decoding. Recognition accuracy depends on this value as it is obvious in tables (5.2) and (5.2), where the same data are tested with different insertion penalties. To measure the best performance, we rerun the experiments changing this parameter. Results can be improved about 20%. For instance, speaker s02, who tends to speak clearly, giving emphasis to his articulation, achieves the best score in audio, but for lipreading, there is a problem with insertions which can be reduced by increasing insertion penalty.

“Rc-DCT”	smoothing window = 3 frames	smoothing window = 5 frames
53.33%	54.67%	54.33%

Table 5.5: Rotation angles smoothing with mean filter

When using the “Nt-DCT” front-end, we observed a significant WER reduction over the “Baseline-DCT” which was measured about 70%. Additionally, “Rc-DCT” yield a 48% WER (52% accuracy), compared to the “Nt-DCT” 58% WER. This amounts to a further WER reduction of 25 %. On the average, “Rc-DCT” front-end gave a “boost” to performance of about 95%. Such a sample of improvement can be seen in table (5.4), in which the recognition accuracy for speaker s20 reached 42% in contrast with the 18% accuracy of the “Baseline-DCT” implementation. This means a 130% accuracy improvement. Actually, speaker s20 exhibited large head inclination angles. Finally, we applied a smoothing to the estimated angles of each frame in order to minimize improper estimates as well as erratic changes. As table (5.5) indicates, we achieved an improvement of 1% on average when applying a mean filter smoothing over the estimated rotation angles.

5.2 Stream Weight Estimation

The main purpose of this section is to report the results obtained from our research in stream weight estimation. The advantages, as well as the drawbacks of our method, are clearly understood through figures and tables, which summarize the raw results we got, using a variety of statistics. In first place, we have chosen to include some representative plots created according to different testing sets and types of noise, in order to compare the classification performance of the audio-only and visual-only recognizers in contrast with three different audio-visual recognizers which use equal, optimal and estimated stream weights, denoted as *“baseline-AV”*, *“est-AV”* and *“opt-AV”* respectively. Classification accuracy of each recognizer, for various SNR levels, is considered to be the performance measure. The next set of figures illustrate this kind of comparison, as well as the estimation of audio stream weights, for five different noisy environments; babble speech noise, white noise, jet noise, interior car noise and factory noise respectively. Note that optimal weights have computed by hand. Also, in the title of each figure, it is attached the type of the mapping function which was used to estimate the stream weights. Note that all the results for recognition as well as classification, for all the types of noise, and all the testing sets, are included in the Appendix.

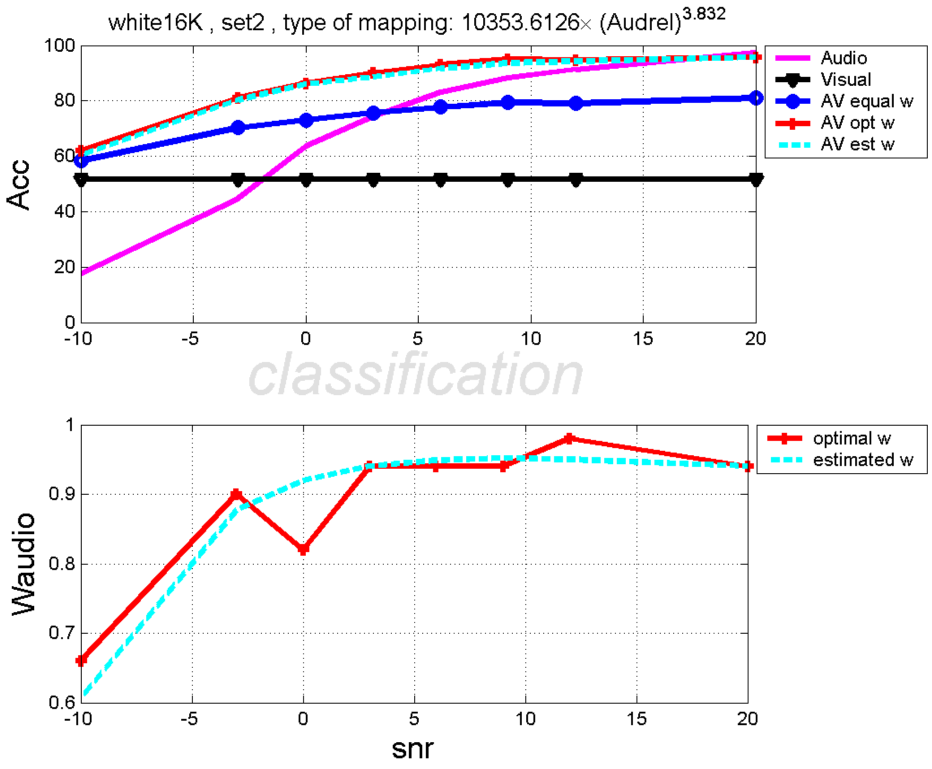
Our aim is to find a global mapping, with a satisfactory behaviour for all the types of noise, having preference to the $a \cdot \sqrt{AudReliability}$ [1] as it proved that this function gives on the average the best results according to Chapter 4. Thus, in order to measure the overall performance when using just one mapping type, we also include the results obtained using this function when an other function gave better results. For example, Fig.(5.4) demonstrates this kind of comparison. Note that in every situation, audio and visual stream confidence measures were mapped into weights using the mapping function which gave the best results. In most of the cases, when using $a \cdot \sqrt{AudReliability}$, weight estimation worked in good levels, comparing with the best mapping could be chosen, with respect to the noise or the testing set. This is not surprising due to the small difference in estimation error, for every situation, between the best mapping and that of $a \cdot \sqrt{AudReliability}$ as table (3.2) depicts. It is definitely preferable to search for a mechanism that will be able to adjust the mapping parameters for every situation, but it is more feasible to use just one function for every posible data set. The only issue we need to overcome, is the estimation of the parameter a. Preliminary only, non-conclusive experiments were carried out in order to achieve this.

In Fig.(5.4), it is obvious that for the most SNR levels, the audio-visual recognizer performs better than the recognizers which use single-modalities. That was our first goal when we started designing the audio-visual recognizer. Indeed, the same thing happens for the rest of the noisy environments as the following figures demonstrate. Another fact is that although the estimated weights are not close to the optimal (as fig. 5.4 depicts), especially for low levels of SNR, the overall distance from the optimal performance is considered to be reduced in contrast with the baseline implementation in which stream weights assumed to be equal. Finally, there is a difference in performance between recognition and classification tasks as it was expected. Actually, it depends on how we handle insertions during recognition as we show in the previous section. The next figures, represent the results for the rest of the noisy environments.

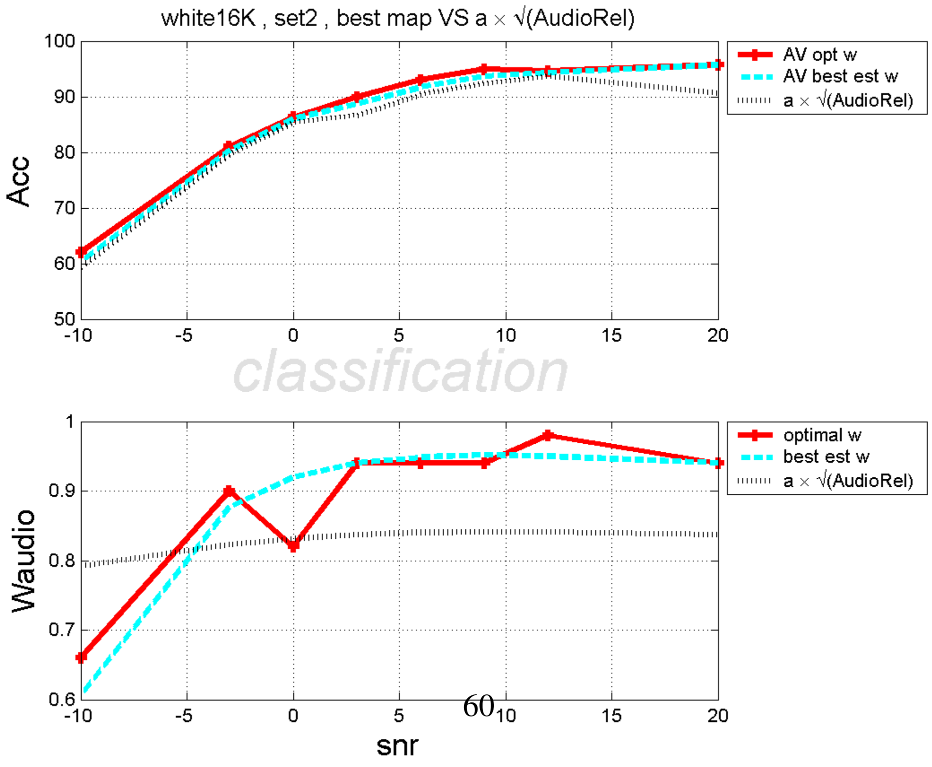
In reference with white noise, weight estimation for the classification task performed above the expectations as the two curves representing the optimal and the estimated audio weights came very close (see fig. 5.1.a). Consequently, the *"est-AV"* recognizer achievied almost identical results with the *"opt-AV"* implementation. Furthermore, the same recognizer outperformed the audio-only recognizer for about 16% on average, reaching an

[1]"AudRel" and "VisRel" in the included text of the figures, are for Audio Reliability and Visual Reliabiity respectively.

WHITE NOISE



(a)



(b)

Figure 5.1: White noise classification results for testing set 2

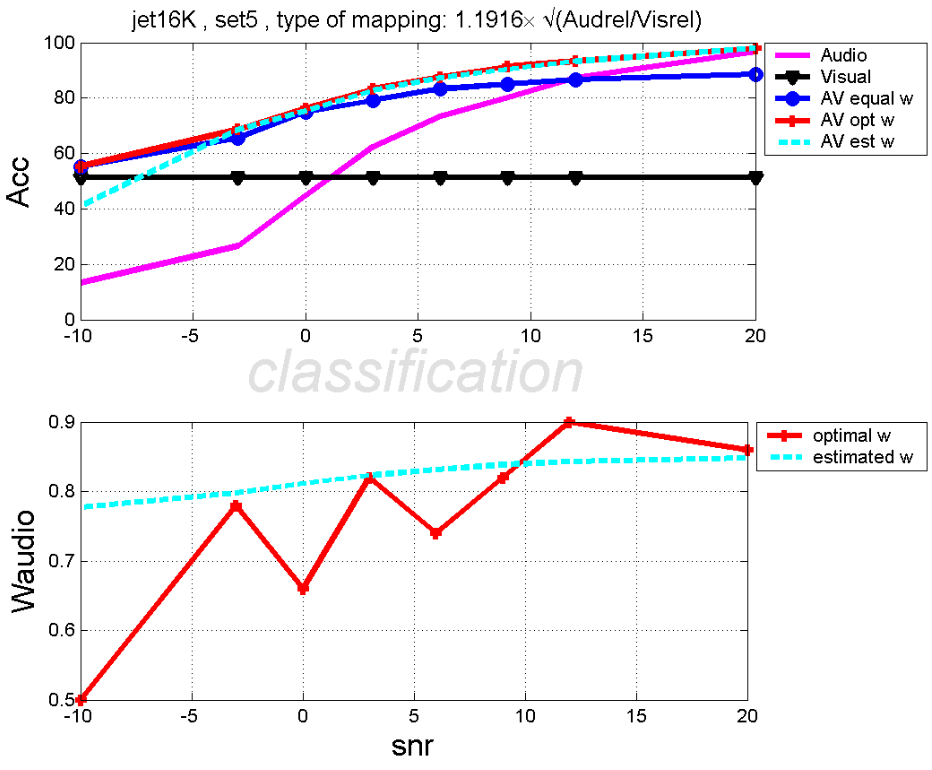
improvement of 66% locally, for low levels of SNR. If we use the square-root mapping function (see fig. 5.1.b), weight estimation is degraded; however, the impact on the overall performance seems to be insignificant.

Figure (5.2) represents the results for the case of jet cockpit noise. It is worth mentioning that although the optimal audio weights indicate a reduced performance for the the "baseline-AV" recognizer, unexpectedly, the results obtained using this recognizer are not far from the optimal. We have noticed that in many cases. However, using the estimated weights, the AV recognizer gets even more closer to the optimal performance. An other conclusion can be made, looking at fig.(5.2), is that there is not a big difference when we use $a \cdot \sqrt{AudReliability}$ in the place of $a \cdot \sqrt{AudReliability/VisReliability}$. It makes sense since the Reliability of the visual stream can be considered constant and unaffected by noise.

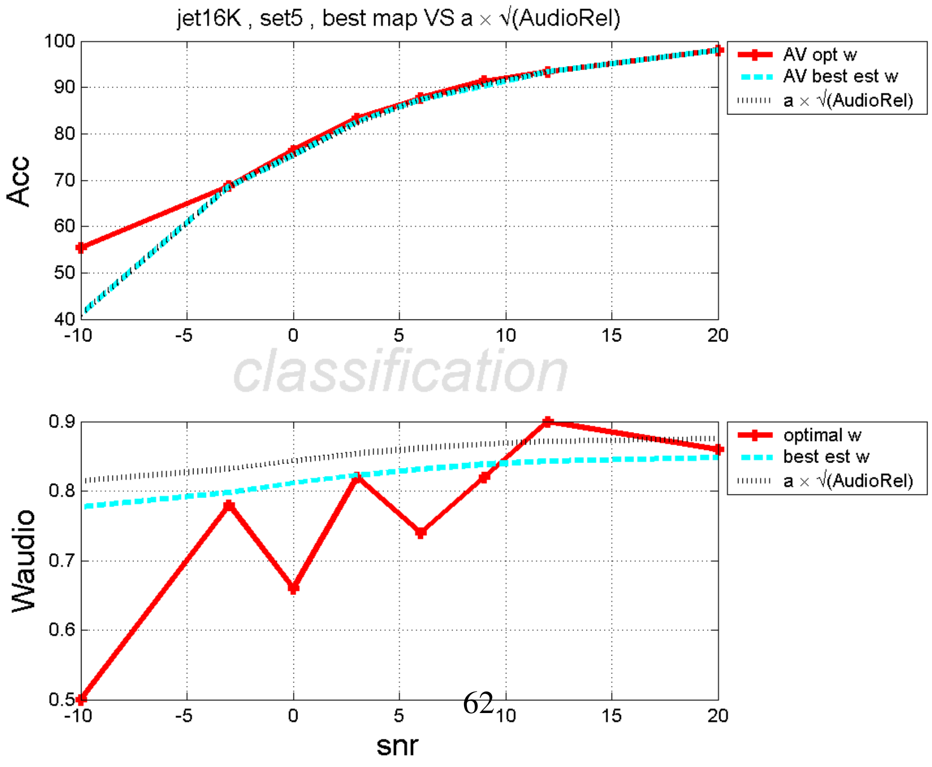
Interior car noise does not seem to degrade audio in the same manner as the rest types of noise do and hence, we expect to see different results. Indeed, Fig.(5.3.a) confirms that. Audio performance is very high and even the optimal AV recognizer cannot reach that levels, a fact that constitutes a reasonable doubt about how HTK manipulates stream weights. We purposely chose to include this figure as the only case in which the audio-only recognizer outperforms the AV recognizer for about 5 % on average. Actually, we noticed that HTK, for audio weights close to one and consequently, visual stream weights close to zero, does not use only the audio stream as it was expected to do.

Finally, figures (5.5) and (5.6) represent the average audio stream estimated weights for classification and recognition respectively, computed for each kind of noise and for all levels of SNR, using data of every testing set (the round-robin as depicted in table 4.2). A first observation about these figures is that recognition curves have almost the same gradients with the classification ones. Thus, they change similarly over SNR, a fact that may possibly indicates a good function of weight estimation, as it seems able to adapt to different noises. Knowing that recognition accuracy is biased, due to the insertion penalty selection, we expect a degraded performance in contrast with classification, which is clearly confirmed by observing the figures of the recognition results. All the recognition accuracy curves seem to be biased in the same way and transfered in lower levels, compared with the classification ones. However, we assume that the visual stream is more affected than the audio, observing that the audio weights increase in recognition.

JET COCKPIT NOISE



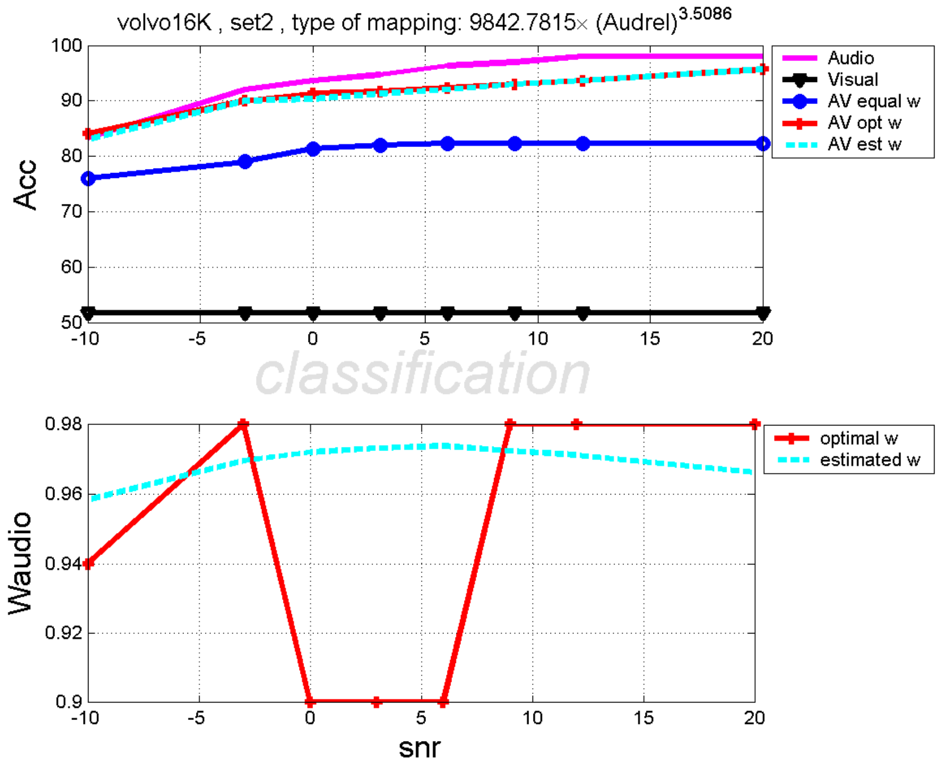
(a)



(b)

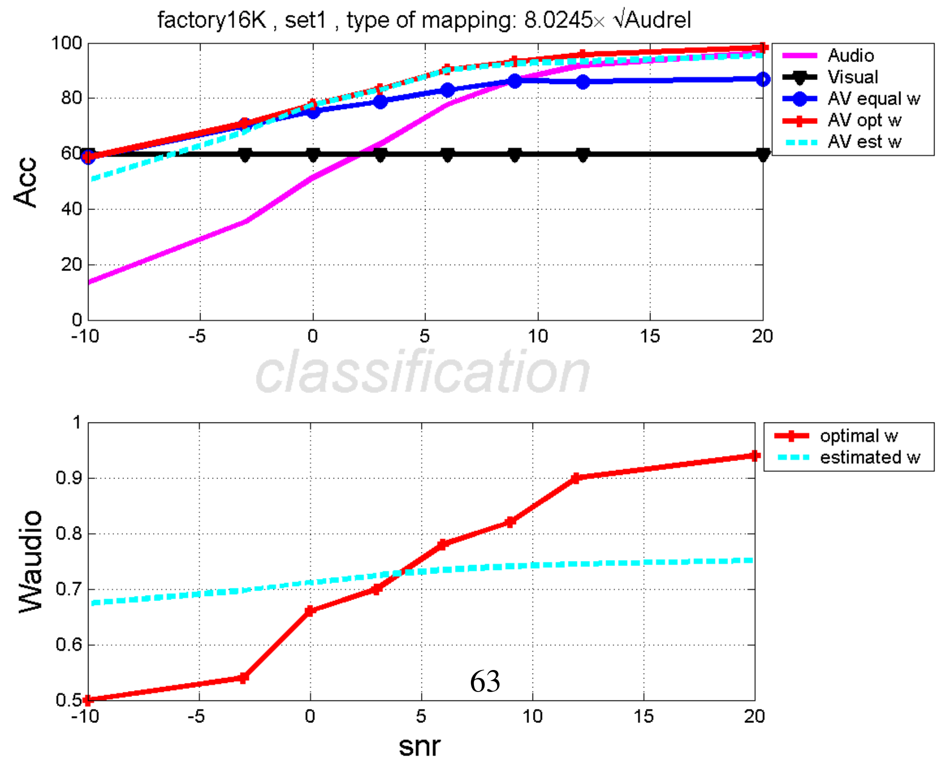
Figure 5.2: Jet noise classification results for testing set 5

CAR INTERIOR NOISE



(a)

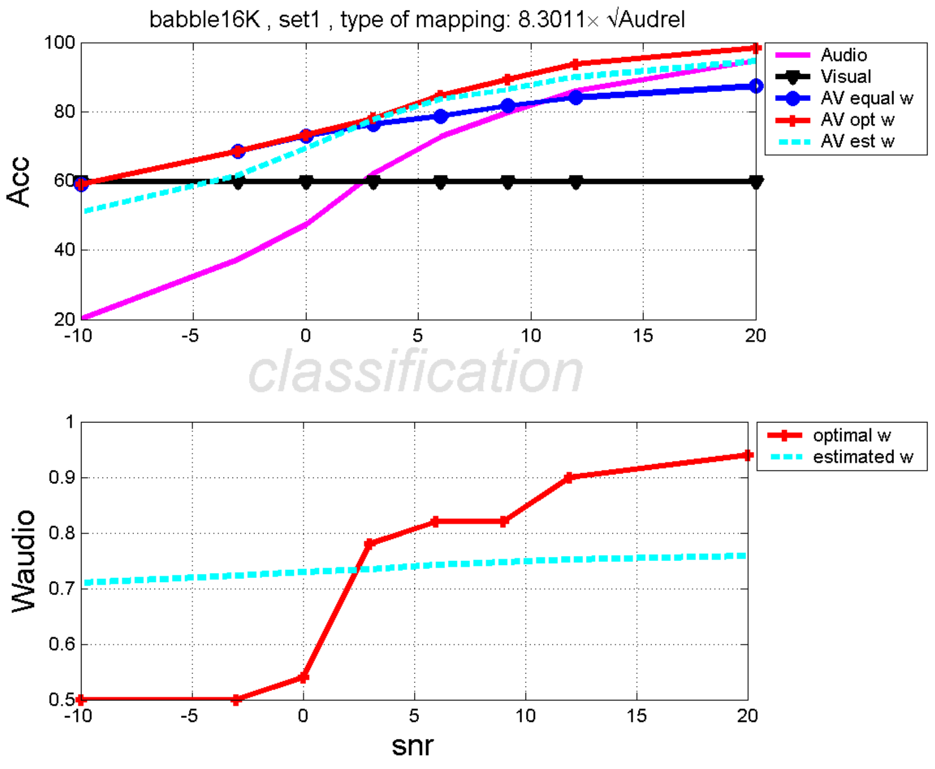
FACTORY NOISE



(b)

Figure 5.3: (a) Car interior noise classification results for testing set 5. (b) Factory noise classification results for testing set 1

BABBLE SPEECH NOISE



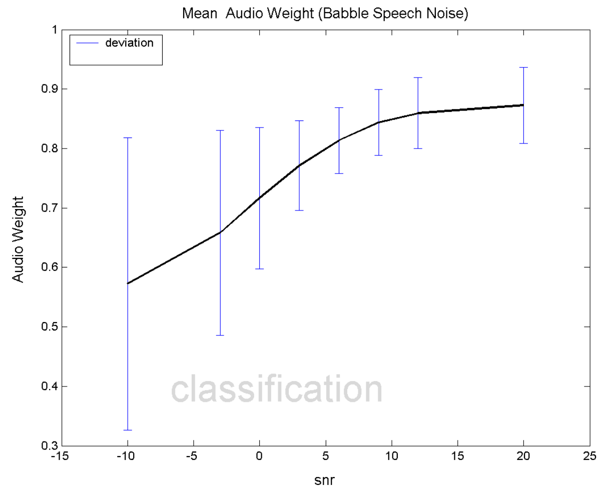
(a)

Figure 5.4: Babble speech noise classification results for testing set 1

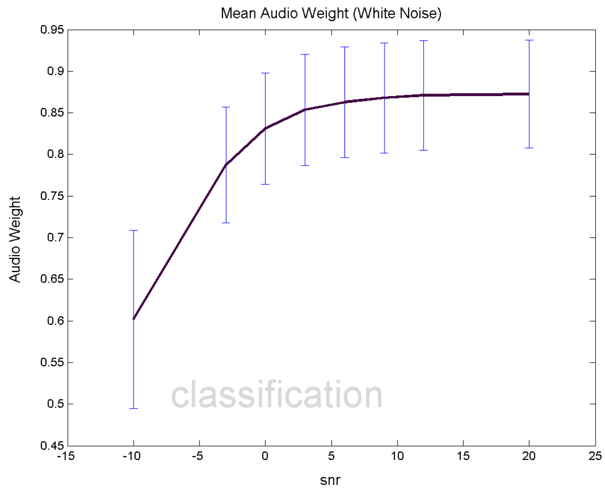
An other fact is that the weights vary reasonably with respect to the SNR changes, being flexible and adaptable as well as representative of the environmental conditions. For instance, audio weights for the car interior noisy environment (fig. 5.5.d), have been estimated over the 0.9 level for all the SNR values, indicating a high-confidence audio stream, which is realistic due to the increased classification accuracies that audio achieved.

Figure (5.7) reports a summary of the mean WER reduction has been achieved over all SNR levels, by using *"est-AV"* instead of *"audio-only"* recognizer, for each type of noise. The same figure reports the gains have been obtained by using *"est-AV"* instead of *"baseline-AV"* and also, the mean performance distance between *"est-AV"* and *"opt-AV"*. We observed a significant improvement in performance when using AV recognizers, compared with the degraded audio performance, specially in low levels of SNR. For the Jet noise case, it is worth mentioning that the *"est-AV"* recognizer exhibited an improvement of 83% over the *"audio-only*" recognizer, for classification, and respectively, a significant improvement of 476% for recognition, where audio performance seems to be diminished. Moreover, in the case of car interior noise, the improvement over the *"audio-only"* recognizer seems to be of no importance but in fact, there was not room for improvement, as the audio achieved high levels of performance. To conclude, *"est-AV"* outperformed both *"audio-only"* and *"visual-only"* recognizers as well as the *"baseline-AV"* in the most of the cases.

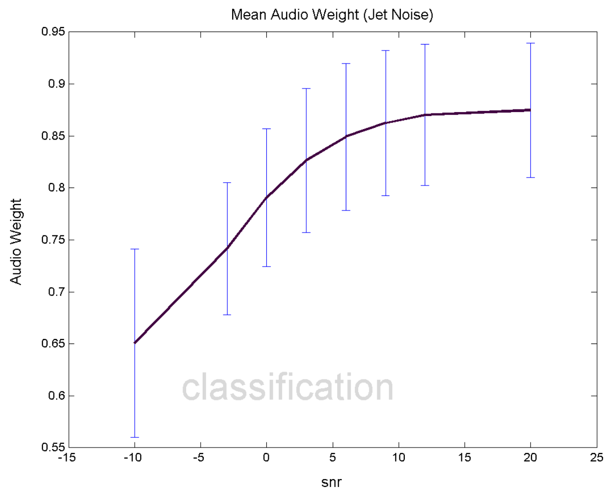
MEAN ESTIMATED AUDIO WEIGHTS FOR CLASSIFICATION



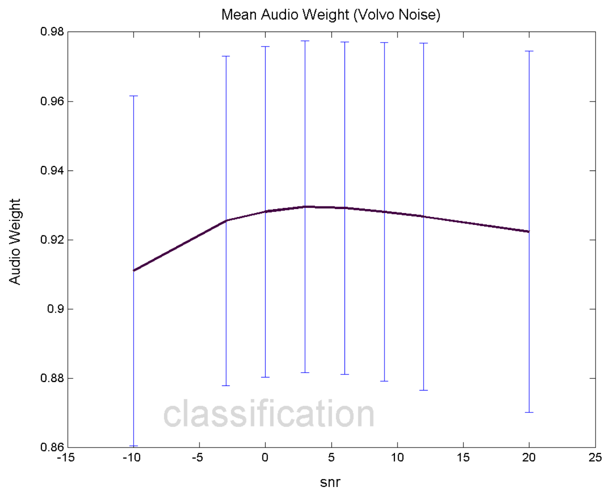
(a)



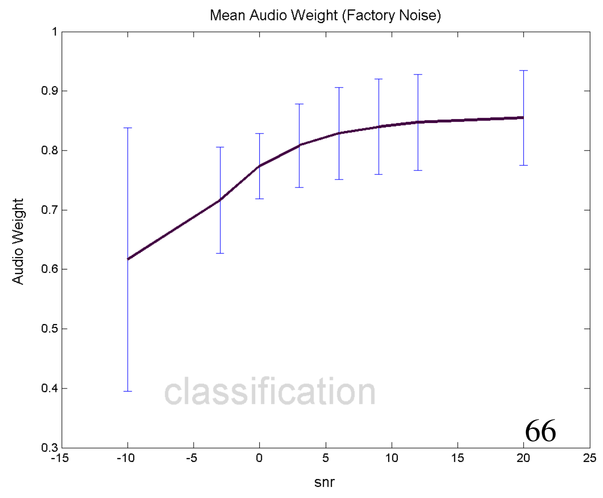
(b)



(c)



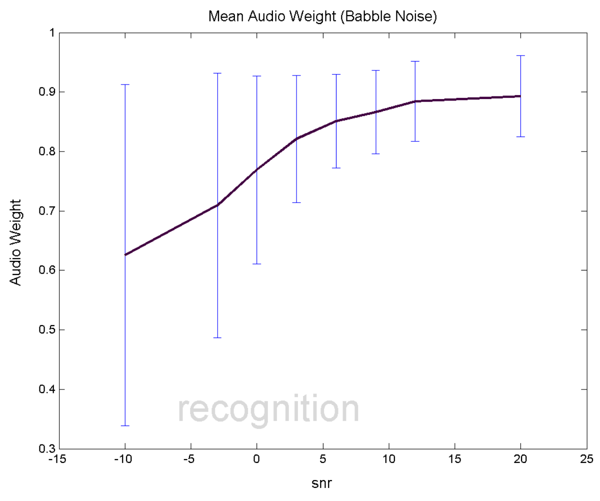
(d)



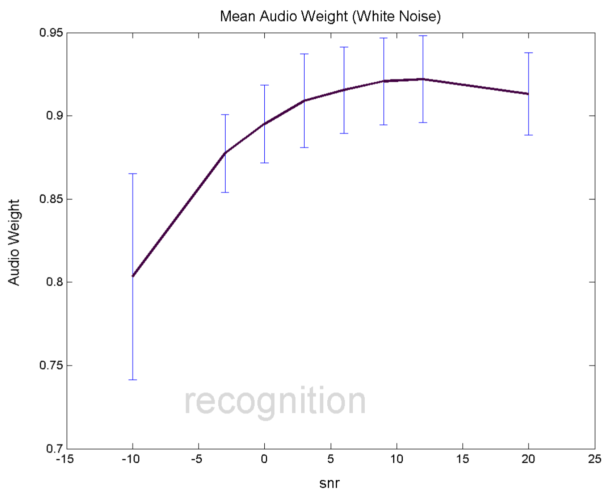
(e)

Figure 5.5: Mean audio stream estimated weights for classification (a) Babble speech , (b) White , (c) Jet cockpit noise, (d) Car interior, (e) Factory

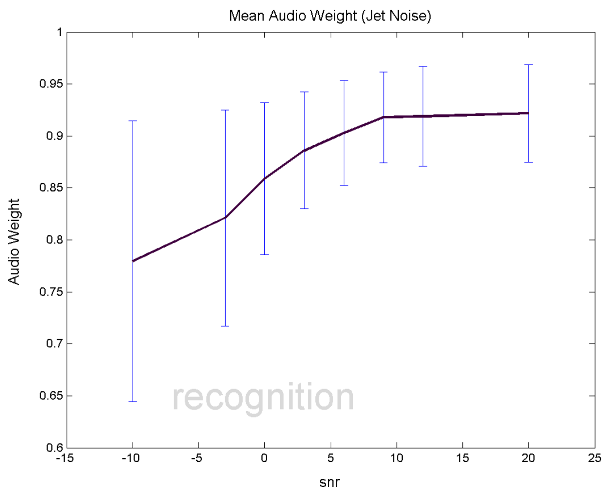
MEAN ESTIMATED AUDIO WEIGHTS FOR RECOGNITION



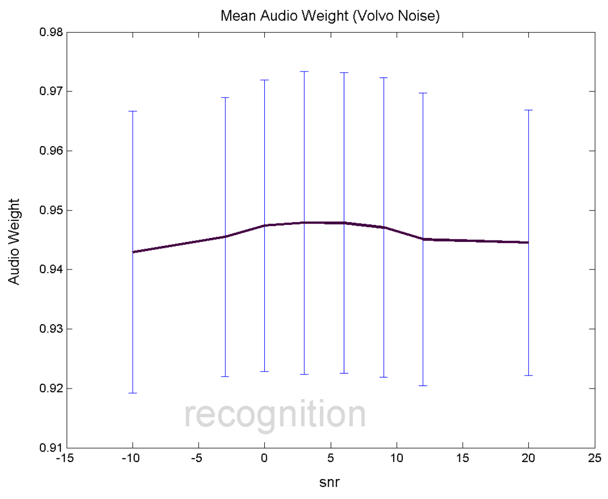
(a)



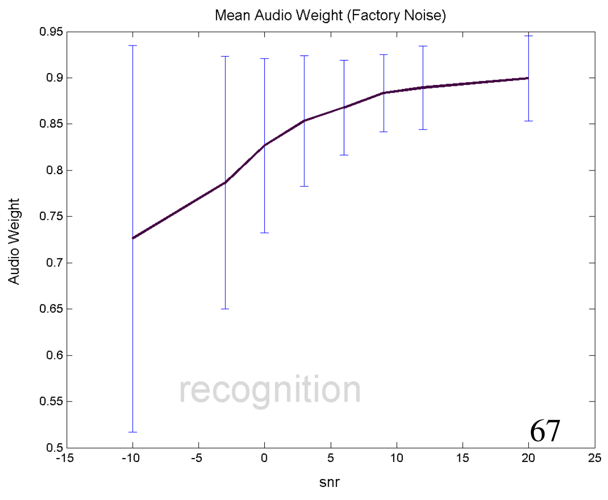
(b)



(c)

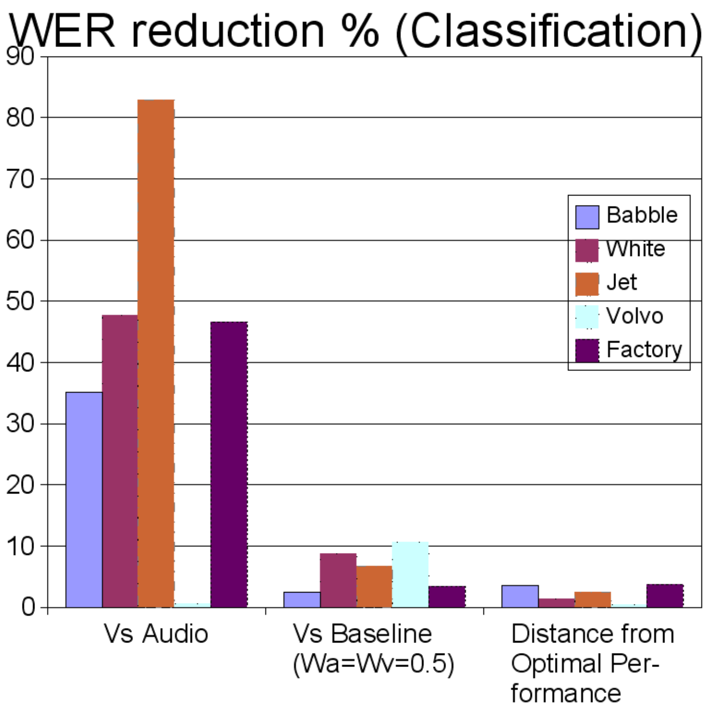


(d)

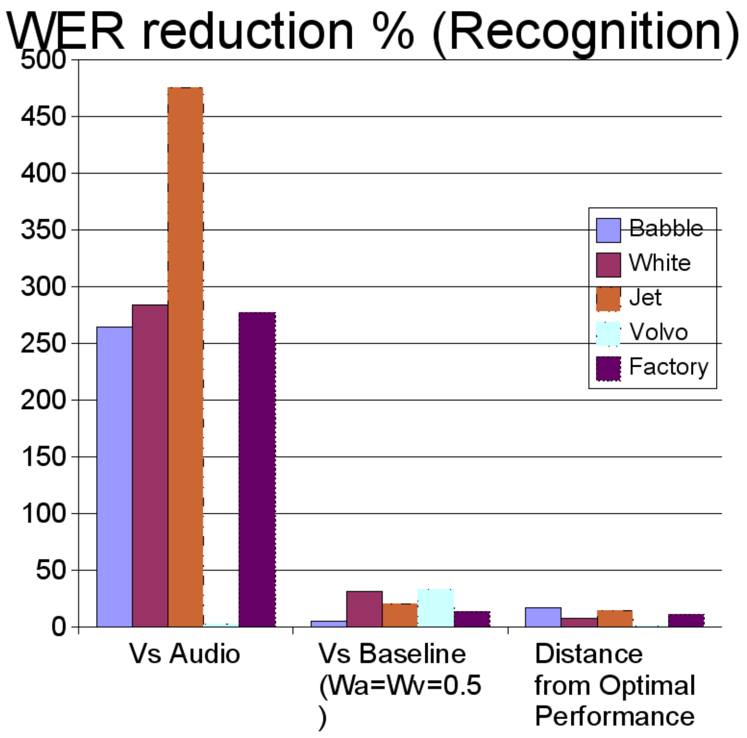


(e)

Figure 5.6: Mean audio stream estimated weights for recognition (a) Babble speech , (b) White , (c) Jet cockpit noise, (d) Car interior, (e) Factory



(a)



(b)

Figure 5.7: (a) Classification mean WER reduction achieved by “est-AV”. (b) Recognition mean WER reduction achieved by “est-AV”.

Chapter 6

Discussion and Future Work

When this project started as an idea, our first goal was to optimize a baseline visual front-end in order to extract visually relevant features. At first, our main focus was a more sophisticated ROI tracking, which was achieved by applying template matching in every frame to detect the nose tip, and use it as reference for the ROI determination. This yielded a significant WER reduction over the baseline front-end which was measured about 70%. Further improvement of 25% was obtained when we corrected the head rotation, as well as the scaling differences which occurred due to slight back-and-forth speaker movements. To perform these experiments, a visual HMM-based recognizer was built.

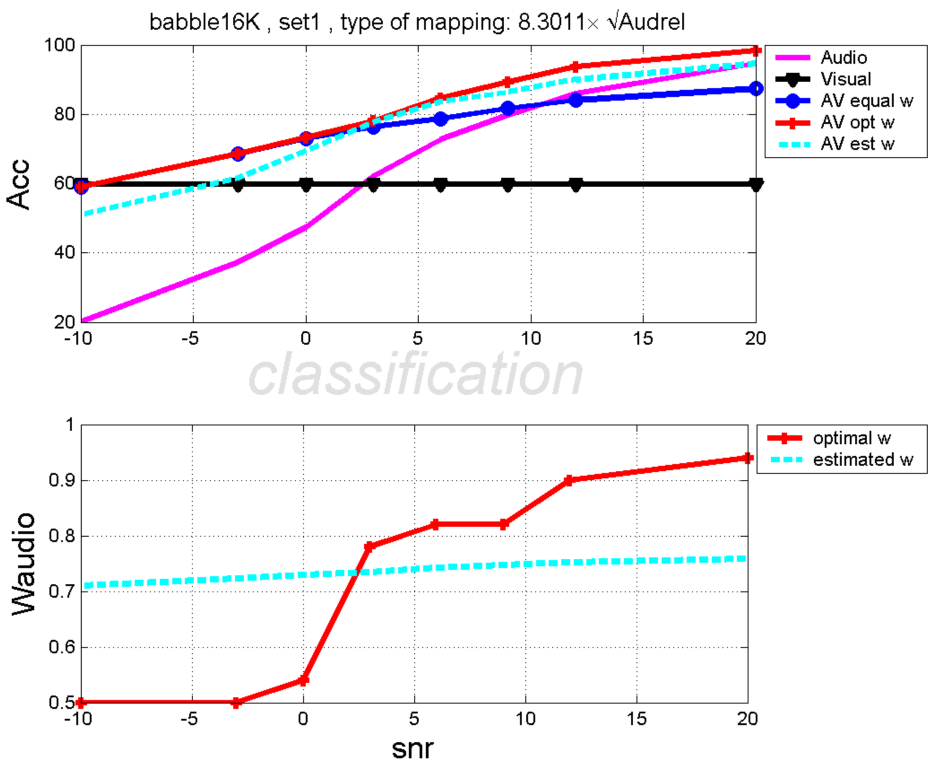
Once visual features were improved, we investigated the unsupervised stream weight estimation for a state-synchronous audio-visual recognizer, in order to emphasize or deemphasize the contribution of each stream in the final decision. A k-means based method was applied to estimate the confidence of each stream and subsequently to compute the appropriate stream weights. To compare the audio-visual implementation with an audio-only recognizer, a multi-stream HMM recognizer was trained using "clean" data and then tested in various noisy environments which was simulated by degrading the audio with different types of noise. Definitely, the audio-visual recognizer outperformed significally both the audio-only and the visual-only recognizers. Also, using the estimated weights, yielded an improvement over the baseline implementation which assumed equal stream weights.

Generally, when observing the figures, it is obvious that the obtained results seem to be very promising. We believe that there is space for further improvements in order to make the visual front-end as well as the stream weight estimation more robust under a variety of visually variable conditions. Given the nature of the CUAVE database, we only used template matching to track the face. Actually, a possible degraded performance of our ROI tracking implementation for moving speakers, is definitely reasonable due to rapid changes. A more sophisticated approach need to be considered in order to overcome this drawback. Moreover, profile view lipreading was out of the project's scope but it is even more challenging. A useful source of information that influences the decision is the experimentally observed asynchrony of the two streams. Being the easiest from the implementation point of view, synchronous feature level fusion was the baseline in our experiments. Although this framework does not allow for modeling this kind of asynchrony, it does allow for weighting the decisions independently. It is important to come up with techniques that exploit our knowledge for the delay levels between the streams. We did not focus in this aspect of the problem during this project.

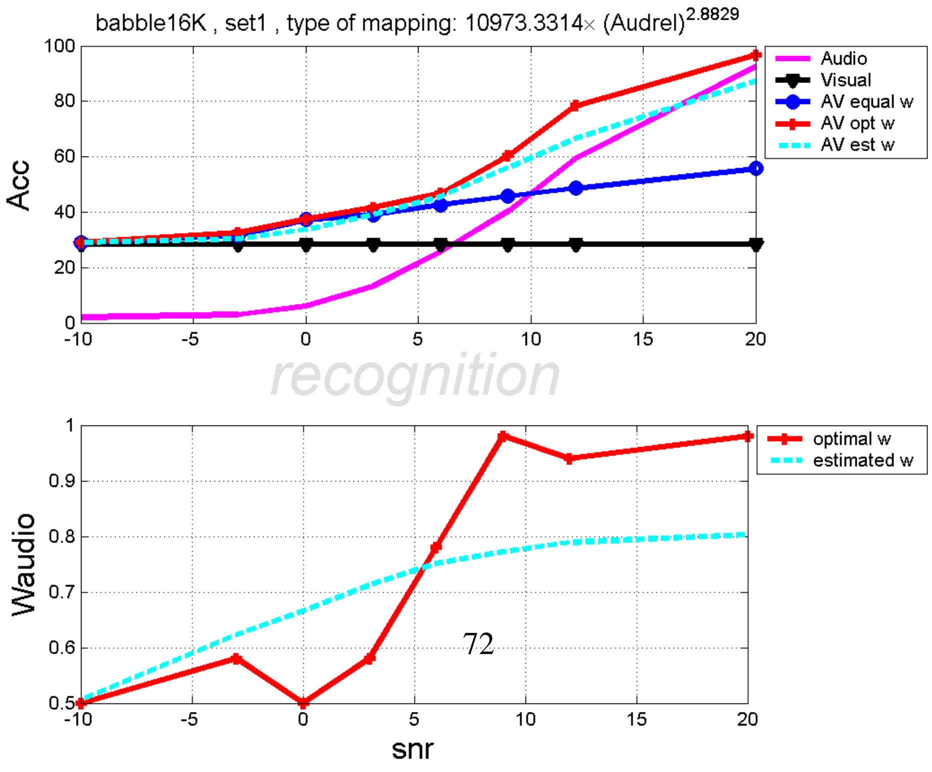
APPENDIX

This appendix contains all the available results obtained from our experiments in stream weight estimation. Our aim is to compare the classification performance of the audio-only and visual-only recognizers in contrast with three different audio-visual recognizers which use equal, optimal and estimated stream weights, denoted as *"baseline-AV"*, *"est-AV"* and *"opt-AV"* respectively. The next set of figures illustrate this kind of comparison, as well as the estimation of audio stream weights, for five different noisy environments; babble speech noise, white noise, jet noise, interior car noise and factory noise respectively. Also, in the title of each figure, it is attached the type of the mapping function which was used to estimate the stream weights. Our aim is to find a global mapping, with a satisfactory behaviour for all the types of noise, having preference to the $a \cdot \sqrt{AudReliability}$ as it proved that this function gives on the average the best results according to Chapter 4. Thus, in order to measure the overall performance when using just one mapping type, we also include the results obtained using this function when an other function gave better results. All the experiments performed for classification as well as recognition, for six different test sets. Finally, a table represents the overall WER reduction achieved for classification and recognition respectively.

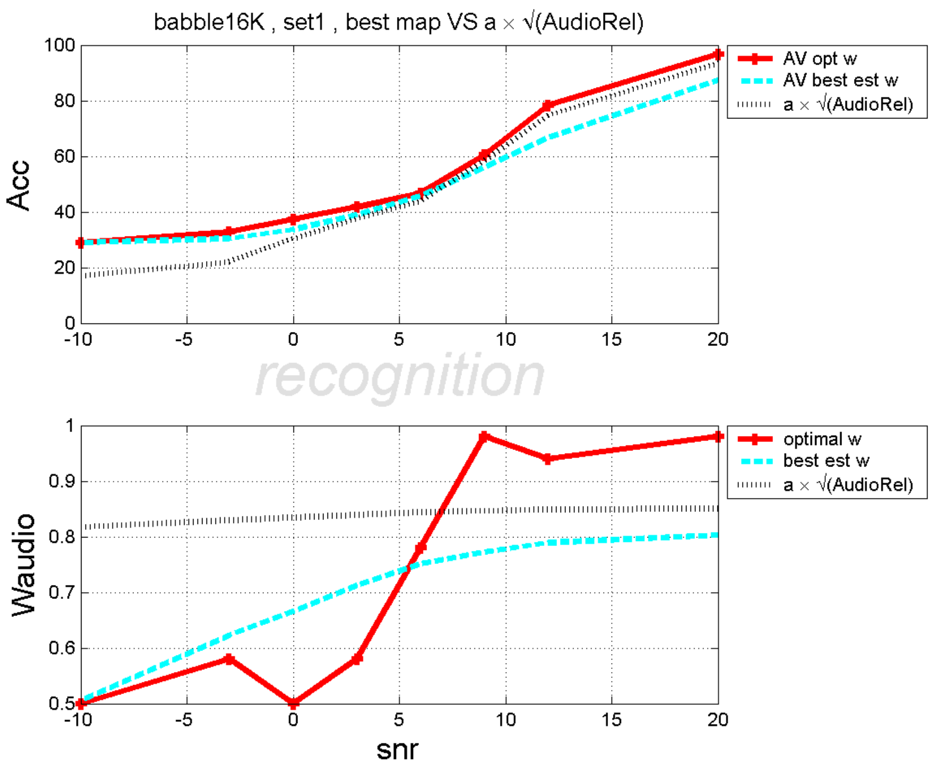
Babble Speech Noise



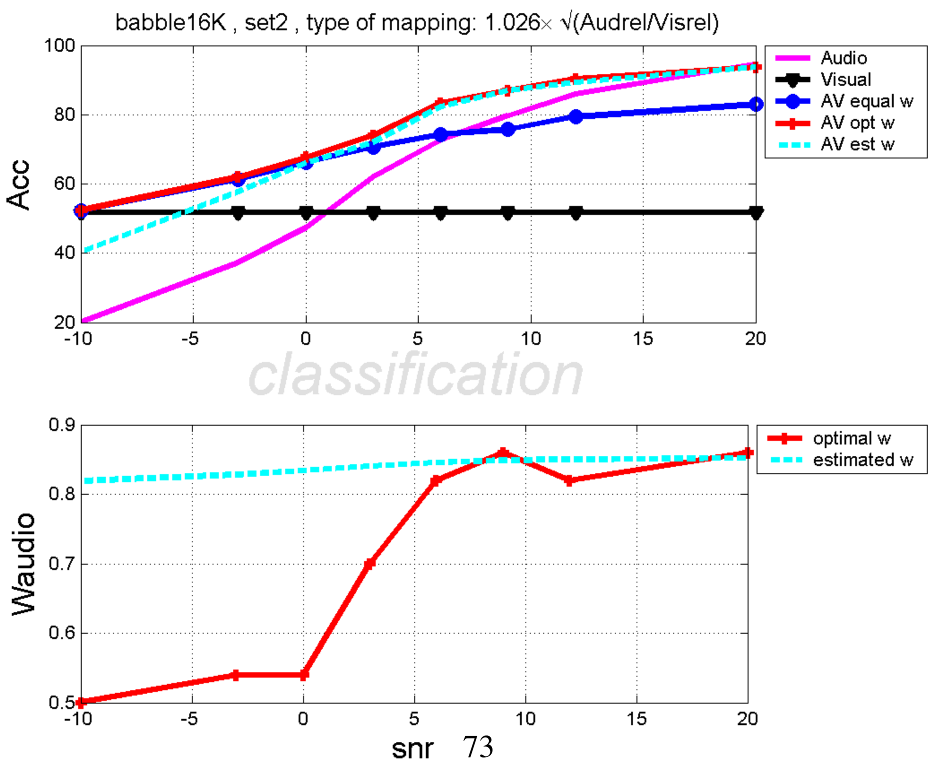
(a)



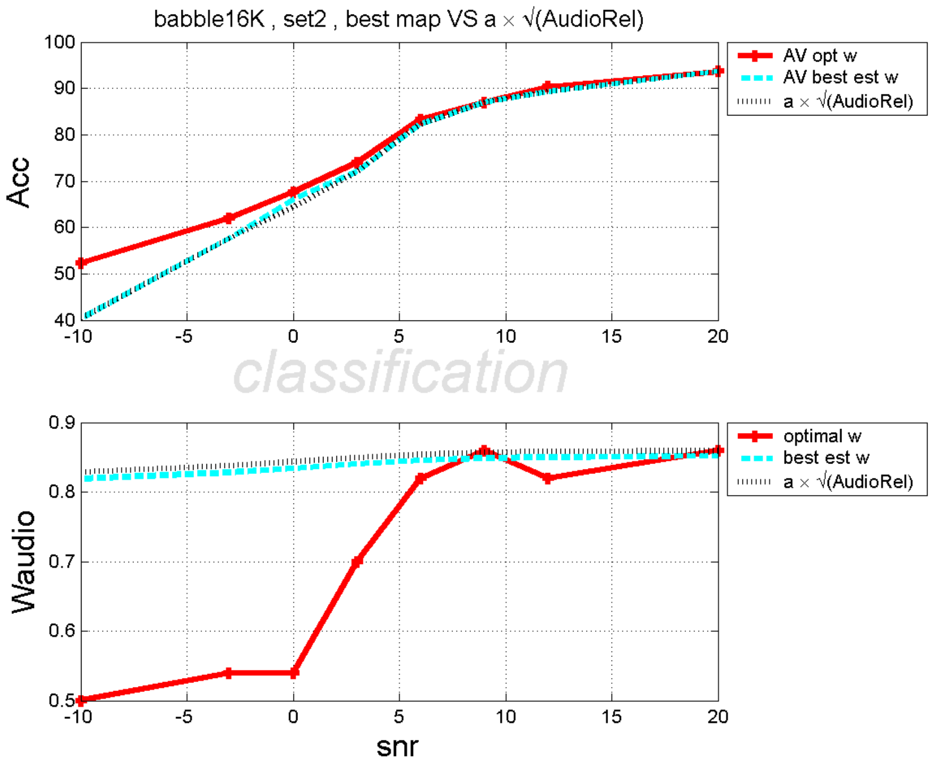
(b)


babble16K , set1 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
recognition
Waudio
optimal w
best est w
a × √(AudioRel)
snr

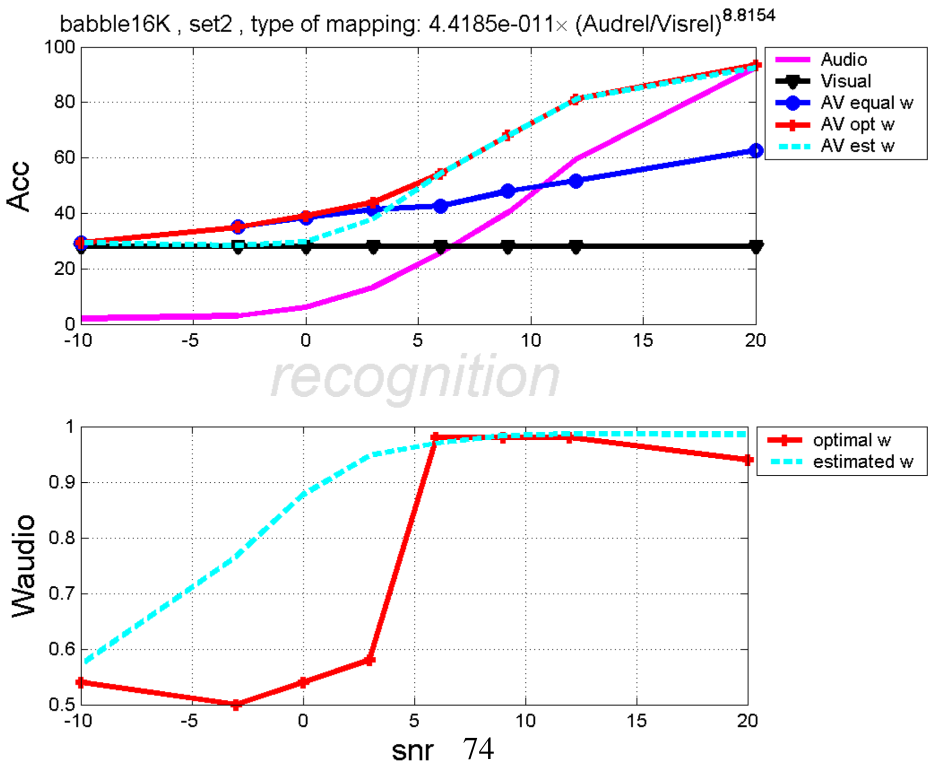
(c)


babble16K , set2 , type of mapping: 1.026× √(Audrel/Visrel)
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
classification
Waudio
optimal w
estimated w
snr

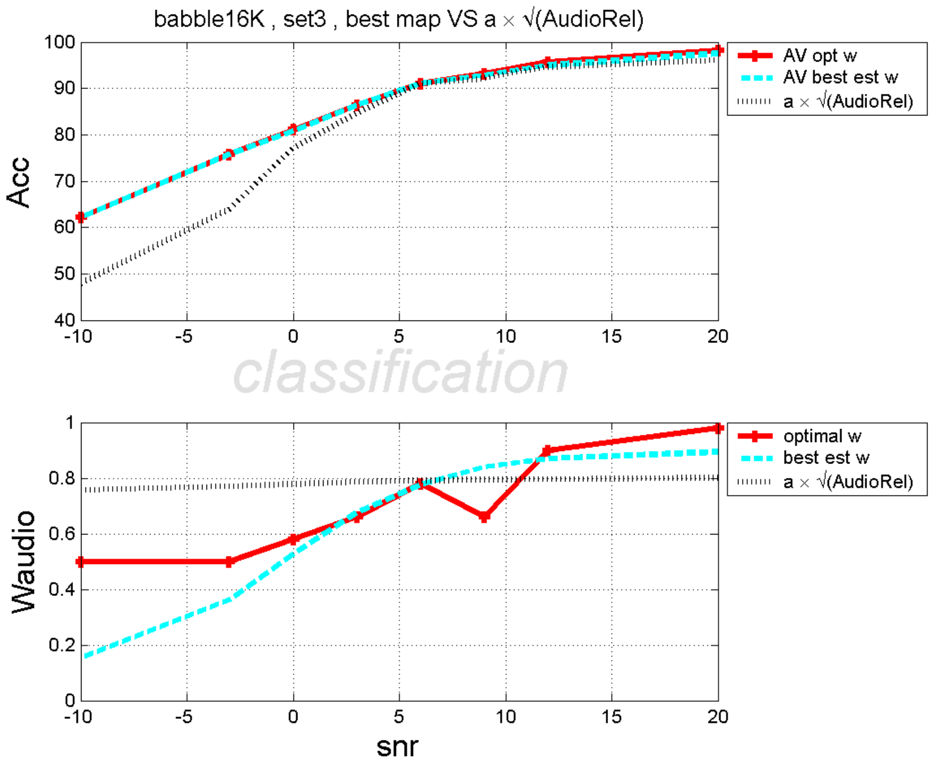
(d)


babble16K , set2 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
classification
Waudio
optimal w
best est w
a × √(AudioRel)
snr

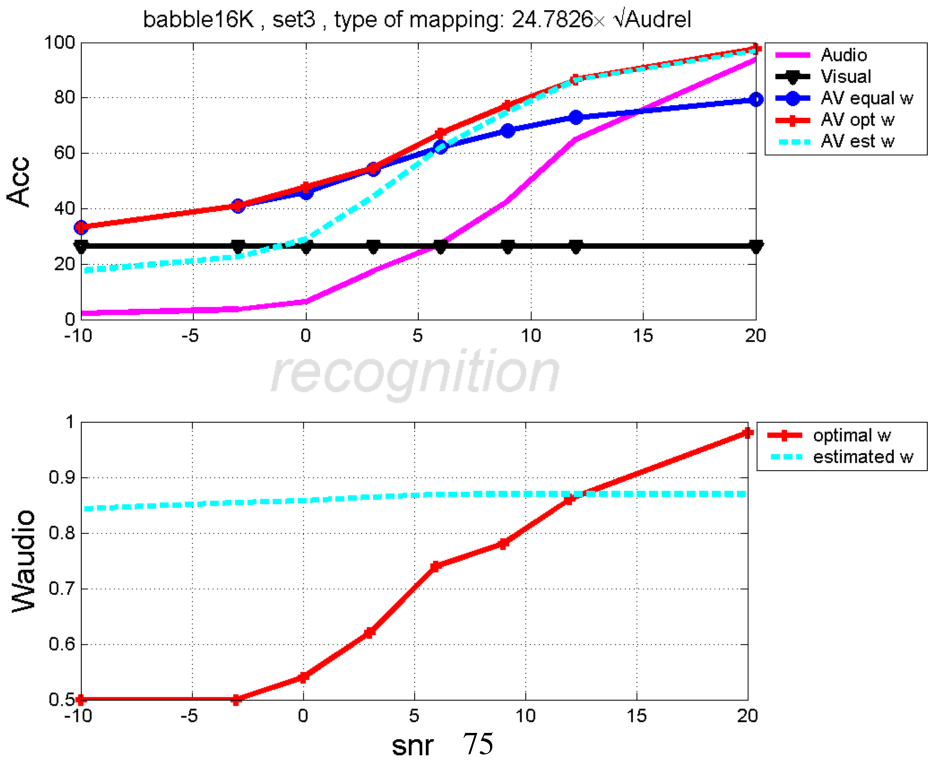
(e)


babble16K , set2 , type of mapping: 4.4185e-011× (Audrel/Visrel)^8.8154
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
recognition
Waudio
optimal w
estimated w
snr

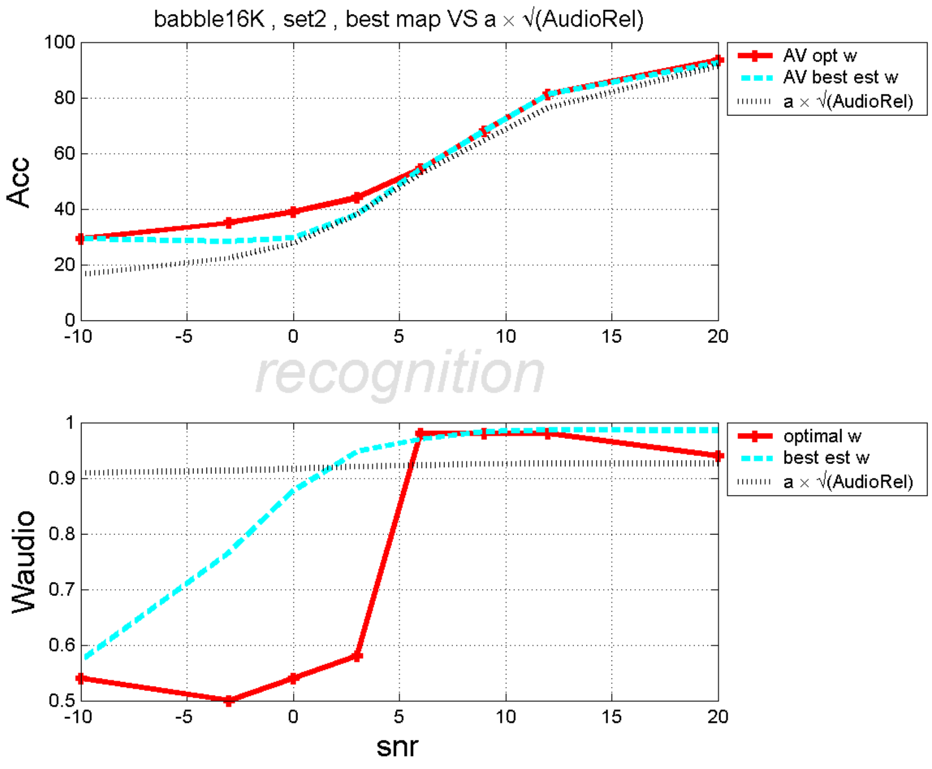
(f)


babble16K , set3 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
classification
Waudio
optimal w
best est w
a × √(AudioRel)
snr

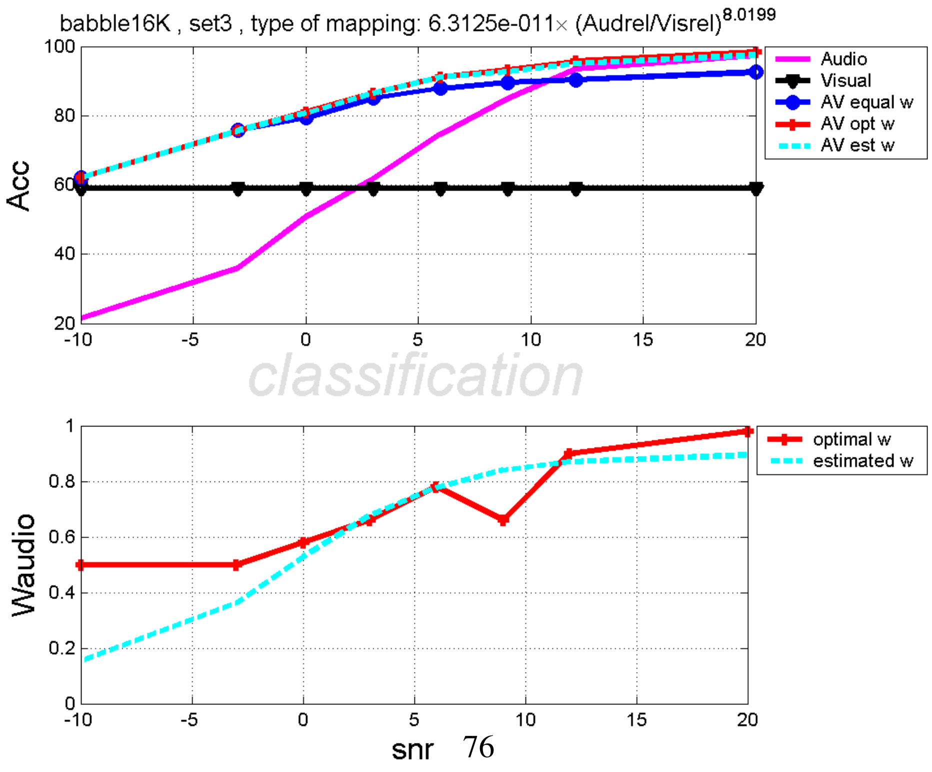
(i)


babble16K , set3 , type of mapping: 24.7826× √Audrel
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
recognition
Waudio
optimal w
estimated w
snr

(j)

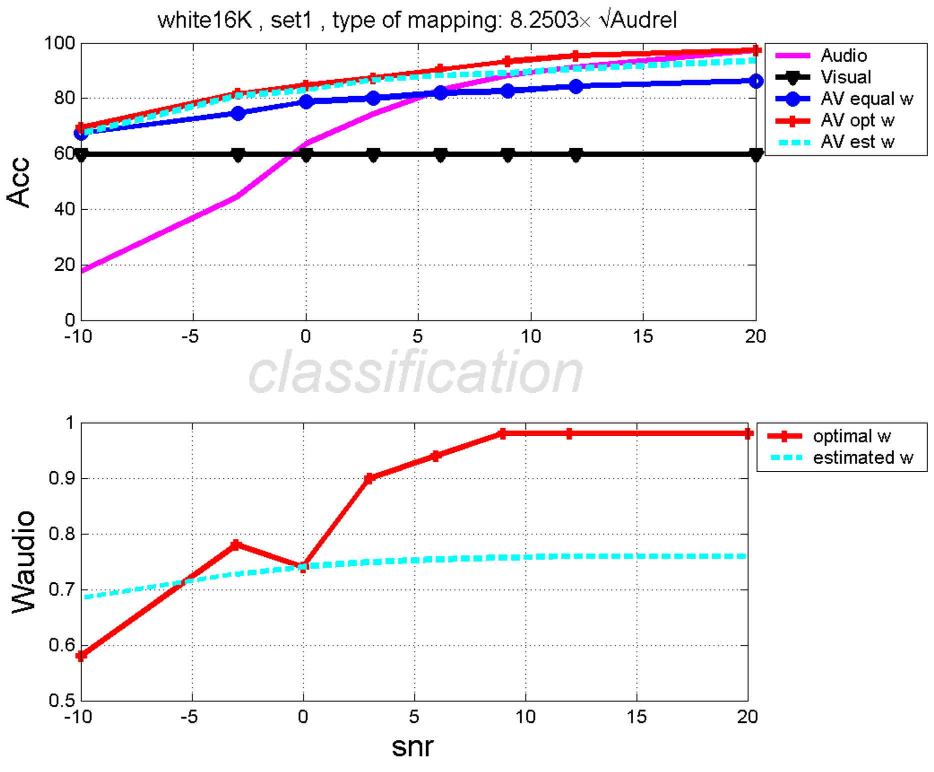

babble16K , set2 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
recognition
Waudio
optimal w
best est w
a × √(AudioRel)
snr

(g)

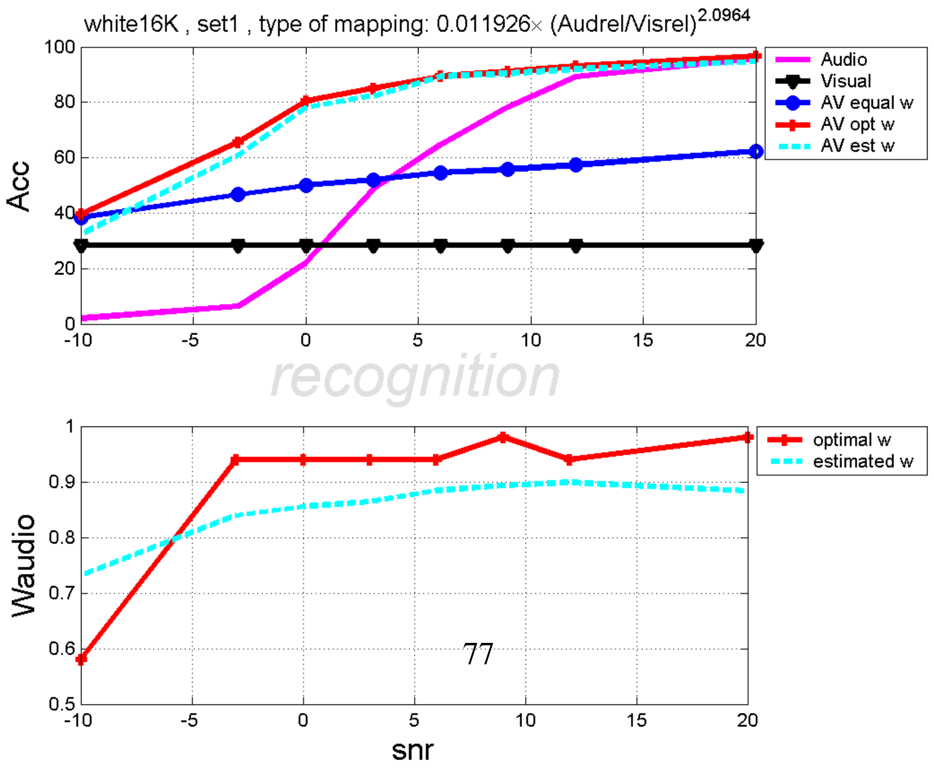

babble16K , set3 , type of mapping: 6.3125e-011× (Audrel/Visrel)^8.0199
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
classification
Waudio
optimal w
estimated w
snr

(h)

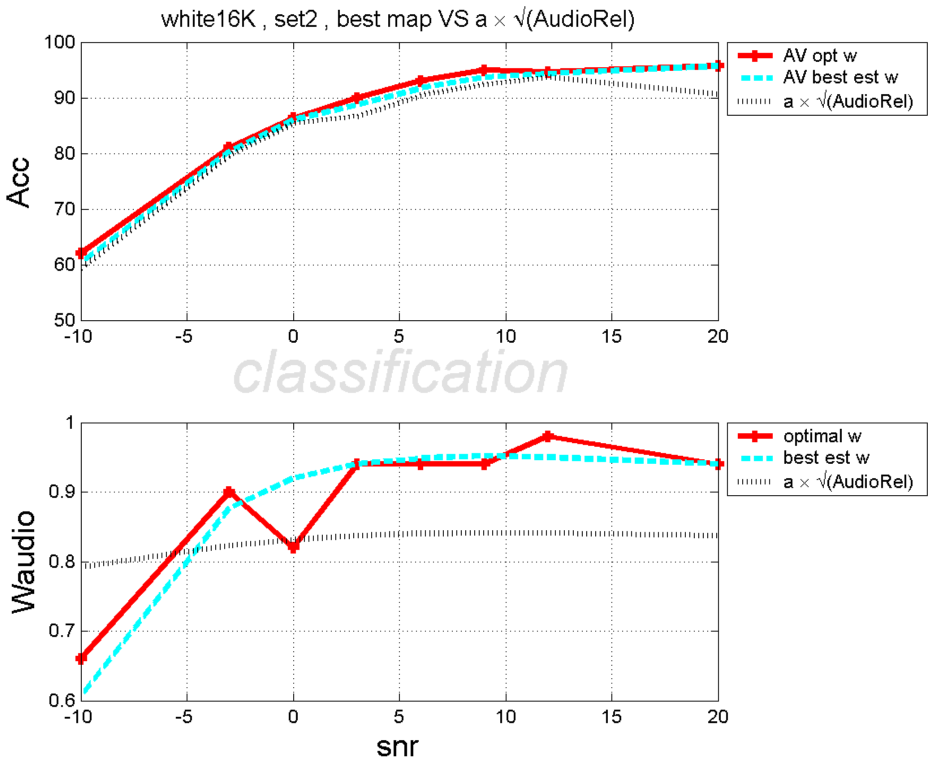
White Noise



(k)

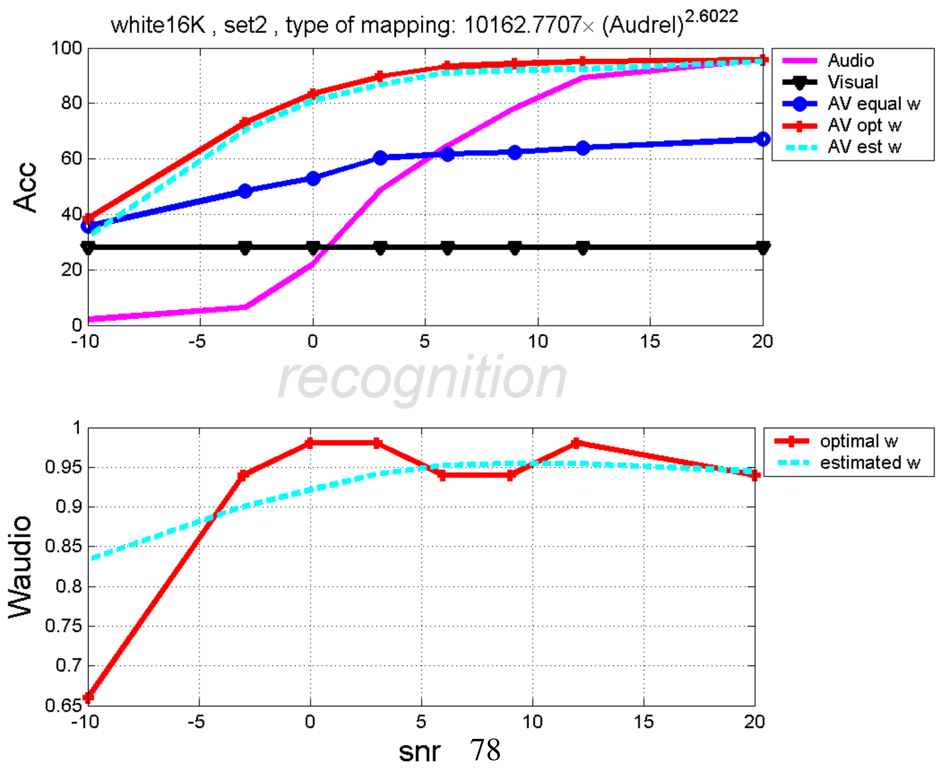


(l)



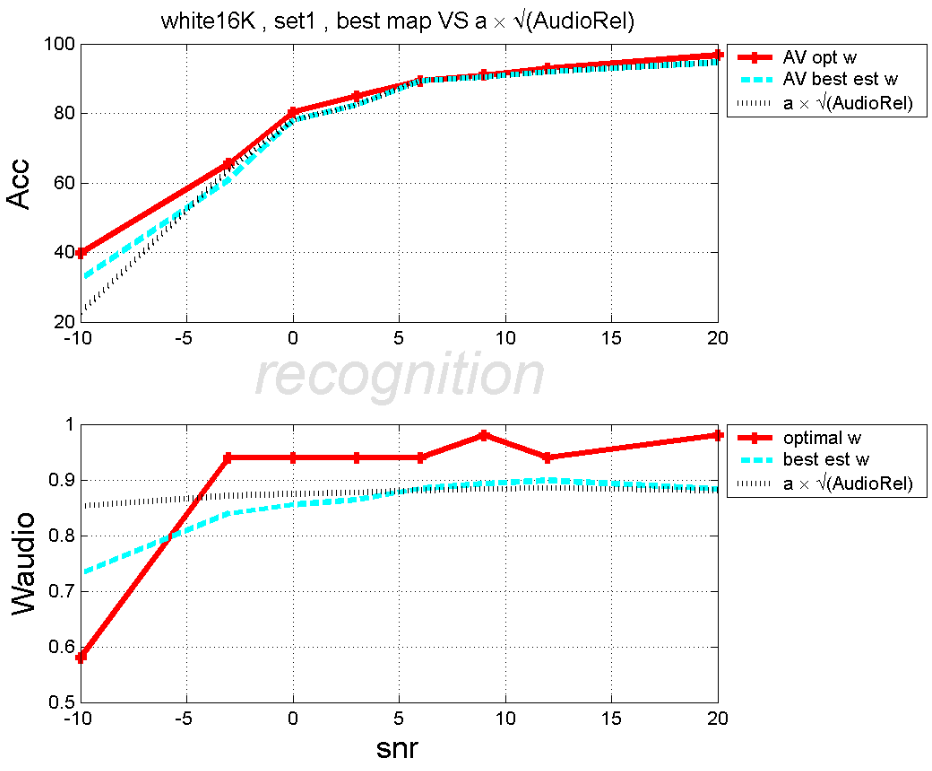
white16K , set2 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
classification
Waudio
optimal w
best est w
a × √(AudioRel)
snr

(o)

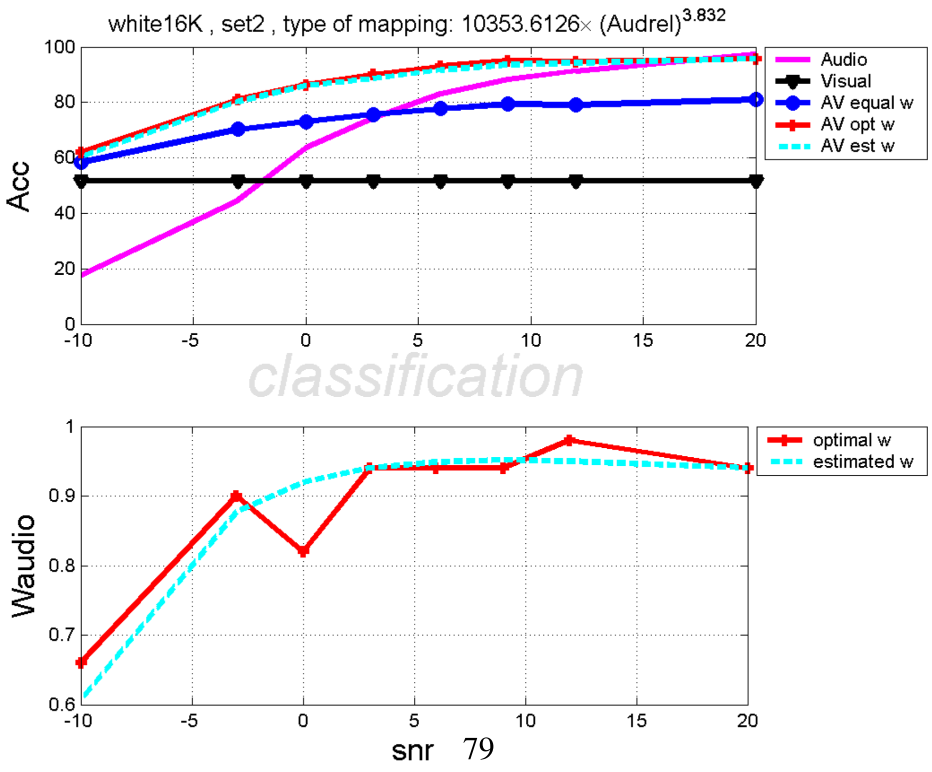


white16K , set2 , type of mapping: 10162.7707× (Audrel)^2.6022
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
recognition
Waudio
optimal w
estimated w
snr

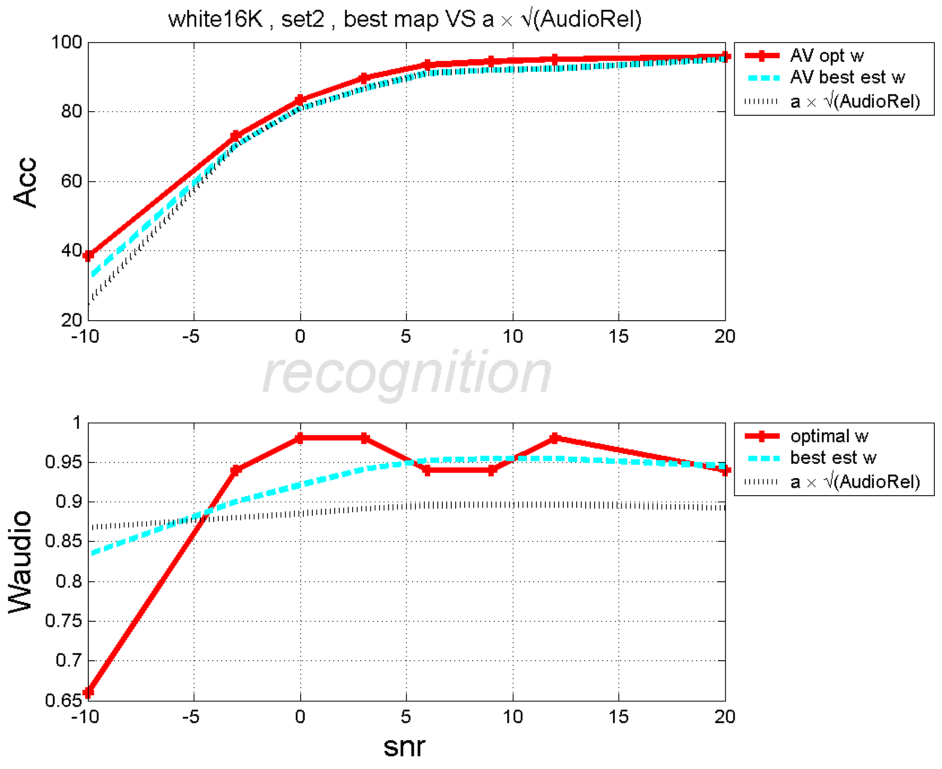
(p)


white16K , set1 , best map VS a × √(AudioRel)
Acc
Waudio
snr
recognition
AV opt w
AV best est w
a × √(AudioRel)
optimal w
best est w
a × √(AudioRel)

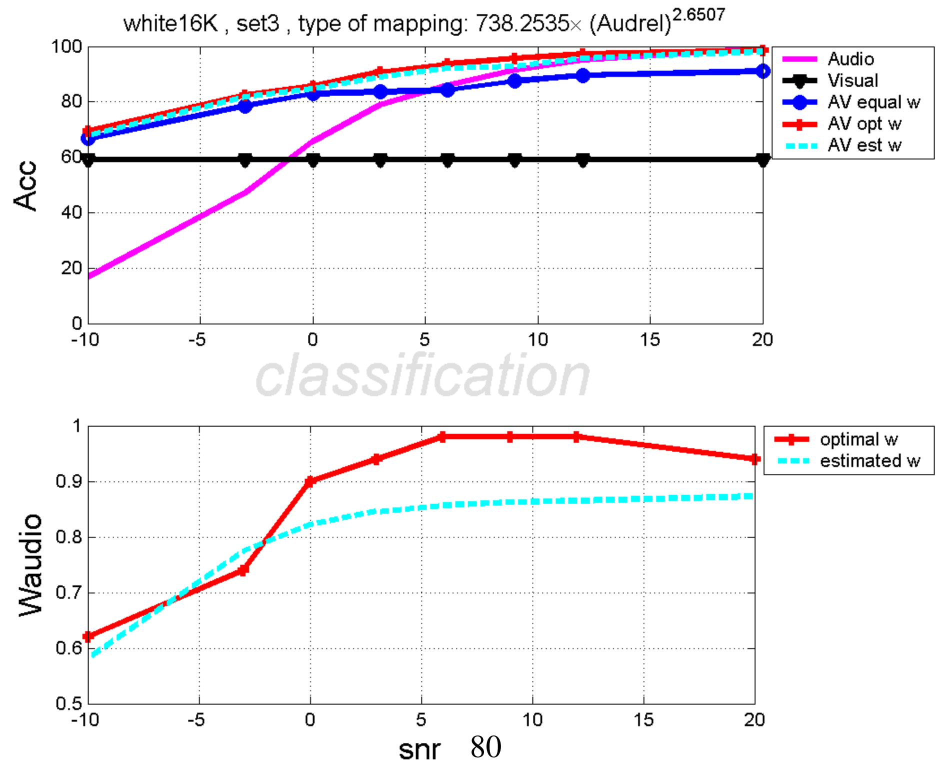
(m)


white16K , set2 , type of mapping: 10353.6126× (Audrel)^3.832
Acc
Waudio
snr
classification
Audio
Visual
AV equal w
AV opt w
AV est w
optimal w
estimated w

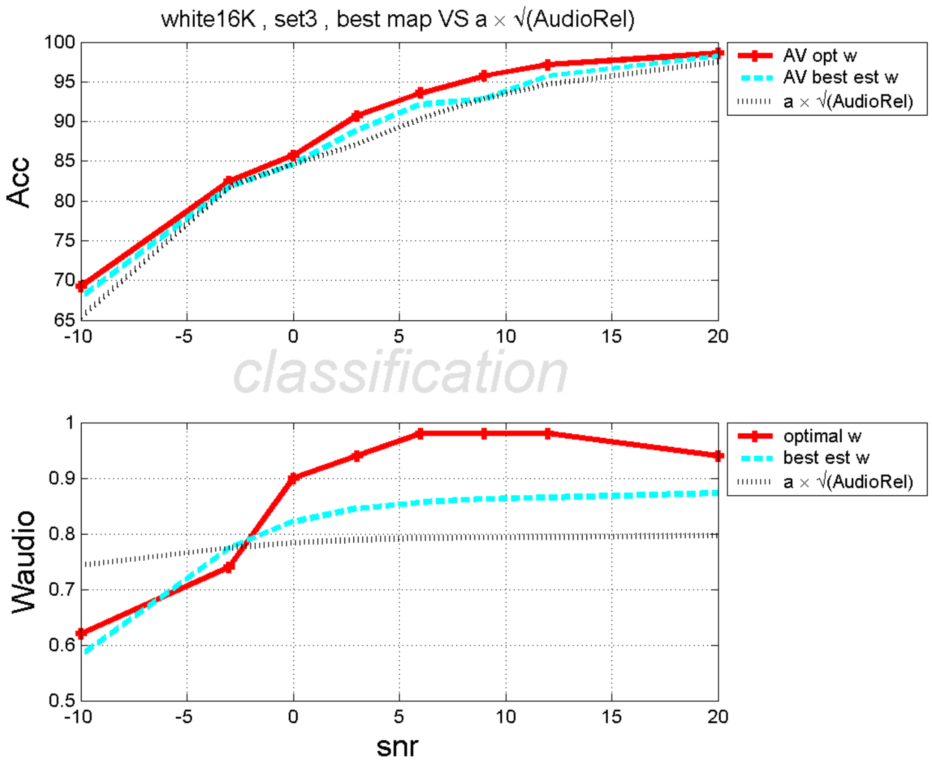
(n)


white16K , set2 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
recognition
Waudio
optimal w
best est w
a × √(AudioRel)
snr

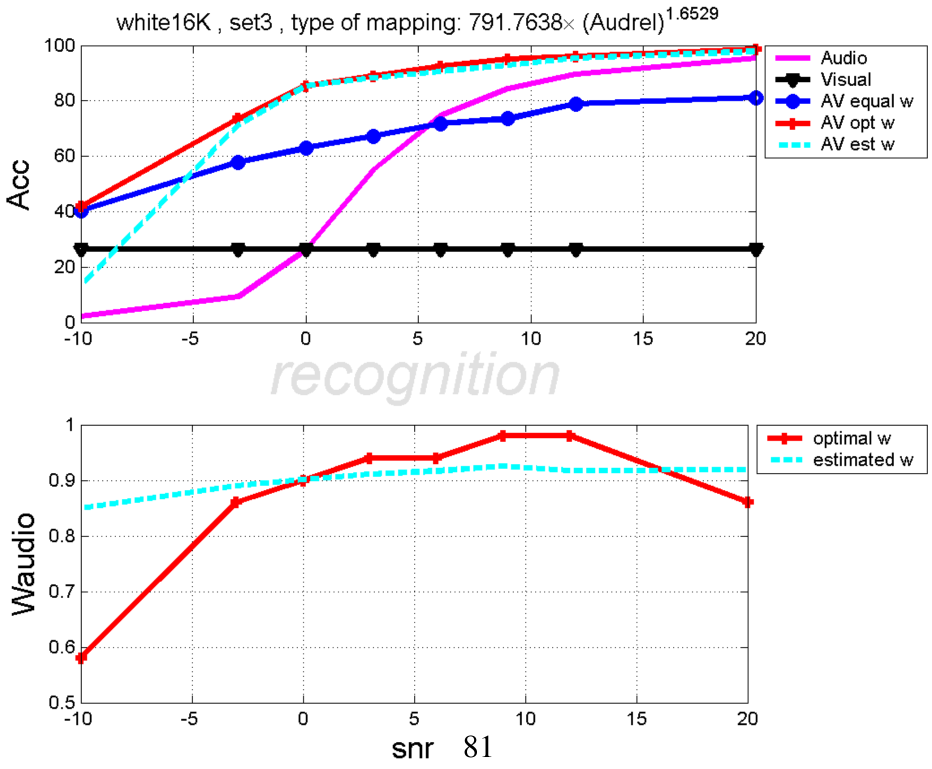
(q)


white16K , set3 , type of mapping: 738.2535× (Audrel)^2.6507
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
classification
Waudio
optimal w
estimated w
snr

(r)

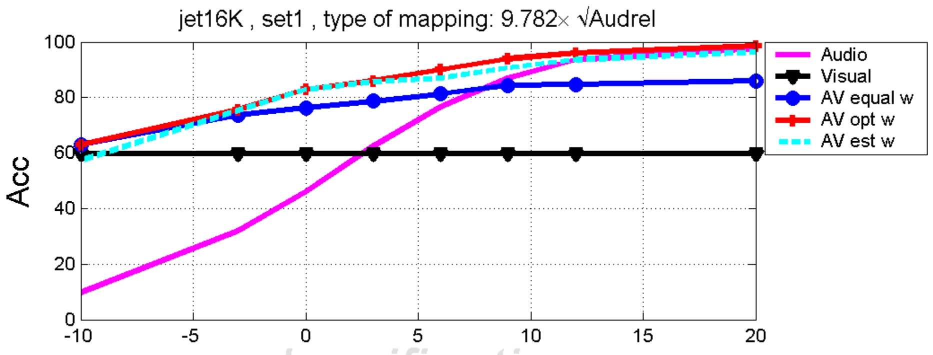

white16K , set3 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
classification
Waudio
optimal w
best est w
a × √(AudioRel)
snr

(s)

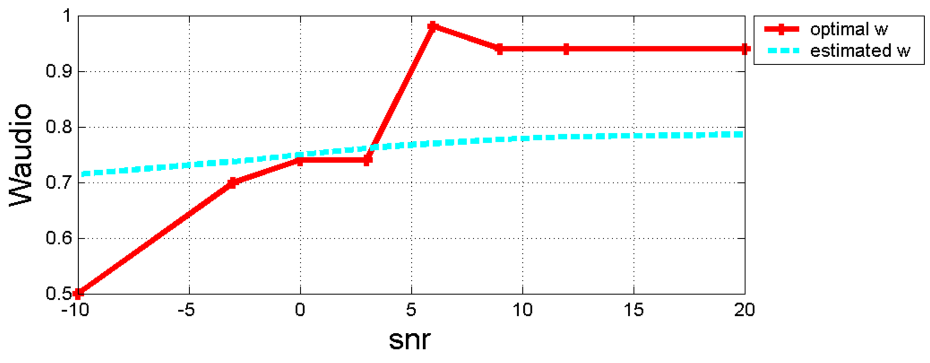

white16K , set3 , type of mapping: 791.7638× (Audrel)^1.6529
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
recognition
Waudio
optimal w
estimated w
snr

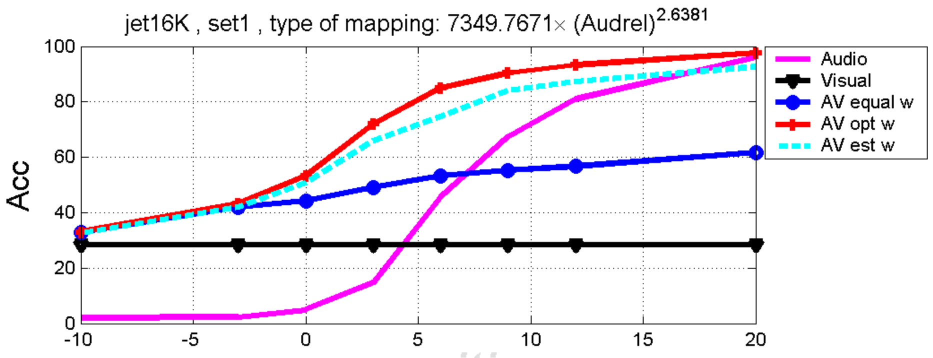
(t)

Jet Cockpit Noise

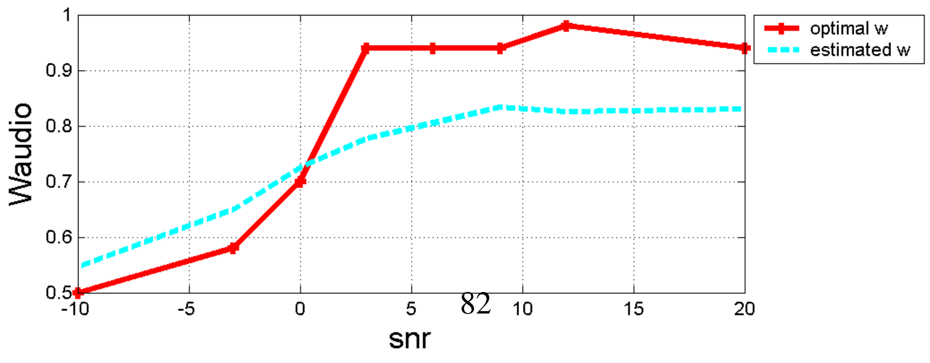


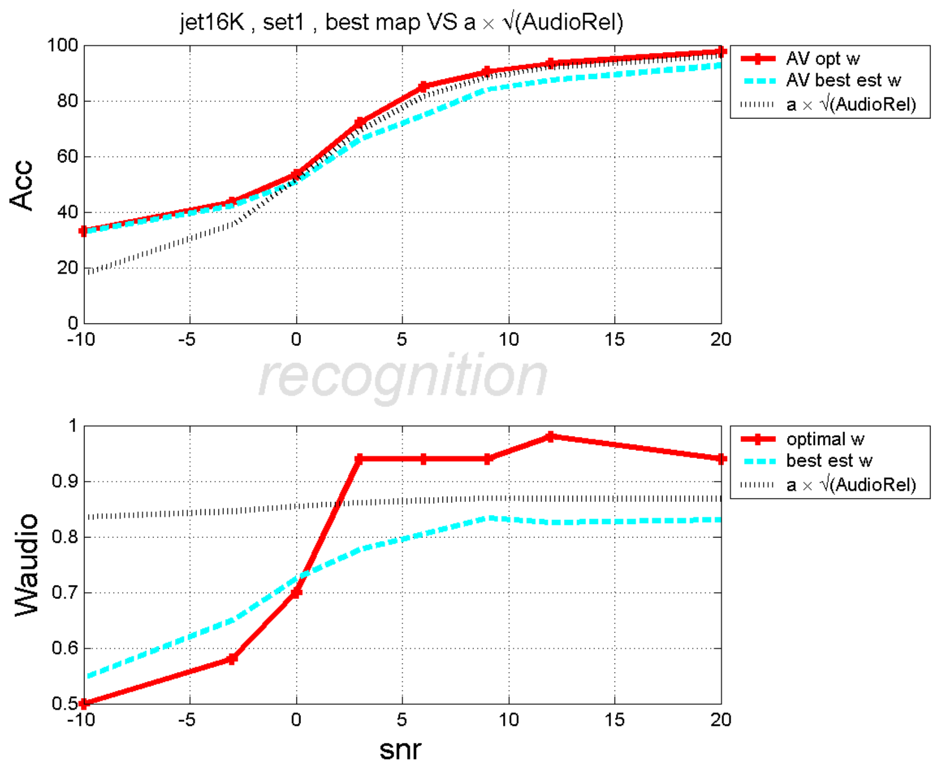
classification



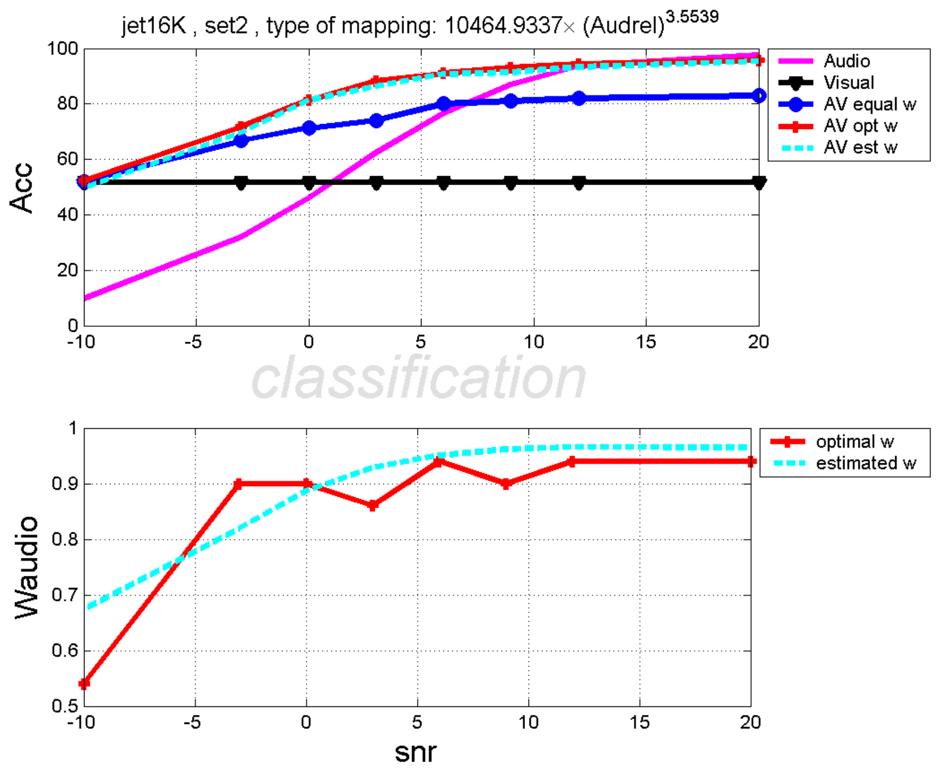


recognition

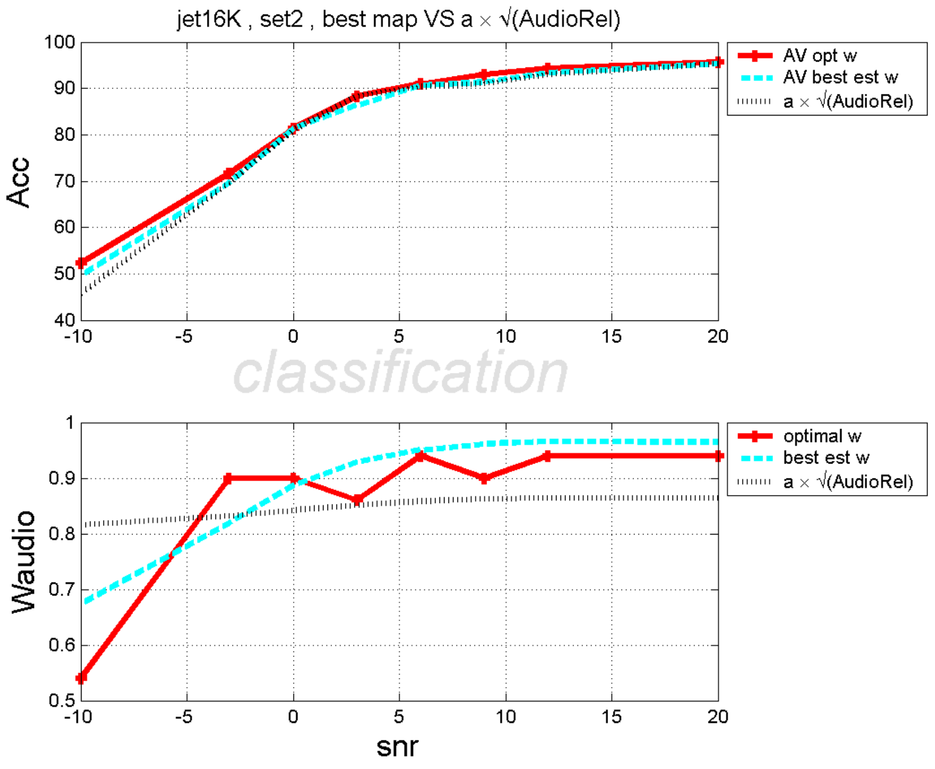


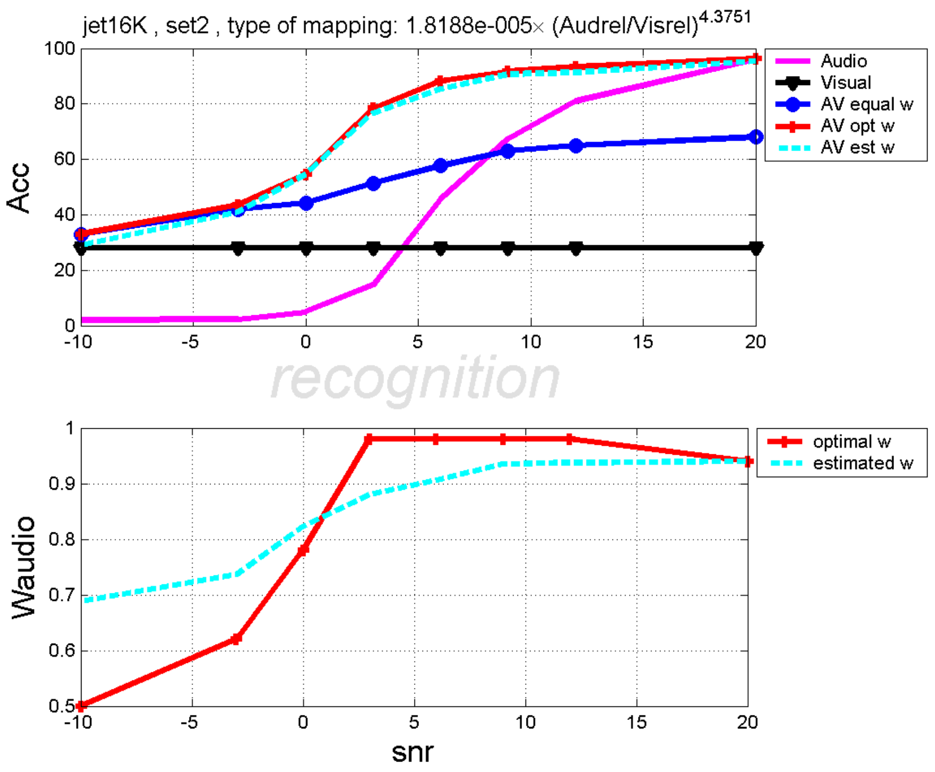


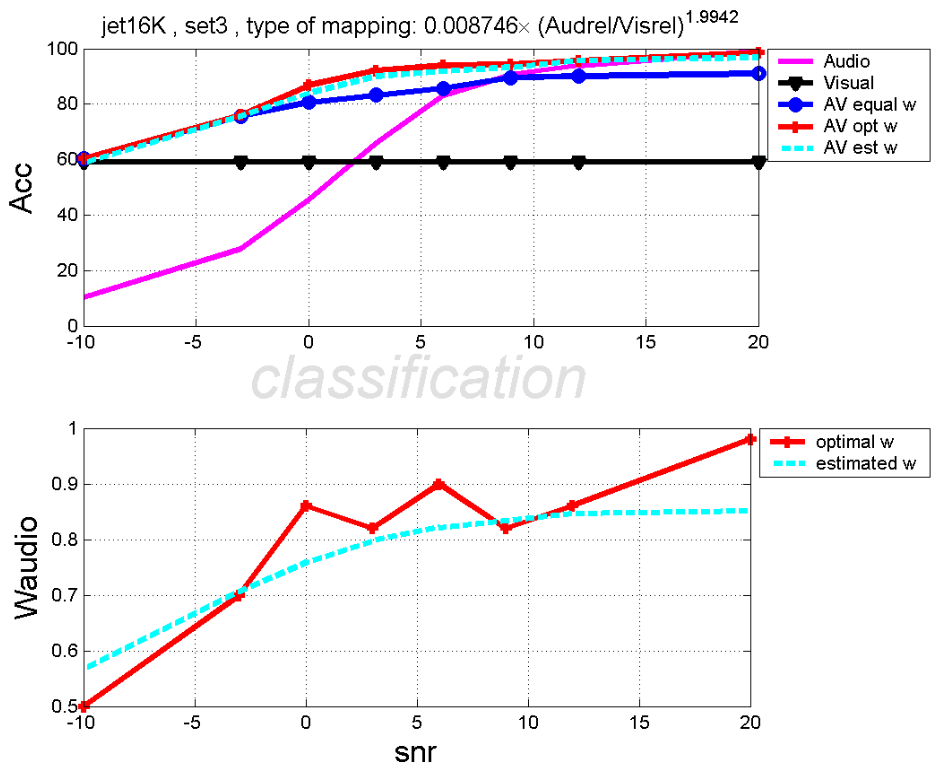
jet16K , set1 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
recognition
Waudio
optimal w
best est w
a × √(AudioRel)
snr

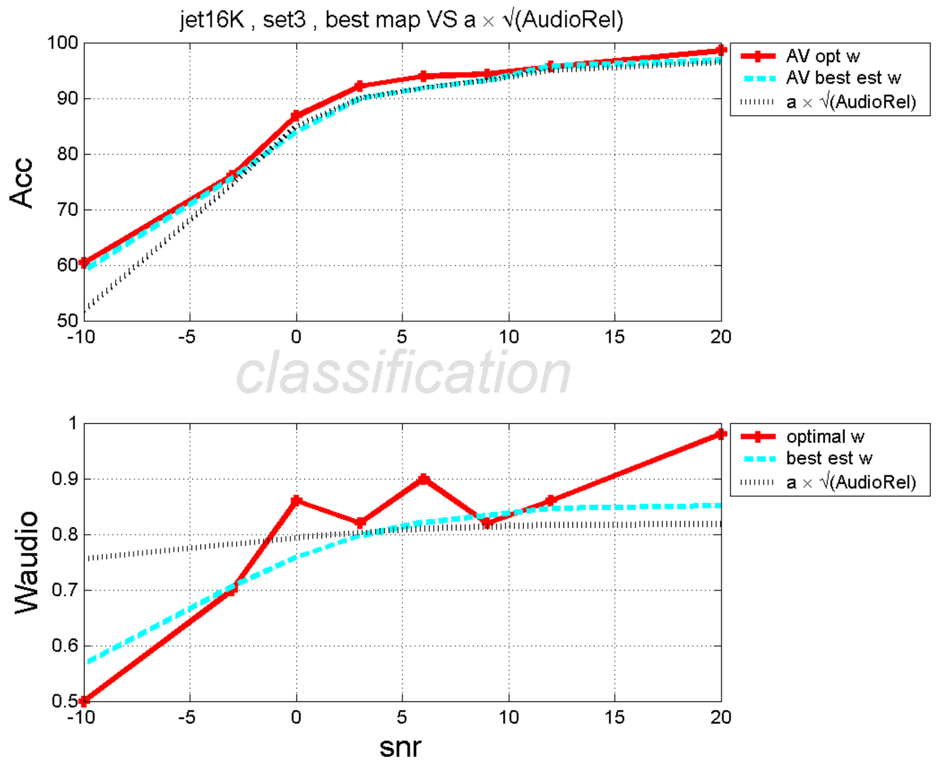


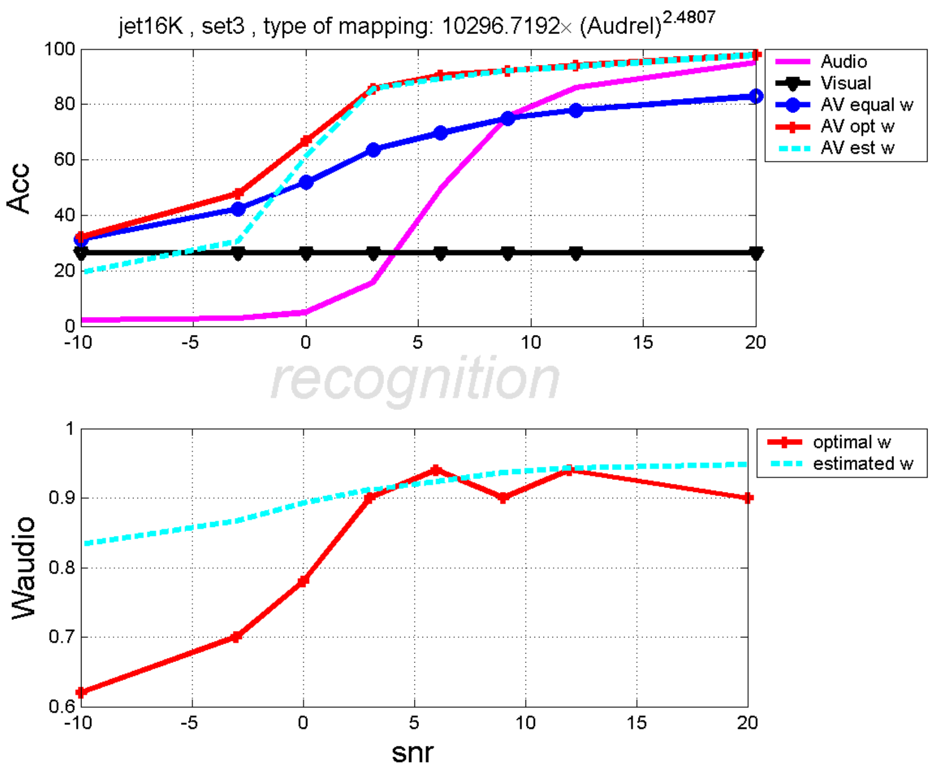
jet16K , set2 , type of mapping: 10464.9337× (Audrel)^3.5539
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
classification
Waudio
optimal w
estimated w
snr

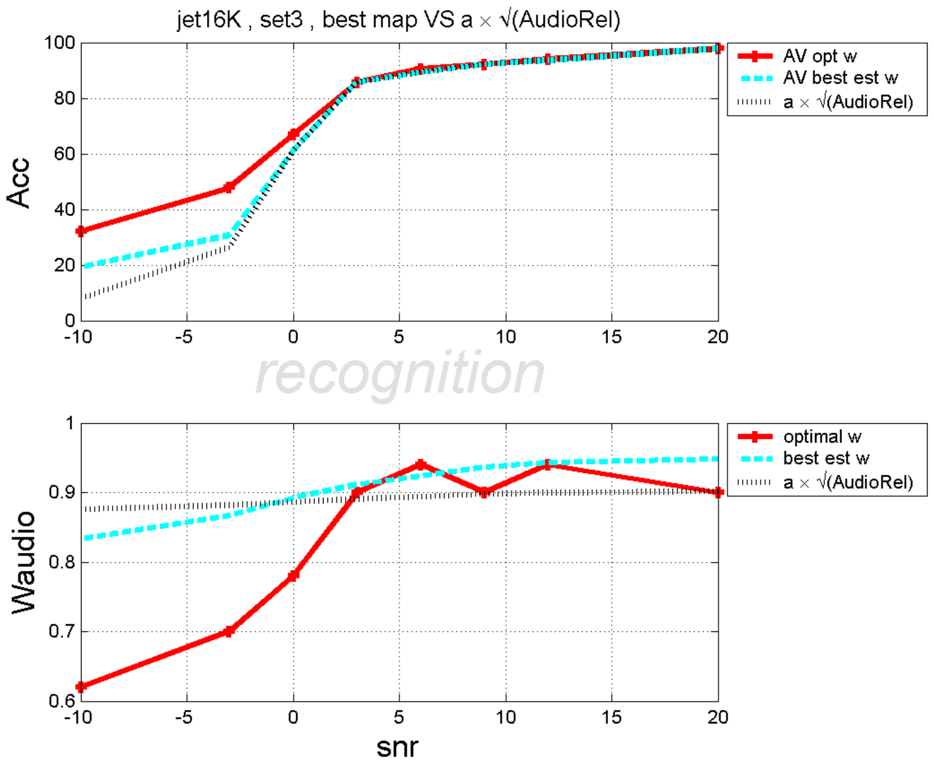

jet16K , set2 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
classification
Waudio
optimal w
best est w
a × √(AudioRel)
snr


jet16K , set2 , type of mapping: 1.8188e-005× (Audrel/Visrel)$^{4.3751}$
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
recognition
Waudio
optimal w
estimated w
snr

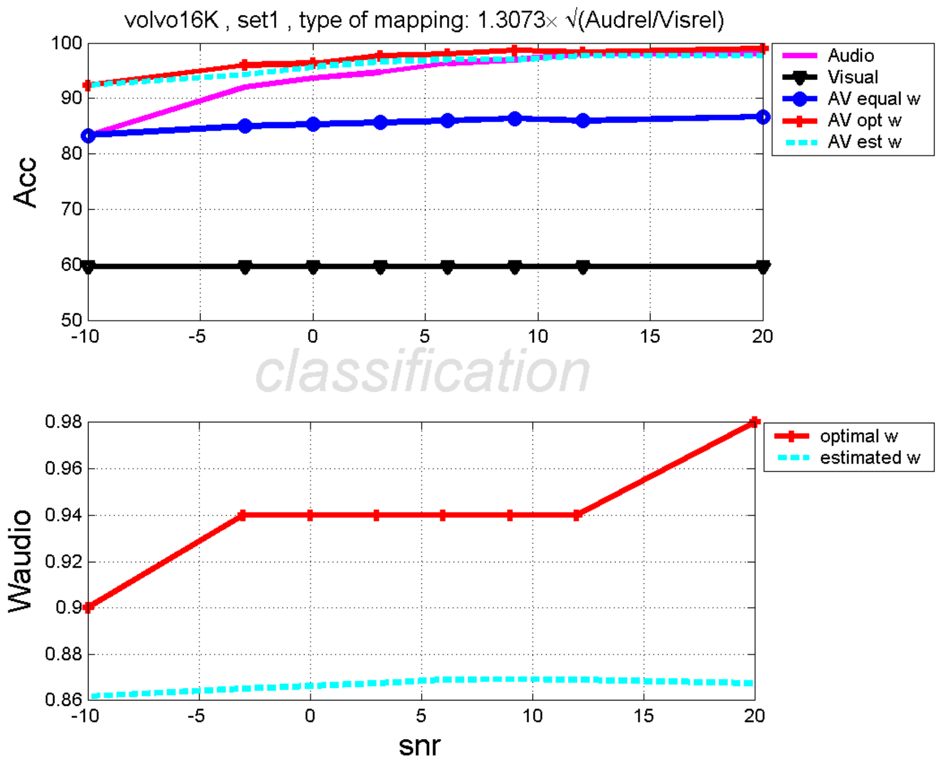

jet16K , set3 , type of mapping: 0.008746× (Audrel/Visrel)^1.9942
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
classification
Waudio
optimal w
estimated w
snr

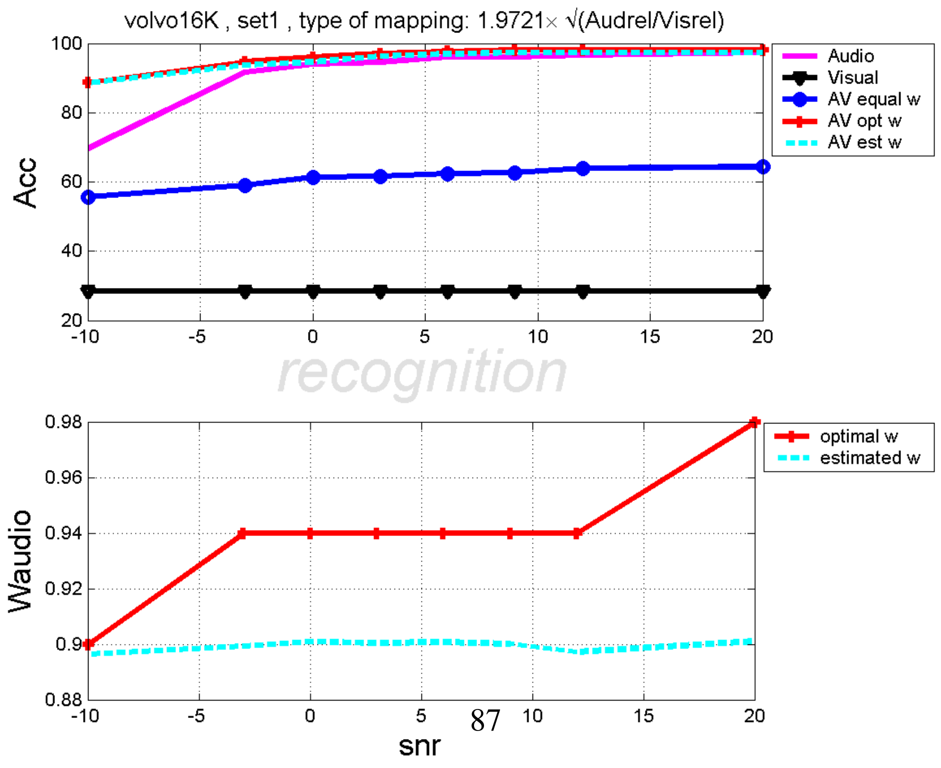

jet16K , set3 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
classification
Waudio
optimal w
best est w
a × √(AudioRel)
snr

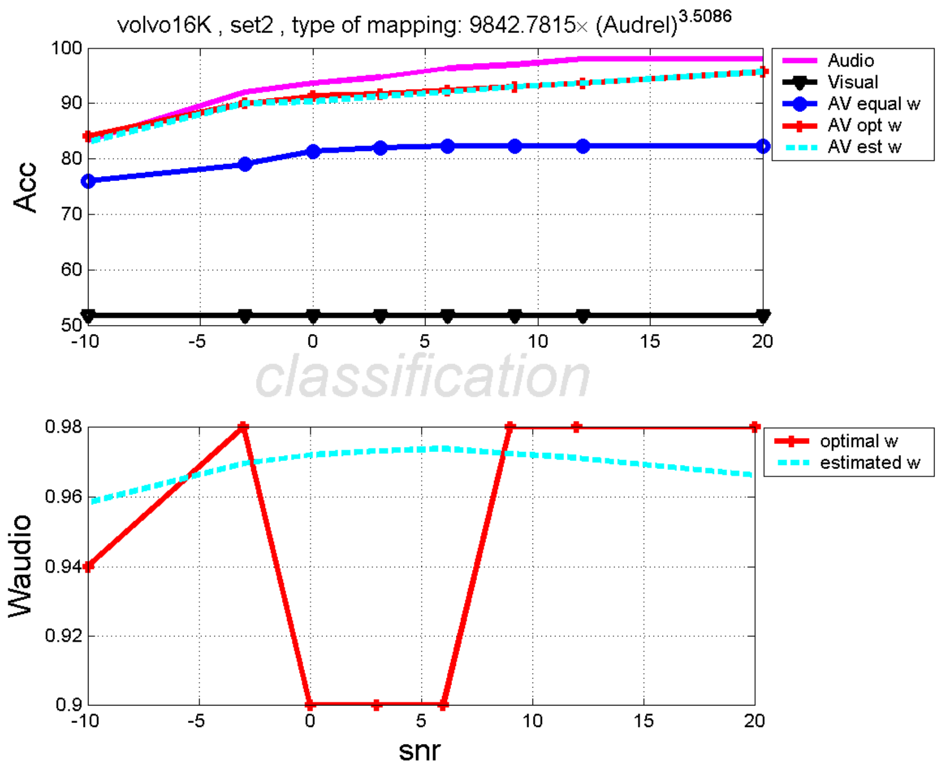

jet16K , set3 , type of mapping: 10296.7192× (Audrel)^2.4807
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
recognition
Waudio
optimal w
estimated w
snr

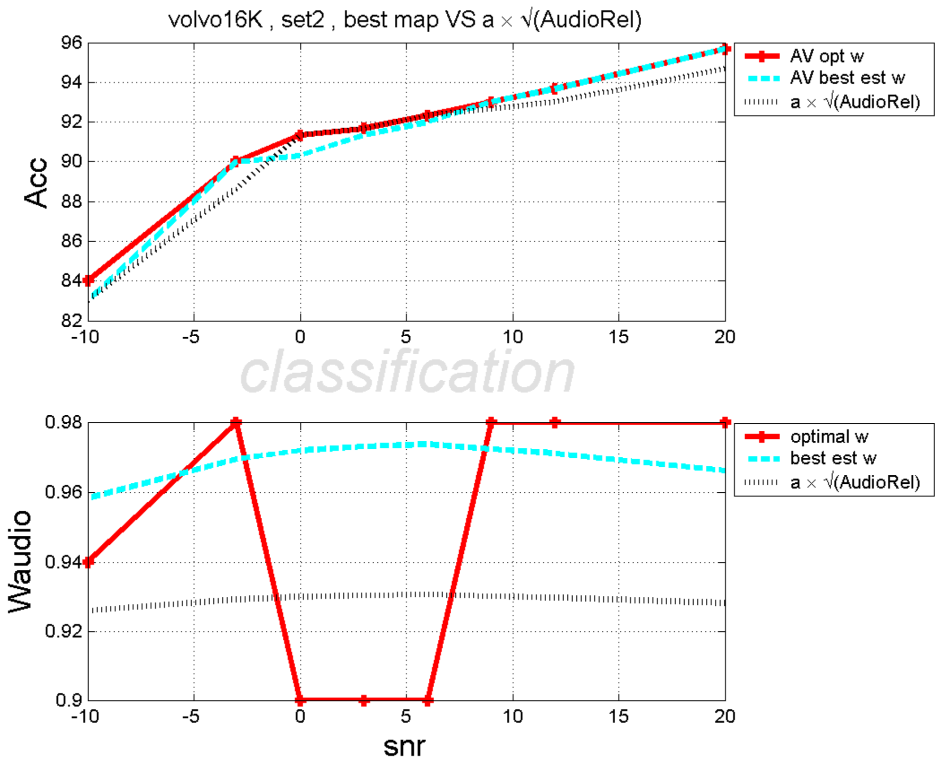

jet16K , set3 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
recognition
Waudio
optimal w
best est w
a × √(AudioRel)
snr

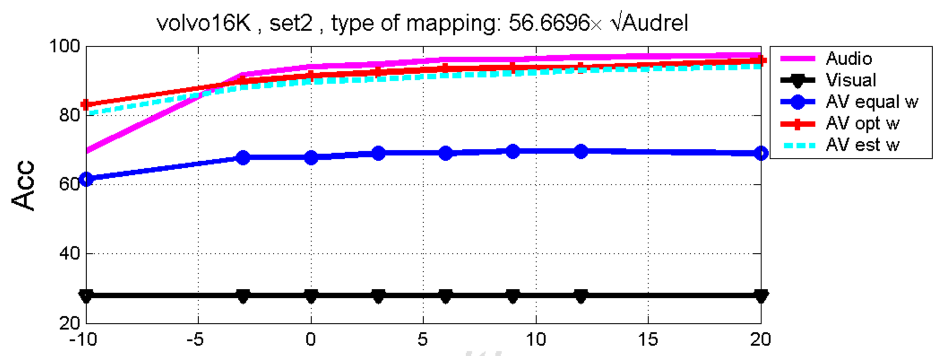
Car Interior Noise



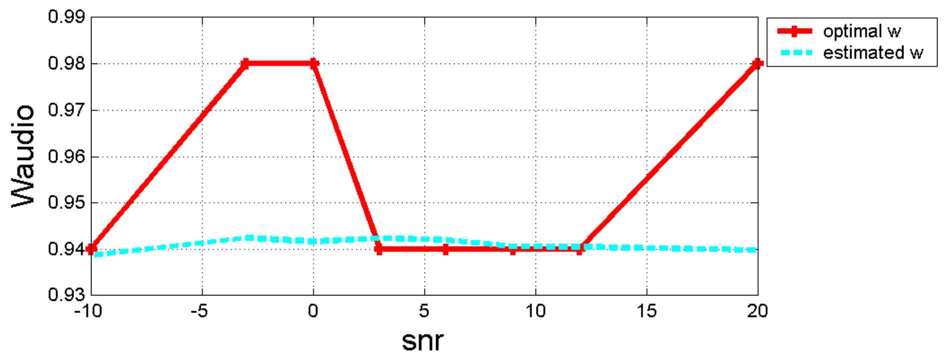


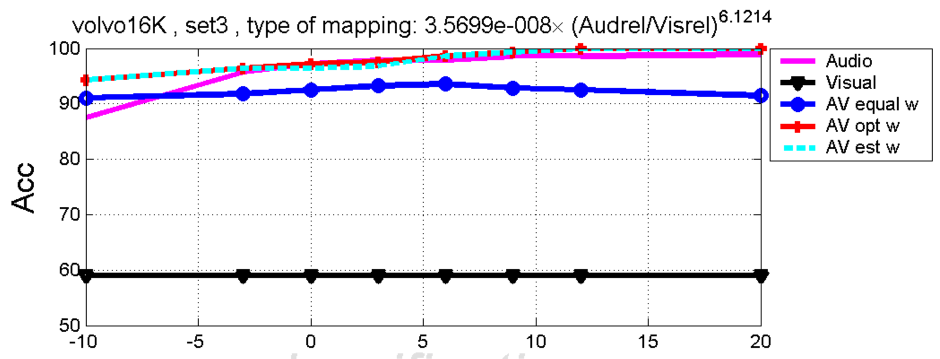

volvo16K , set2 , type of mapping: 9842.7815× (Audrel)^3.5086
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
classification
Waudio
optimal w
estimated w
snr


volvo16K , set2 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
classification
Waudio
optimal w
best est w
a × √(AudioRel)
snr

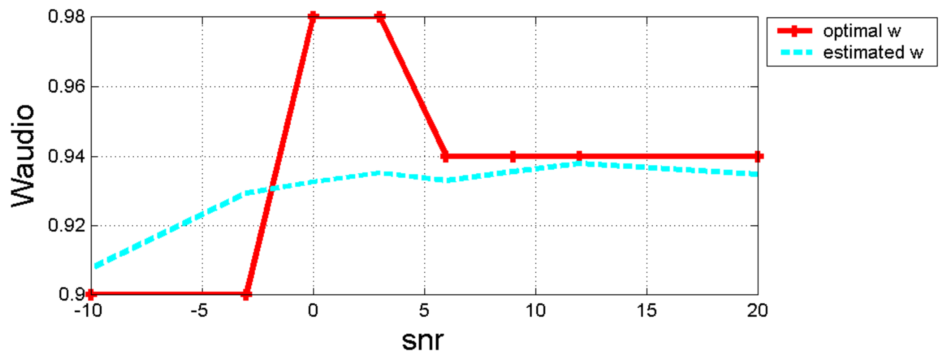


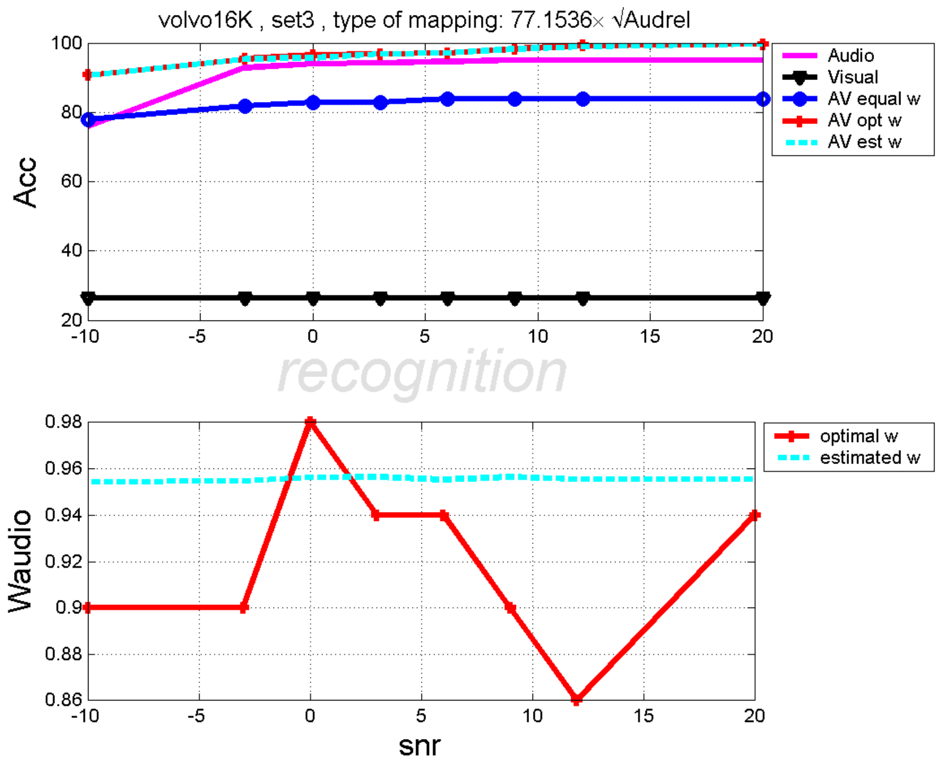
recognition

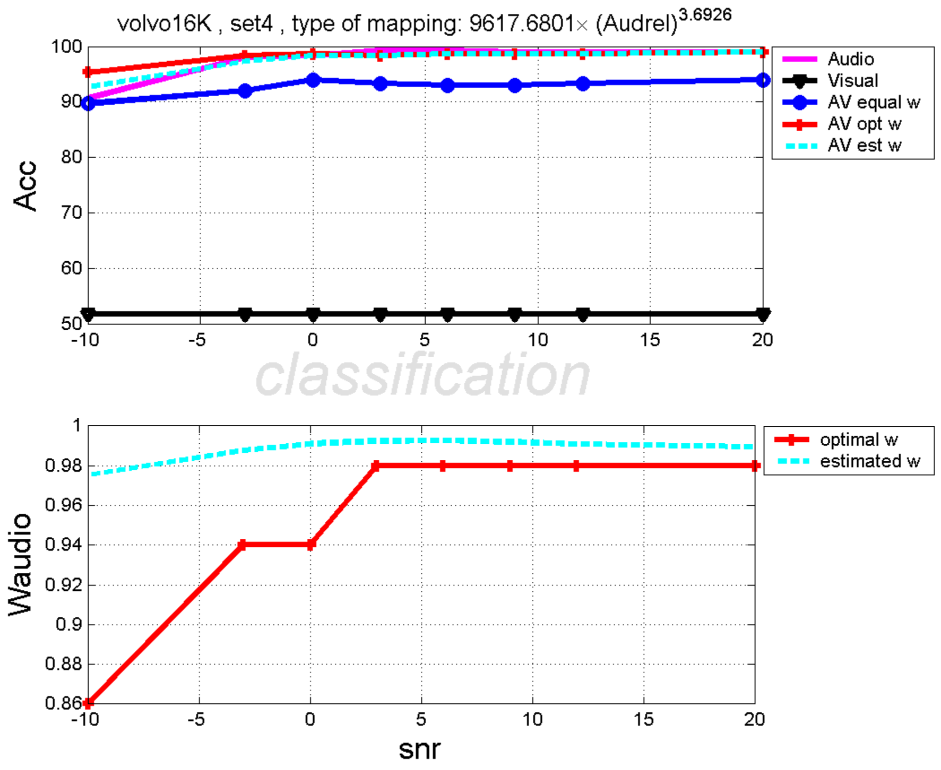




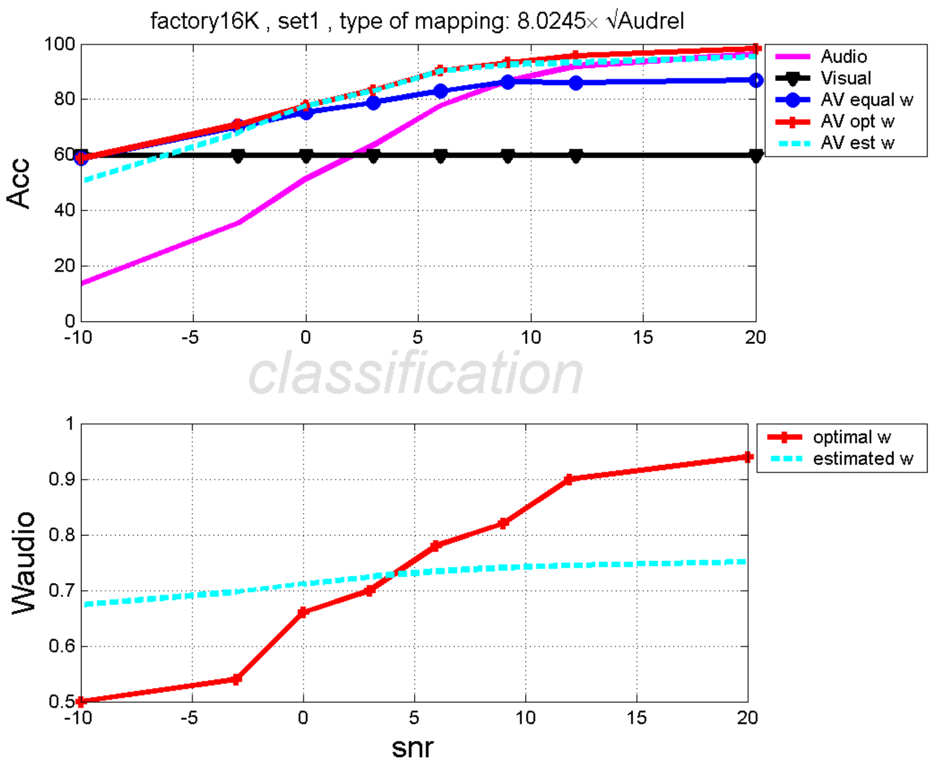
classification



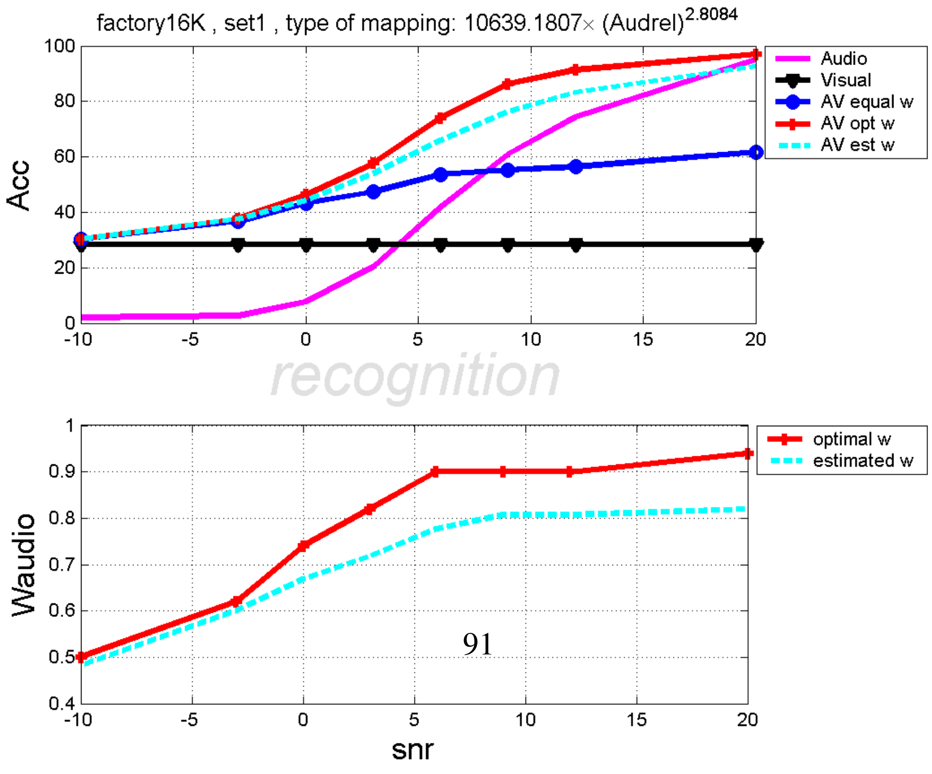

volvo16K , set3 , type of mapping: 77.1536× √Audrel
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
recognition
Waudio
optimal w
estimated w
snr


volvo16K , set4 , type of mapping: 9617.6801× (Audrel)^3.6926
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
classification
Waudio
optimal w
estimated w
snr

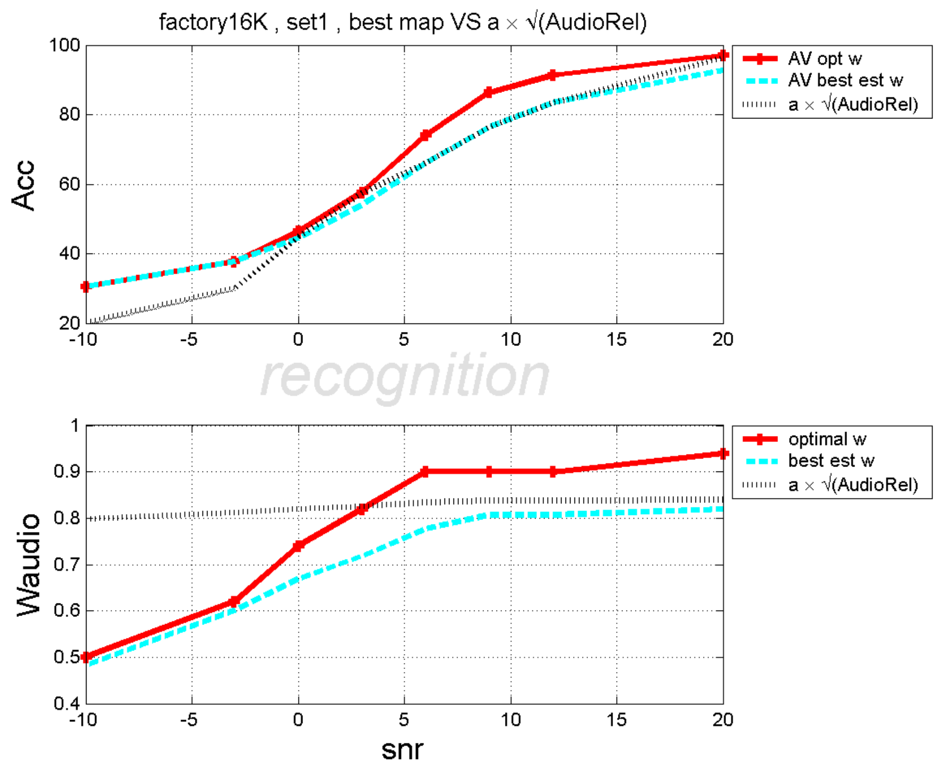
Factory Noise

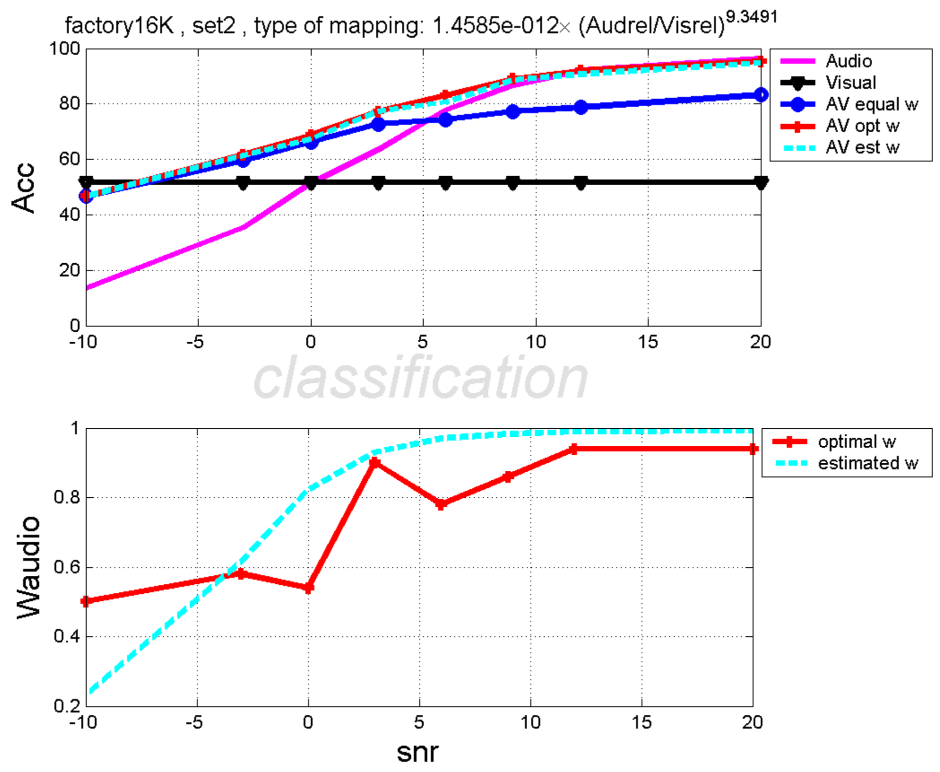


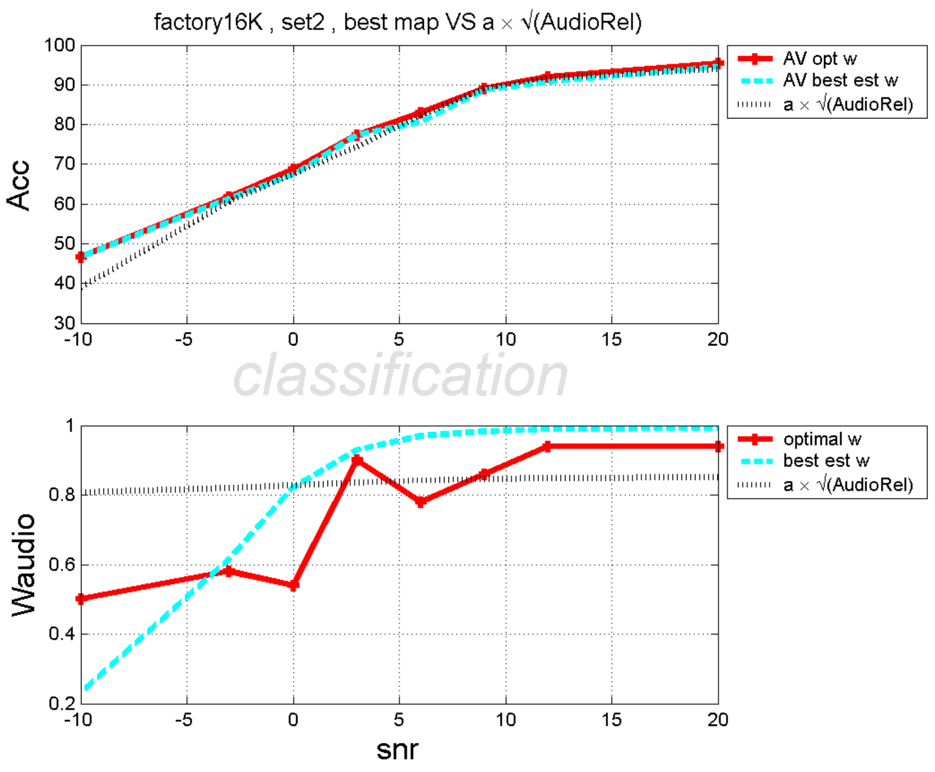
(u)

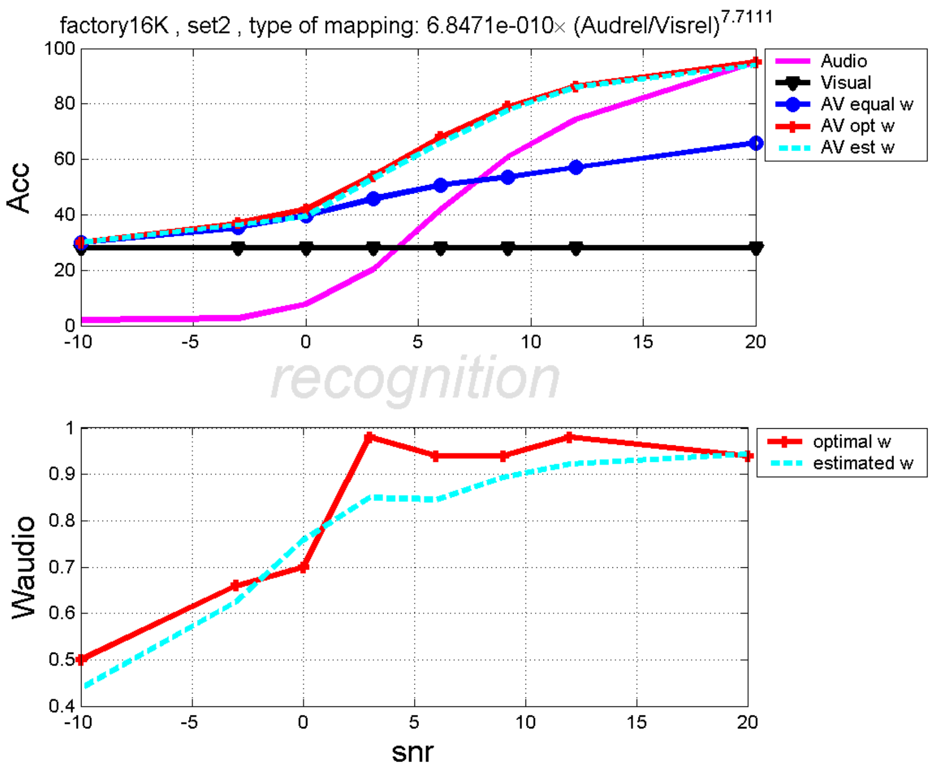


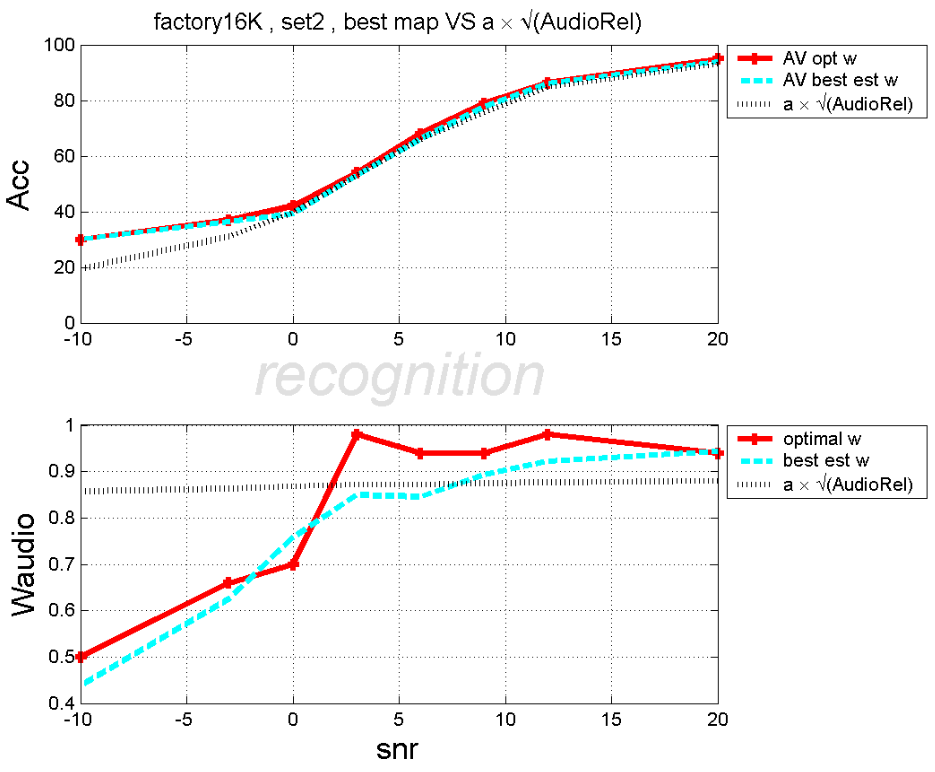
(v)

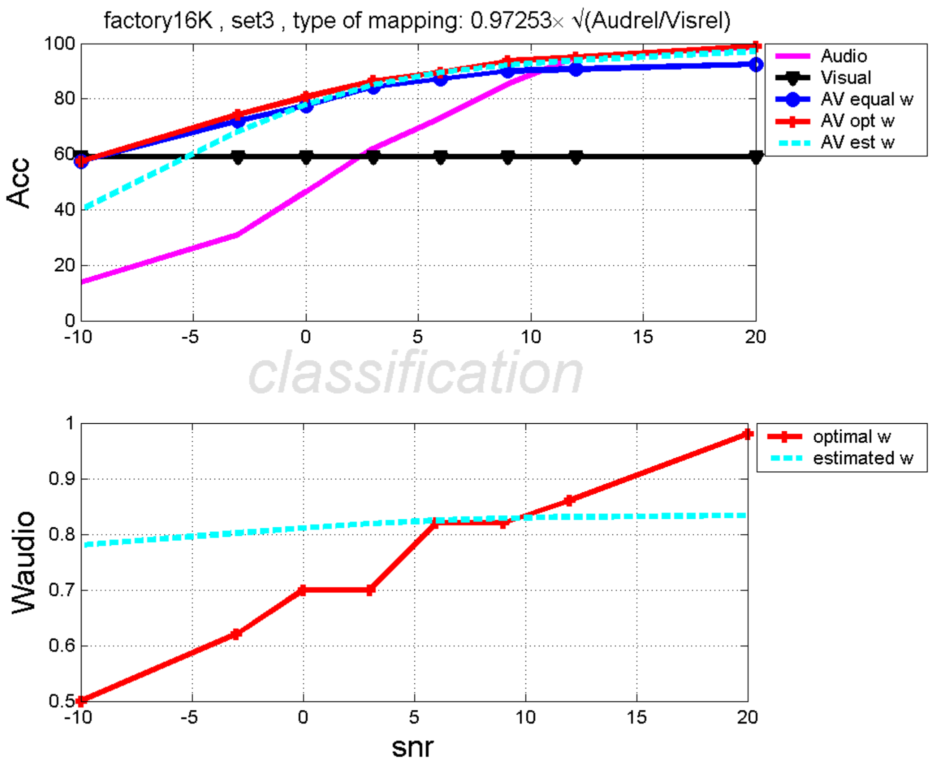

factory16K , set1 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
recognition
Waudio
optimal w
best est w
a × √(AudioRel)
snr

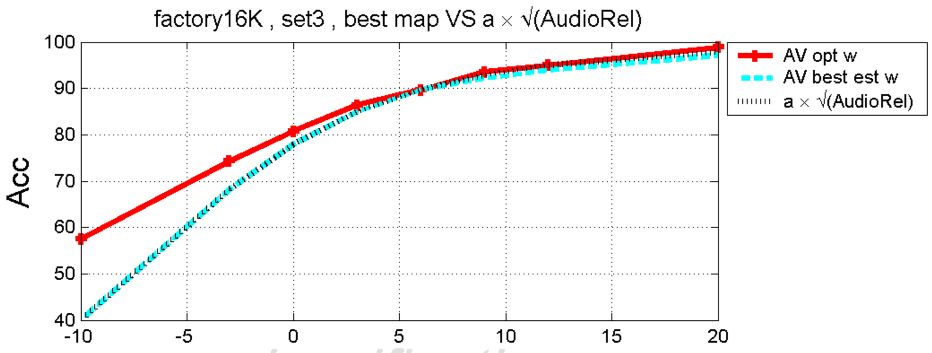

factory16K , set2 , type of mapping: 1.4585e-012× (Audrel/Visrel)^9.3491
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
classification
Waudio
optimal w
estimated w
snr


factory16K , set2 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
classification
Waudio
optimal w
best est w
a × √(AudioRel)
snr

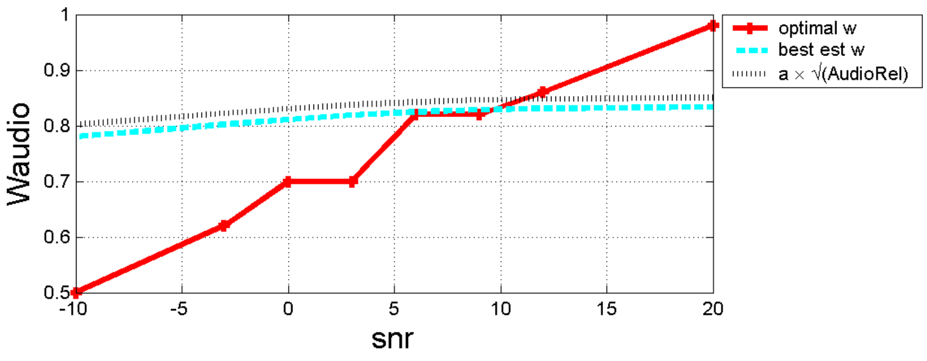

factory16K , set2 , type of mapping: 6.8471e-010× (Audrel/Visrel)^7.7111
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
recognition
Waudio
optimal w
estimated w
snr

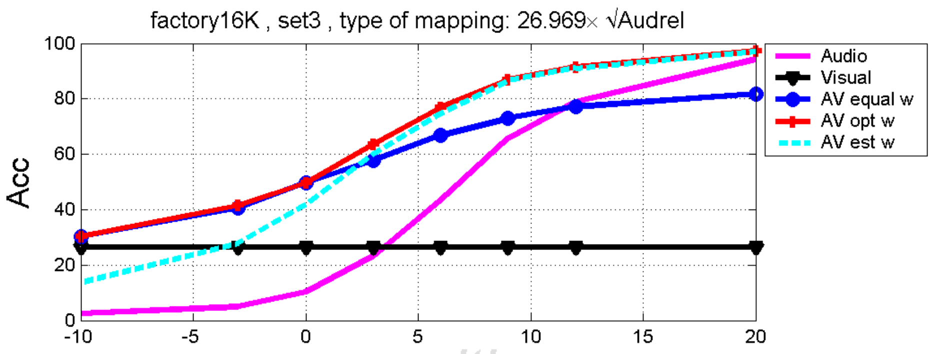

factory16K , set2 , best map VS a × √(AudioRel)
Acc
AV opt w
AV best est w
a × √(AudioRel)
recognition
Waudio
optimal w
best est w
a × √(AudioRel)
snr


factory16K , set3 , type of mapping: 0.97253× √(Audrel/Visrel)
Acc
Audio
Visual
AV equal w
AV opt w
AV est w
classification
Waudio
optimal w
estimated w
snr

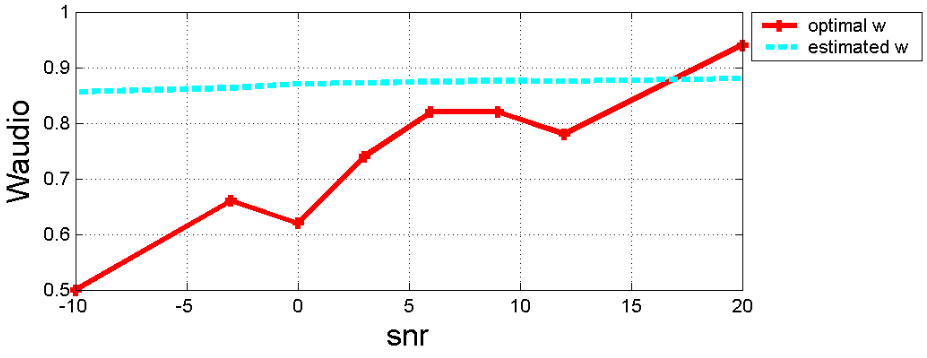


classification





recognition



noise / implementation	*babble*	*white*	*jet*	*car*	*factory*
Classification					
vs Audio-only	35.20	47.76	82.87	0.54	46.61
vs Baseline-AV	2.51	8.69	6.73	10.69	3.34
Distance from opt-AV	3.63	1.45	2.50	0.41	3.78
Recognition					
vs Audio-only	264.95	283.97	475.82	2.46	276.86
vs Baseline-AV	5.09	31.43	20.37	33.32	14.10
Distance from opt-AV	17.53	8.26	14.44	0.83	11.23

Table 1: Mean WER reduction % achived by “est-AV”

Bibliography

[1] Htk-hidden markov model toolkit. http://htk.eng.cam.ac.uk.

[2] *Machine Perception Toolbox.* Machine Perception Laboratory, University of California San Diago. http://mplab.ucsd.edu/grants/project1/free-software/mptwebsite/API/.

[3] H. Bourlard and S. Dupont. A new asr approach based on independent processing and recombination of partial frequency bands. *International Conference on Spoken Language Processing*, 1:426–429, 1996.

[4] S. Dimopoulos. Instantaneous stream weight computation in audio-visual speech recognition. In *Diploma Thesis*, Chania, Technical University of Crete, October 2007.

[5] Gurban, Mihai, Thiran, and Jean-Philippe. Using entropy as a stream reliability estimate for audio-visual speech recognition. In *16th European Signal Processing Conference*, Lausanne, Switzerland, 2008.

[6] Liming Chen Kun Peng, Su Ruan, and Georgy Kukhare. A robust algorithm for eye detection on gray intensity faces.

[7] A. Mashari, J. Sison, C. Neti, G. Potamianos, and J. Luettin. Modeling visual co-articulation for large vocabulary continuous visual speech recognition. In *ICASSP*, 2001.

[8] D. W. Massaro and D. G. Stork. Speech recognition and sensory integration. *American Scientist*, 86(3):236–244, 1998.

[9] M. McGrath and Q. Summerfield. Intermodal timing relations and audio-visual speech recognition. *American Scientist*, 77(2):678–685, February 1985.

[10] E.K Patterson., S. Gurbuz, Z. Tufekci, and J.N. Gowdy. Moving-talker, speaker-independent feature study, and baseline results using the cuave multimodal speech corpus. *Eurasip Journal on Applied Signal Processing*, 11:1189, November 2002.

[11] A. Potamianos, E. Sanchez-Soto, and K. Daoudi. Stream weight computation for multi-stream classifiers. In *ICASSP*, Toulouse, France, April 2006.

[12] G. Potamianos and C. Neti. Stream confidence estimation for audio-visual speech recognition. *ICSLP*, 3:746–749, October 2000.

[13] G. Potamianos, C. Neti, G. Iyengar, A.W. Senior, and A. Verma. A cascade visual front end for speaker independent automatic speechreading. *International Journal of Speech Technology*, 4:193–208, 2001.

[14] E. Sanchez-Soto, K. Daoudi, and A. Potamianos. Unsupervised stream weight computation in a segmentation task : Application to audio-visual speech recognition. In *International Conference on Signal Processing and Communications*, Dubai, UAE, November 2007.

[15] E. Sanchez-Soto, A. Potamianos, and K. Daoudi. Unsupervised stream weight estimation using anti-models. In *ICASSP*, Honolulu, Hawai, USA, April 2007.

[16] Patrice Y. Simard, Yann A. Le Cun, John S. Denker, and Bernard Victorri. Transformation invariance in pattern recognition tangent distance and tangent propagation. In *Lecture Notes in Computer Science*, pages 239–274. Springer, 1998.