Institute of Microtechnology
University of Neuchâtel
Switzerland

Detection and Recognition of Impulsive Sound Signals

Alain Dufaux

*Thèse présentée à la faculté des sciences
de l 'université de Neuchâtel
pour l 'obtention du grade de docteur ès sciences*

January 2001

Institute of Microtechnology
University of Neuchâtel
Switzerland

Detection and Recognition of Impulsive Sound Signals

Alain Dufaux

*Thèse présentée à la faculté des sciences
de l 'université de Neuchâtel
pour l 'obtention du grade de docteur ès sciences*

January 2001

IMPRIMATUR POUR LA THESE

Detection and Recognition System for Impulsive Audio Signals

de M. Alain Dufaux

UNIVERSITE DE NEUCHATEL

FACULTE DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel sur le rapport des membres du jury,

MM. F. Pellandini (directeur de thèse), H. Hügli, M. Ansorge et R. Ryser (EPF Lausanne)

autorise l'impression de la présente thèse.

Neuchâtel, le 13 février 2001

Le doyen:

J.-P. Derendinger

Abstract

This thesis is dedicated to the development of automatic methods for *detecting and recognizing wideband impulsive sounds*. An extensive database with more than 1000 sound samples has been built. This database is made of 10 diversified sound classes connected with surveillance and security applications: door slams, glass breaks, human screams, explosions, gunshots, dogs barking, low bangs, phone rings, children voices and machines running. Furthermore, one supplementary class with varied types of sounds was built, to simulate different background noises as car traffic, music, or conversations.

At first, the proposed system performs a robust detection of the impulsive sounds that may occur. The detection methods are based on a continuous measure of sound energy variations. Those high-performance techniques can be adapted to the background noise level, and can encompass important and difficult environments, with very low signal-to-noise ratios, such as -10 dB.

Then, robust statistical methods for the classification of the previously detected sounds have been proposed and used. Those methods perform a time-frequency analysis of the sounds, similar to the analysis of the human ear. The statistical distributions of the extracted parameters are then modeled with mixtures of gaussians (GMM), Hidden Markov Models (HMM), or Perceptron Neural Networks. The recognition rates obtained with such methods exceed 97% of correct classification for the 10 classes mentioned above. When the intra-class variance of the sounds is small, the performance can even reach 100%.

When impulsive sounds are disturbed by a difficult background environment, those rates get lower, but they still reach 80%, even when the signal-to-noise ratio decreases to 0 decibels. This result was made possible thanks to the quality of the signal analysis, and because of advanced techniques that use models adapted to the noise level. The perspective of using such a recognition system in surveillance and security applications (alarm detection) is therefore possible, on the condition that sound class models could be learned and built at the place to control. In such a situation, particular properties of the sounds to recognize would be hold in the models, providing very good results.

In the future, further work should be dedicated to the problem of rejecting sounds that do not belong to the set of considered classes. This evolution is

required in the perspective of using such a recognition system in more complex sound environments, for example in the street. Such a development of the sound recognition system could make it useful for the assistance of hearing-impaired persons, at home in their daily occupations, or outside in the street, to make them aware of potential dangers.

Résumé

Les travaux effectués dans le cadre de cette thèse ont été consacrés au développement de *méthodes automatiques pour la détection et la reconnaissance de signaux audio à large bande, de type impulsif*. Une importante base de donnée contenant plus de 1000 sons a été créée. Elle est constituée de bruits liés aux applications de surveillance et de sécurité, à savoir des fermetures de portes, des bris de verres, des cris, des explosions, des coups de feu, des aboiements, des coups sourds, des sonneries de téléphone, des voix d'enfants et des machines en fonction, qui forment ainsi un ensemble de 10 classes bien diversifiées. De plus une classe supplémentaire, constituée de bruits très variés, a été construite afin de simuler différents fonds sonores perturbés tels que musique, trafic automobile, ou conversations.

Dans un premier temps, le système proposé cherche à détecter d'éventuels bruits de type impulsifs, en utilisant des *techniques basées sur la mesure des variations de l'énergie du signal*. Grâce à ces méthodes performantes, capables de s'adapter au type de fond sonore, les impulsions peuvent être détectées dans des environnements très perturbés, de rapport signal-sur-bruit extrêmement faible (–10dB).

Ensuite, des *méthodes statistiques performantes et robustes* ont été mises en oeuvre pour la classification des sons préalablement détectés. Ces méthodes effectuent une analyse temps-fréquence des sons, similaire à celle de l'oreille humaine. Les distributions statistiques des paramètres ainsi obtenus sont ensuite modélisées à l'aide de mixtures de gaussiennes (GMM : "Gaussian Mixtures Models") ou de modèles de Markov cachés (HMM : "Hidden Markov Models"). Les taux de reconnaissance obtenus à l'aide de ces techniques surpassent les *97% de classification correcte*, pour les 10 classes de sons mentionnées ci-dessus. Lorsque les signaux internes à chaque classe sont très bien définis et peu diversifiés, les performances peuvent même atteindre 100%.

Lorsque les signaux sont perturbés par un environnement sonore difficile, ces taux deviennent plus faibles, mais restent toutefois supérieurs à 80% même quand le rapport signal sur bruit diminue aux alentours de 0 décibels. Ces résultats sont possibles grâce à la finesse de l'analyse des signaux effectuée, ainsi qu'à la mise en place d'une technique intégrant des modèles adaptés au type et niveau de bruit ambiant observé. Ainsi, la qualité de ces

résultats permet d'entrevoir positivement l'utilisation de telles méthodes dans le cadre *d'applications de surveillance* (sécurité et détection d'alarmes), pour autant qu'un apprentissage préalable soit possible, pour construire des modèles de bruits caractéristiques du lieu à surveiller.

Dans le futur, de nombreux efforts devraient être consacrés à l'élaboration de méthodes permettant un meilleur rejet des sons extérieurs aux catégories d'intérêt. Une évolution favorable de ce problème sensible permettrait d'envisager l'utilisation de tels systèmes dans des milieux plus diversifiés (par exemple dans la rue), pour apporter une aide aux personnes malentendantes.

Table of Contents

1. Introduction

This thesis is dedicated to the algorithmic study and design of an automatic detection and recognition system, for impulsive sounds.

After an overview of the system philosophy, first trying to detect some impulsive acoustical event and then to identify it, this chapter presents the major motivations for considering the sound recognition problem. A description of the most two important applications is provided: the automatic alarm detection/identification for surveillance systems and the assistance to people affected in hearing capabilities.

The main publications about state-of-the-art studies in automatic audio signal recognition are considered, and the scope of this particular research is specified. The end of the chapter is devoted to the description of the report organization.

This Ph.D. research was first suggested during a CTI ("Commission for Innovation and Technology" in Switzerland) project, whose theme was the compression of wideband audio signals for surveillance tasks. The work consisted of compactly recording the acoustic environment in the controlled scene, when an alarm situation occurred. In this way, some information of the event was kept as a legal witness. An improvement of the system was later proposed, trying to automatically detect and validate those alarm situations, by means of some audio signal analysis. However, it rapidly became obvious that this would be a sizeable work and a scientific challenge, better to be considered in more detail. As a consequence, a new basic research project was defined for this study, supported by the Swiss National Foundation for Scientific Research.

1.1 Detection and recognition system

Some terminology precision is required at first. All along this text, both *sound recognition* and *sound classification* terms will be used with the same

meaning. They include a notion of discrimination capability between different types of impulsive sounds. On the opposite, in a *sound detection* system, the only given information is that something special has happened in the acoustical environment, typically a sudden energy increase. No further detail about the kind of produced sound is known. In some sense, detection can be considered as a preliminary operation, required for recognition, as shown on Figure 1.1. That's the reason why *recognition system* is sometimes used globally, to characterize the complete process.

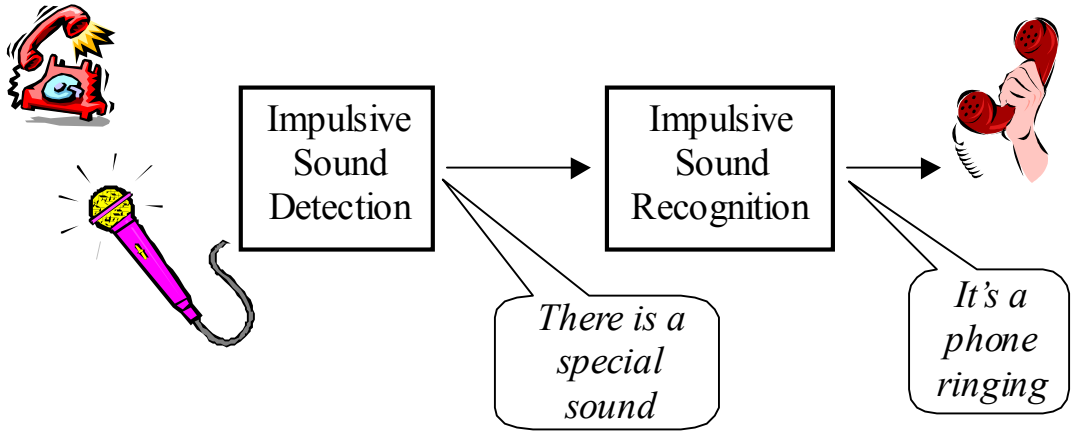


Figure 1.1: *Detection and recognition system*

Figure 1.1 shows the recognition system philosophy adopted in this work, involving two steps:

- The detection block is continuously tracking some important changes in the acoustical environment.
- If some audio disturbance is found, the corresponding signal is transmitted to the recognition stage, for identification.

In this way, the detection block works as a recognition pre-processing. It operates a pre-selection of the candidate signals to be identified and prevents the recognition process from being constantly activated. This feature of the system is very important, because of the high complexity of most reliable recognition algorithms. Furthermore, when the identification functionality has to be implemented in a portable system, a low-power consumption and a fast processing are generally required. Therefore, the design of such a system seems to be the best compromise, using good but complex recognition methods only when necessary.

1.2 Motivations

Beside the purely scientific interest, two major domains of application do motivate the study of impulsive sound recognition. They are:

- The automatic alarm detection and identification for surveillance systems, based on the acoustic environment.
- The assistance to disabled or elderly persons affected in their hearing capabilities.

In surveillance and security applications, an important contribution to alarm detection and alarm verification or validation can be supplied, using sound recognition techniques. In particular, these methods could be helpful for intrusion detection in places like bank offices, stores, and private homes, or for the supervision of public premises exposed to person aggressions (e.g. parking garages). In all these cases, a recognition system can simply report an abnormal acoustic activity. More interestingly, it could further identify sounds like glass breaks, door slams, gunshots or human screams. Sometimes, the alarm is triggered by other types of detectors (e.g. temperature or video-based), and the sound recognizer would be associated to these other modalities, to verify and validate the alarm, with the purpose of decreasing the global false alarm detection rate.

In telematics applications, a sound recognition system can offer assistance to disabled and elderly persons affected in hearing capabilities, helping them to keep or recover some independence in their daily occupations. For example, typical indoor sounds like acoustic alarms and door or telephone rings could be signaled to a person who is hard of hearing, using some tactile or visual means. Outdoors, a portable recognition system could be used to catch the attention of a deaf person, in case of a potential dangerous situation. Actually, a robust recognition system could even identify typical urban sounds like car horns, tramway bells, sirens, explosions, or dog barks, and rapidly warn the disabled person.

Otherwise, interest for sound recognition algorithms have also been mentioned in domains like the study of environmental noise pollution. Investigations to try and reduce this increasing problem have been reported [Couv97], using the automatic detection and recognition of different kind of disturbing noises. After identification, each classified sound was stored (eventually in a compressed form) to build a database, this database being

useful for further noise property analysis and statistical studies. In such an application, the considered sounds were generally stationary.

More recently, another promising multimedia application of sound identification is arising with the automatic search (content-based retrieval) of audio databases [Wold96], available on the Internet for example. In this case, the type of considered sounds could be extended to all kind of acoustical events, such as sound effects or musical samples. Furthermore, the recognition algorithms could be used to automatically classify some new sound for adjunction to the database. Techniques are also being developed for the automatic indexing and segmentation of those databases.

To conclude this section, one can see that in most interesting application domains, the elaborated recognition system will have to satisfy some global quality level, if it is intended for practical use. This quality level is reflected by the following properties:

- a low detection and recognition error rate and a good reliability;
- a good robustness to background noise disturbance;
- a low complexity of the involved algorithms for portable feasibility;
- a fast processing time.

Simultaneously satisfying those four constraints is the scientific challenge of impulsive sound recognition, that will be kept in mind all along this study.

1.3 State-of-the-art

The automatic recognition of sounds is based on both traditional pattern recognition theories [Fuku90, Theo98] and audio signal analysis methods [Rabi93, Colo96]. In the recent past, such techniques have been widely applied to some particular physiological sounds (respiratory [Pesu96] and heart valve sounds [Bent97], animal sounds [Cogg98] or vocalization [Ashi93]), and especially to the speaker identification problem [Camp97]. For that reason, potentially interesting techniques could be inspired from the speaker recognition world. However, in the case of impulsive sounds, a good recognition system should take care of the highly non-stationary properties of the signals, and the developed methods should be designed, considering temporal dynamics with great attention.

Few studies have been published concerning automatic sound recognition. In 1988, the Institute for Signal and Information Processing at the ETH Zurich starts working in the field, proposing a "Sound alerting aid for the profoundly deaf" [Uvac88a, Uvac88b]. This first study is mainly devoted to the tactile system, while a rather basic statistical classifier is being used, exploiting mixed spectral and temporal signal features. Later, between 1993 and 1995, the same laboratory explores new classifier solutions, using neural networks [Lebe93, Osun93, Osun95] and hidden Markov models [Ober95]. In all cases, four classes of stationary signals are considered: bells, horns, phone and door rings. It comes out that the hidden Markov models solution gives a better recognition rate (94%) than the diverse tried neural networks (~80%).

Other works have been reported, dealing with rather stationary sounds: Ringing sounds classification [Para90], helicopter noise identification [Cabe92], and recognition of music instruments [Eron00] or music types [Solt99]. An excellent and extensive thesis was published in 1997 by Couvreur [Couv97], including a complete bibliographic review of the sound recognition domain, and considering the identification of environmental sounds for noise pollution problems.

In 1992, a first article [Wood92] discusses the matter of discriminating between three types of natural impulsive sounds. The particularity of the study is the use of two independent hidden Markov models in each class, one for the LPC (linear prediction coefficients) spectral shape and one for its gain. A good recognition rate of 96% is obtained for the three classes. One year later, another interesting study [Gold93] mentions excellent results for the recognition of numerous mixed stationary and impulsive environmental sounds, using standard statistical classification procedures. However, this work is presented as a preliminary study, and it uses very small databases with few and well-isolated signals in each class.

Finally, two papers concerning the classification of acoustic transients hold one's attention. In [Delf98], DFT and Wavelet based features are tested for musical applications like the identification of piano attack phases. In [Shie90], although no information is given on the databases used, very interesting results are reported with hidden Markov models. For the first time, tests under gaussian white noisy conditions are referred to, with encouraging results when noise is taken into account during training stage.

It must be noted that none of these publications concretely addresses the problem of a prior detection of the impulsive signal. In the detection domain, elaborated works have been reported in fields like industrial installation

surveillance, and automatic failure detection. The involved techniques often try [Bass86] to detect some change in the statistical properties of the audio signal. However, those techniques are generally too complex and not adapted for use as simple pre-processing in a real-time detection system. Other simple and signal-processing familiar methods, based on some energy temporal evolution, should better be considered, like the median filter detection scheme proposed in [Rice98].

1.4 Scope and organization of this research

In this thesis, emphasis is placed on the overall system design, involving both detection and recognition aspects. The goal is to study the feasibility of a rather reliable system, with good recognition performance, both under clean and real world noisy conditions. For that, different detection methods, signal analysis schemes, classifier architectures, and techniques for improving the robustness against noise are considered and explained in a way to be accessible to readers that are not involved in recognition problems. The following chapters will make a review of the major involved principles and theories, and describe in detail the numerous realized experiments.

Chapter 2 introduces the way of building sounds databases and describes the types and properties of the recorded signals. Three different databases of increasing sizes are considered. A convention for Signal-to-Noise Ratio (SNR) estimation is decided. In Chapter 3, the detection problem is addressed and the performance of three selected methods is measured on the database.

Then, the world of pattern recognition is more concretely issued. Chapter 4 is first devoted to the impulsive sound analysis techniques, including traditional time-frequency transformations, speech-typical coefficients and psycho-acoustical features. In Chapter 5, the pattern recognition theories and principles are addressed. Four considered architectures are presented: Bayes mono-gaussian classifiers, gaussian mixture models (GMM), hidden Markov models (HMM), and Neural Networks solutions. For each of them, the performance is measured on the involved databases, and compared.

Noisy environment is the subject of Chapter 6, evaluating the robustness properties of selected classifier architectures and features schemes. A solution is first proposed for white noise backgrounds, and then adapted in the case of real world conditions. Then, Chapter 7 studies diverse solutions for the

difficult task of rejecting sounds that do not belong to the set of considered classes.

In Chapter 8, the behavior of the complete system, associating pulse detection, SNR estimation and recognition, is evaluated. The global performance of the system is addressed, and measures to prevent a potential performance reduction due to the diverse block association are adopted. Some considerations about algorithm complexities and real-time implementations on digital signal processors (DSP) are also given. Finally, Chapter 9 concludes the report.

Main Contributions

One originality of this dissertation stands in the philosophy of the proposed recognition system, relying on a prior detection. However, the main scientific contributions are hold in the innovative detection techniques presented in Chapter 3, as well as in the robust recognition solutions of Chapter 6. The construction of an important database that encloses more than one thousand sounds is to be mentioned too. One goal of this study is to present the problem, in a way that could represent an introduction for people starting in the recognition domain. On another side, the objective is to concretely present both experienced observations and obtained results, allowing the reader to get rapidly used to the behavior of the diverse techniques and to the practical problems that do arise.

1.5 Related publications by the author

Part of the work described in this report has already been subject of the following publications:

[Dufa99] A. Dufaux, L. Besacier, M. Ansorge, F. Pellandini, "Automatic detection and classification of wide-band acoustic signals", in Proceedings of *ASA 99, 137th Meeting of the Acoustical Society of America and Forum Acusticum 99*, Berlin, G, March 14-19, 1999.

[Besa99] L. Besacier, A. Dufaux, M. Ansorge, F. Pellandini, "Automatic sound recognition relying on statistical methods, with application to telesurveillance", in Proceedings of *COST 254, International Workshop on Intelligent Communication Technologies and*

Applications, with Emphasis on Mobile Communication, Neuchâtel, CH, May 5-7, 1999, pp. 116-120.

[Dufa00] A. Dufaux, L. Besacier, M. Ansorge, F. Pellandini, "Automatic sound detection and recognition for noisy environment", in Proceedings of *EUSIPCO 2000, European Signal Processing Conference 2000*, pp. 1033-1036, Tampere, FI, September 5-8, 2000.

The first paper [Dufa99], presented during the 137th Meeting of the Acoustical Society of America and Forum Acusticum 99 in Berlin, introduces the system philosophy, with detection and recognition aspects. Only three classes of impulsive sounds are considered for recognition, comparing both Bayes statistical classifier and Multi-Layer Perceptron (MLP) neural network.

In the second paper [Besa99], published in the proceedings of the *COST 254 International Workshop on Intelligent Communication Technologies and Applications, with Emphasis on Mobile Communication*, more elaborated statistical classifiers (hidden Markov models and gaussian mixture models) are considered and tested on an extended database containing six classes of sounds.

The third publication [Dufa00], at the *European Signal Processing Conference 2000*, places some emphasis on disturbed background environment. Possible solutions to the robustness problem of the recognition algorithms are proposed and evaluated on the 6-class database.

The author was also involved in projects dealing with speech and audio compression methods, as well as with speaker recognition tasks, through the GSM mobile phone network (GSM stands for Group Special Mobile). Those domains are also concerned with subjects connected to sound recognition, such as audio signal analysis, especially human ear modeling, or classification architectures. The related publications are:

[Gras97] S. Grassi, A. Dufaux, M. Ansorge, F. Pellandini, "Efficient Algorithm to Compute LSP Parameters from 10-th order LPC Coefficients", ICASSP'97, IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 3, pp. 1707-1710, Munich, D, April 21-24, 1997.

[Mane99] D. Manetti, A. Dufaux, E. Meurville, M. Ansorge, F. Pellandini, P. Ryser, D. Wieser, "Audio Coder for Telesurveillance

Applications with Real Time Implementation on a Multimedia DSP", Proc. of COST 254, International Workshop on Intelligent Communication Technologies and Applications, with Emphasis on Mobile Communication, pp. 276-279, Neuchâtel, CH, May 5-7, 1999.

[Gras99] S. Grassi, L. Besacier, A. Dufaux, M. Ansorge, F. Pellandini, "Influence of GSM Speech Coding Algorithms on the Performance of Text-Independent Speaker Identification", Proc. of COST 254, International Workshop on Intelligent Communication Technologies and Applications, with Emphasis on Mobile Communication, pp. 307-311, Neuchâtel, CH, May 5-7, 1999.

[Gras00a] S. Grassi, A. Dufaux, L. Besacier, M. Ansorge, F. Pellandini, "Speaker Recognition on Compressed Speech", Proceedings of the International COST 254 Workshop on Friendly Exchanging Through the Net, pp. 117-122, Bordeaux, F, March 23-24, 2000.

[Besa00] L. Besacier, S. Grassi, A. Dufaux, M. Ansorge, F. Pellandini, "GSM Speech Coding and Speaker Recognition", Proc. of the International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2000, Vol. 2, pp. 1085-1088, Istanbul, T, June 5-9, 2000.

[Gras00b] S. Grassi, L. Besacier, A. Dufaux, M. Ansorge, F. Pellandini, "Influence of GSM Speech Coding on the Performance of Text-Independent Speaker Recognition", Proc. of the European Signal Processing Conference 2000, EUSIPCO 2000, Vol. 1, pp. 437-440, Tampere, FI, September 4-8, 2000.

2. Impulsive Sound Database

This chapter presents the different classes of impulsive sounds used in the study, namely: explosions and gunshots, human screams, door slams, glass breaks, dog barks, phone rings, cymbals, and low bangs. Furthermore, other sounds have been recorded for test purposes, like machine noise, pieces of music or children voice sounds.

The way of building the database and the diversities of the constituent signals are discussed, considering some pattern recognition requirements. The main properties of the signals are described, like sampling frequency and signal-to-noise ratios. The recognition experiments involve three sub-databases, which are respectively constituted of 3, 6 and 10 classes. This allows some measure of the influence of an increasing number of sound types on the recognition rates.

Finally, a convention for the measure of signal-to-noise ratios is fixed, to deal with fast time variations of impulsive sound amplitude.

2.1 Database properties

The major part of the impulsive sound samples used in the detection and recognition experiments is taken from different sound libraries available on the market [Leon]. Considering several sound libraries is necessary for building a representative, large, and sufficiently diversified database. The following series of audio CDs are concerned. In the majority of cases, they have been recorded completely digitally ('DDD'):

- BBC Sound FX Library (sets 1 & 2);
- Digiffects (set J);
- Sound Ideas Series 6000 (CD #6004, 6040 & 6050);
- EFX Guns (Vol. I & II);
- The Producers Sound Effects Library (CD DO-01);
- De Wolfe Sound FX (CD #18);

- Bainbridge Living Sound Effects Library (Vol. 8);
- AudioPro Sound Effects (CD #15).

Some particular classes of sounds have been built or completed with hand-recorded signals. Those sounds were digitally recorded with a *Sony TCD-D8* Digital Audio Tape recorder (R-DAT) at a 44100 Hz sampling frequency. A unidirectional microphone *Sony ECM-MS907* was used, featuring a frequency bandwidth ranging from 100 Hz to 15 kHz. All sounds were transferred to PC hard disks with the *Turtle Beach Fiji* digital I/O audio device.

All signals in the database have a 16 bits resolution and are sampled at 44100 Hz. In this way, all possible audio spectrum components can be exploited for recognition purposes. This point is very important for impulsive sounds, whose frequency bandwidth can be rather extended, because of sharp temporal attacks (guns, explosions). Furthermore, some considered sounds show an important energy content in the highest frequencies, as glass breaks for example.

During database construction, great care must be devoted to the selection of the signals, because they can notably influence identification results. In some sense, recognition algorithms try to optimally place boundaries between classes of signals, based on examples of these signals. If bad examples are given to the recognition system at learning time, performance will be poor in further use. As an illustration, if only one signal is very different than the main group of other sounds in a given class, the space domain attributed to this class will be enlarged, allowing more confusion with some neighboring types of sound.

However, when a rather general use of the recognition system is required, some kind of intra-class diversity in the signal properties is important and should be integrated in the database. Even if it would be better for a given recognition system, to be designed for the specific type of encountered signals, it was decided in this study, to incorporate sufficiently diverse signals in the same category. For example, when constructing a set of door slamming sounds, different types of door materials were used, as well as different slamming strengthes. As a result, one class of signals can be composed of very different temporal or spectral characteristics, amplitude levels (consequently signal-to-noise ratios), duration and time location, as illustrated on Figure 2.1 and Figure 2.2. Among all sets of impulsive signals, most of them show a mean duration of one second. Nevertheless, they can sometimes reach 6 seconds, for some explosion or scream samples.

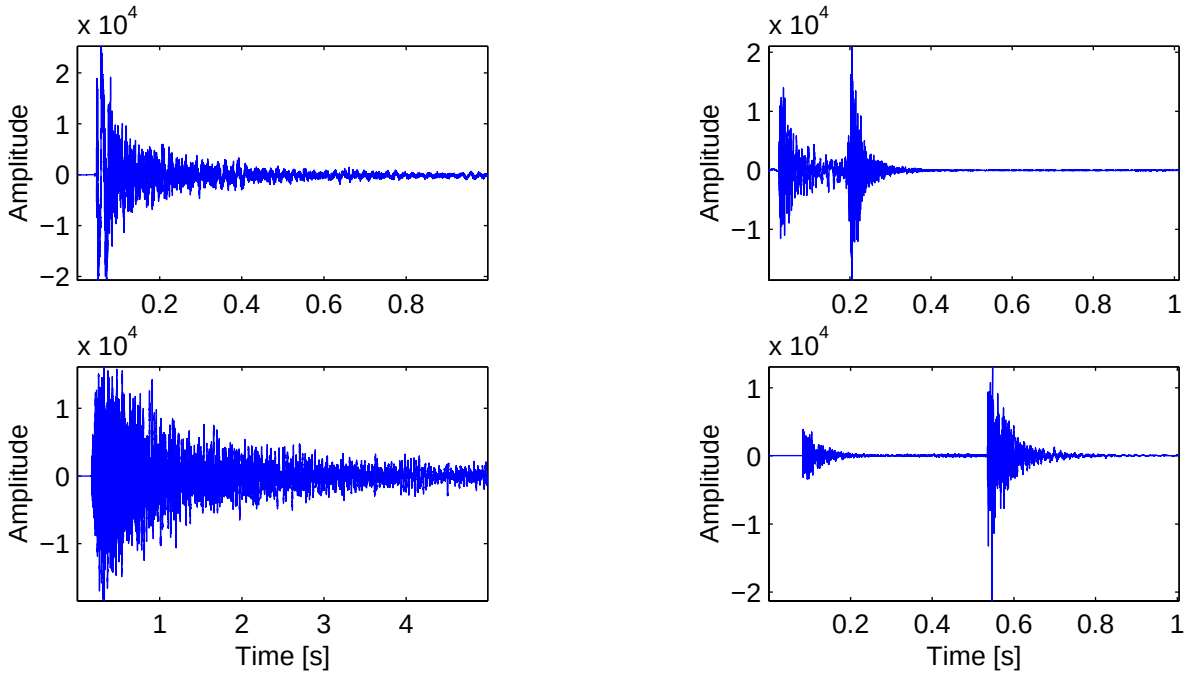


Figure 2.1: *Temporal diversity inside of the same sound class. At left, short and long duration explosions. At right, fast and slow door shuts.*

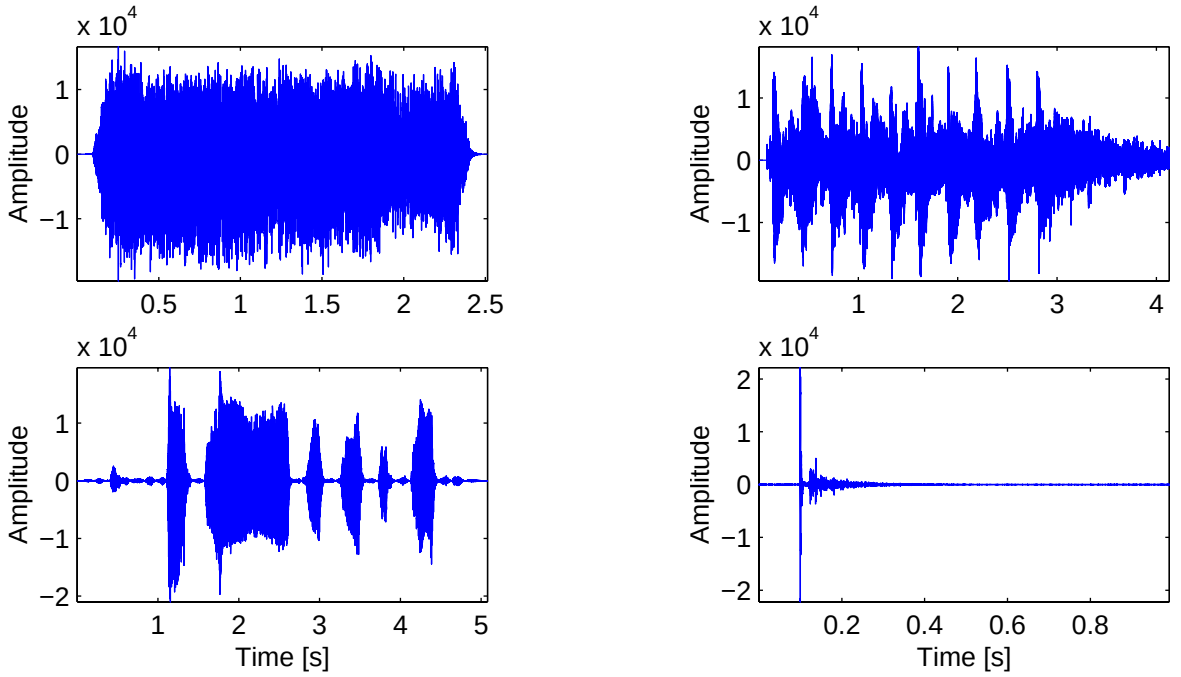


Figure 2.2: *Structural diversity inside of the same sound class. At left, long single and irregular human screams. At right, burst and single gunshots.*

The sound attack generally occurs during the first second. No normalization is applied on 16 bits quantized amplitude values, corresponding to signal-to-noise ratios between 40 and 70 dB (see Section 2.3 for details on signal-to-noise ratio determination).

2.2 Sound classes

The selected impulsive sounds belong to the classes listed below. All categories are typical of surveillance, outdoors or domestic applications. The number of considered samples for each sound category is indicated:

- Door slams 314
- Explosions 62
- Glass breaks 88
- Human screams 73
- Gunshots 225
- Dog barks 55
- Phone rings 51
- Cymbals 52
- Low bangs 81

Furthermore, other non-impulsive classes of sounds are sometimes integrated in the experimentation. They can be used as background noise, on the occasion of robustness evaluation:

- Machines (engine sounds by example) 60
- Music 60
- Children voices 87
- Various real world background noise 41

This database description requires the following comments:

- The number of items in each class is deliberately not equal, and sometimes very different. This allows recognition experiments, simulating non-uniform or unknown probability of sound appearance;
- Explosion and gunshot sounds are very close to each other. Even for a person, it is sometimes not obvious to discriminate between them. They are intentionally differentiated, to test the ability of the system in

separating very close classes of sounds. To some extent, the same remark could hold for cymbals with glass breaks and doors with low bangs;
- Glass break sounds include both bottle and window breaking situations;
- Phone rings are either electronic or mechanic alarms;
- Machine sounds are extracted from the Noisex-92 database [Varg92], and re-sampled at 44100 Hz;
- Music samples are representative of very different music styles;
- Real world background noise is a hand-recorded varied set of sounds like singing birds, cars passing by, people walking and talking, or dog barking in the distance.

All these different signals are grouped to form three sub-databases, used for testing the effect of increasing the number of considered sound categories on the recognition algorithms. These sub-databases are arranged as follows:

- *Little set*: 3 classes containing similar sounds in each class:
  - 69 door slams,
  - 58 explosions,
  - 38 bottle breaks.

- *Medium set*: 6 classes containing sounds with important diversity:
  - 314 door slams,
  - 62 explosions,
  - 88 glass breaks,
  - 73 human screams,
  - 225 gunshots,
  - 60 machine noise.

- *Larger set*: 10 classes containing sounds with important diversity:
  - 314 door slams,
  - 62 explosions,
  - 88 glass breaks,
  - 73 human screams,
  - 225 gunshots,
  - 60 machine noise.
  - 55 dog barks,

- 51 phone rings,
- 81 low bangs,
- 87 children voices.

In particular cases, and for the interest of the experiments, some other combinations will sometimes be mentioned.

2.3 Signal to noise ratios

In this study, the determination of Signal-to-Noise Ratios (SNR) will frequently be referenced to, either on characterization purposes, or as a required pre-processing of the recognition stage, taking part of the class model selection. The purpose of this section is to precisely specify the way of calculating the SNR, to prevent uncertainty about the presented values. The standard definition of the SNR is the following, considering both signal x and noise n individually, during respective time periods L and N:

$$SNR = 10 \cdot \log_{10} \frac{Signal\ Power\ P_x}{Noise\ Power\ P_n} = 10 \cdot \log_{10} \frac{\frac{1}{L}\sum_{i=1}^{L} x_i^{\,2}}{\frac{1}{N}\sum_{i=1}^{N} n_i^{\,2}} \qquad (2.1)$$

It is generally supposed that the noise properties are stationary. Then, the noise power level P_n is constant, and can be measured easily. Conversely, the level of impulsive signals importantly varies during the pulse progress. Therefore, measuring the signal-to-noise ratio of impulsive sounds is inherently difficult and only approximations can be obtained. Ideally, each impulsive sound should be characterized by the evolution of a segmental signal-to-noise ratio, or at least by its mean. Nevertheless, in this work, it is not important to know the exact value of the SNR, because it will only be used in some comparative purpose. Thus, it is sufficient to decide upon a convention to be applied all along this report. For instance, it could be simply declared that the pulse level is the level of its maximum sample. However, this would give a too sensitive and overestimated SNR measure. On the opposite, another solution would be to measure a global value of the signal level, taking the whole pulse duration into account, including its release period till it reaches a pre-defined lower bound. This could be a more consistent way to characterize the signal, as the recognition process effectively uses all the signal evolution to give classification results. In fact,

the problem is the selection of a good value for the parameter L of equation (2.1). Before taking a decision, let's consider the two SNR-related operations that will be considered in this report:

- Sometimes, the objective is to add some noise to a clean impulsive signal, in order to test the system ability to recognize noisy pulses, or to build a noisy class model. In this case, the SNR is specified, and the problem is to set the correct level P_n, for the noise generation. Then, (2.1) can be used directly,

$$10 \cdot \log_{10}(P_n) = 10 \cdot \log_{10}(P_x) - SNR \tag{2.2}$$

performing a power estimation P_x of the clean signal. In this estimation, the choice of L is not critical and any method for measuring the signal level can be selected, as the pulse release is fully available.

- In other circumstances, the objective is to measure the SNR of a detected noisy pulse $xn = x + n$. This problem is more complex, because the incoming signal is the superposition of both noise and clean signal. However, if the noise power P_n is known (from past samples) and assumed quasi-ergodic, the SNR can be approximated, starting from an estimation of the incoming signal power P_{xn}:

$$SNR \cong 10 \cdot \log_{10} \frac{P_{xn} - P_n}{P_n} \tag{2.3}$$

Then, the point is to decide what time boundaries to consider for measuring P_{xn}? The shorter the time zone (starting from the pulse attack), the higher the SNR! Considering approximation (2.3), one can observe that it gets more and more correct when $P_{xn} \gg P_n$ (because if so $P_{xn} - P_n \to P_x$), that is when we consider a shorter zone. Ideally, the answer to that question could be given in an adaptive manner by the detection operation that should be able to locate good boundaries, corresponding to the time zone where the pulse is dominant. Of course, those boundaries will vary with the noise level: the more important the noise level, the more restricted the observable pulse release.

In fact, the choice of L has to be done according to the following constraint: both SNR measures (2.1) and (2.3) should match as much as possible. In other words: $P_{xn} - P_n \to P_x$. Therefore, to encompass the noisiest situations, where P_{xn} is to be measured over a brief time window, a sufficiently short duration after the pulse attack should be considered for measuring the pulse level P_x. For that reason, for all pulse level evaluations, it was decided, as far as possible, to consider a time window starting at the beginning of the pulse

attack, and ending one quarter of a second later. Most encountered pulses have a sufficient duration, to still possess significant components at that time:

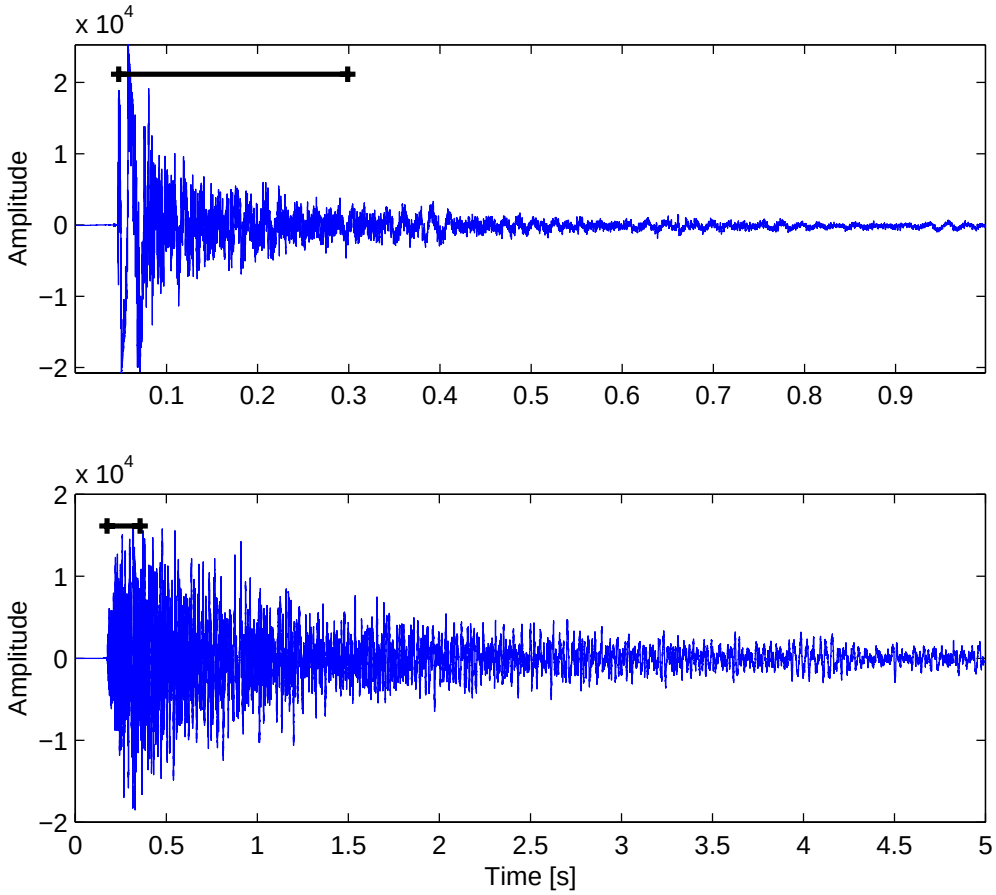


Figure 2.3: *Position of signal level measurement for SNR estimation. Two examples of explosions.*

More concretely, supposing that the detection stage gives a flag at the moment of the pulse start, a measure of the noise level P_n can be performed using the previous second of signal, an estimation of P_{xn} can be done during the next quarter of second, and the SNR can finally be determined using (2.3).

3. Impulsive Sound Detection

This chapter introduces the pre-processing techniques needed to detect potentially alerting signals, to be further analyzed in the recognition stage.

Three simple and efficient time domain pulse detection schemes are proposed and studied. All methods operate by measuring the power evolution of the input audio signal. A threshold can then be selected, to take detection decisions in case of significant power variations. Techniques like non-linear median filtering or running standard deviation estimation are studied. They can be used, either to transform the power sequence in a normalized domain, where its pulses are emphasized and more easily detectable, or to track the background noise level and adapt some detection threshold.

Those techniques provide rather good performance under important and variable background noise degradations. Beside the automatic adaptation of the detection threshold to the noise level and variance, the proposed methods feature tunable detection sensitivities, for specific application needs.

3.1 Detection Constraints and Requirements

According to the system philosophy described in Section 1.1 (Figure 1.1), the detection methods described in this chapter should be designed to represent a low computational load, since they are intended to be running all the time. Fortunately, to relax those constraints, the time resolution of consecutive possible detection instants can be rather coarse, at first sight. If needed, a more precise positioning of the pulse attack can be performed in a second time. This last point will be considered in Chapter 8, with respect to the behavior of the recognition algorithms.

Secondly, the *sensitivity* of the detection algorithm is considered in the following lines, as the ability of detecting irregularities whose level is more or

less close to the background noise. Of course, the sensitivity parameter is tied to the well-known error compromise between the detection of non-existing event (false detection) and the non-detection of a real event (missed detection):

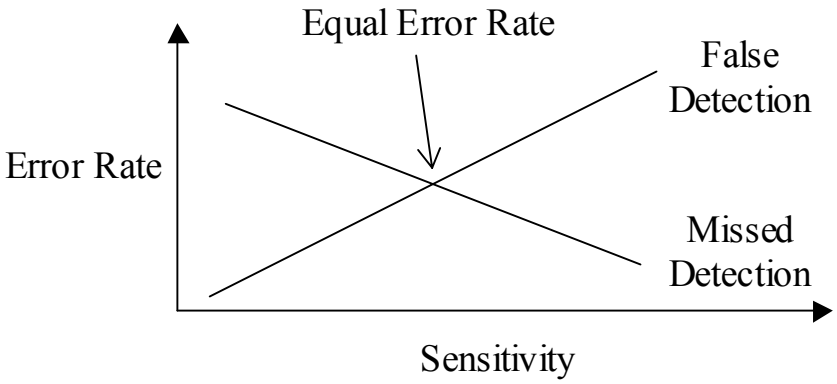


Figure 3.1: *Detection error trade-off.*

Depending on the application, the sensitivity may have to be high, ignoring frequent false detections, or low, to minimize this risk. The choice of the sensitivity must be considered with regard to the classifier ability to correctly reject signals not belonging to any of the interesting sound classes. At this step and for that reason, the detection techniques will be designed to allow a manually tunable sensitivity.

It must be noted that false and missed detections are not the only kinds of encountered errors that can affect the later recognition process. In some situation, in presence of an important background noise, the localized instant of detection can be rather distant from the real beginning of the pulse, without possibility of refining it. In this case, the so-called *bad detection* situation has to be considered, if the imprecision exceeds some time duration. In this work, one second is considered for this duration.

Finally, another important requirement for detection methods is their adaptability to either sudden or slow variations of the background noise level and variance. To take some examples, if a machine is being turned on, the step in noise energy should be signaled, and then, the detection threshold should be quickly updated to be able to notice some possible scream arising just a few seconds later. In other situations, a good detector could indicate the beginning of a slowly increasing sound from a hypothetical-approaching car, and then, be able to adapt its threshold to the sound level of the car passing

by. This would allow the detection of a possible explosion happening during the passage of the vehicle.

In the following section, a first very simple detection method, based on the standard deviation of normalized power sequences, will be presented, describing each step, discussing the choice of the involved parameters, and characterizing the obtained results. Then, more advanced techniques involving non-linear median filters are investigated, and two other solutions are proposed. In the last section, the questions of real-time processing and complexity of the detection schemes are addressed.

3.2 Detection Using Normalized Power Sequences

The first method presented in this chapter is conceptually very simple, and can be described by the following steps:

- Estimation of the signal power, for each consecutive non-overlapping block of audio samples;
- Windowing of the obtained power sequence to consider only its more recent elements;
- Normalization of the windowed power sequence ;
- Determination of the variance of the resulting normalized sequence;
- Application of a threshold on the variance for making the detection decision;

For each step, the choice of the methods and parameters are discussed in the corresponding following sub-sections.

3.2.1 Power Estimation

All the detection process is based on the power evolution of the audio signal. Considering the digitized input audio signal *x(n)*, an estimation of its power is done for each successive block *k* consisting of *N* samples, yielding to the following power sequence *e(k)*:

$$e(k) = \frac{1}{N}\sum_{n=0}^{N-1} x^2(n+kN) \qquad k = 0,1,.... \tag{3.1}$$

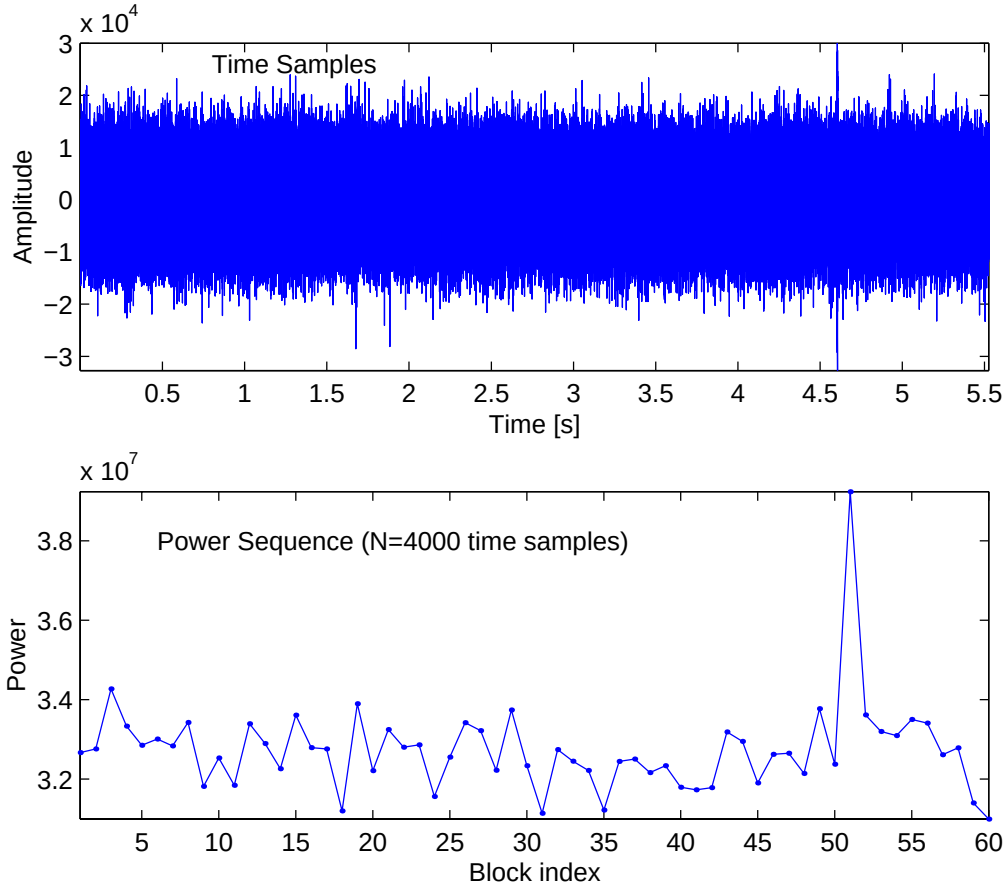


Figure 3.2: *Amplitude and power sequence (N = 4000) in case of a short gunshot signal (amplitude = 30000) arising at time 4.6 seconds, disrupted by gaussian white noise (standard deviation = 8000). SNR=0.5 dB.*

The choice of the block length N must be done carefully, considering temporal properties of the impulsive sounds, and in such a way to facilitate the detection at further stages of the process. Therefore, choosing a precise time resolution is not good, because too much detail would needlessly appear in the power sequence, occasionally disturbing the detection process. On the other hand, if N is too large, some difficulties could arise while detecting fine pulses, especially in case of important background noise. Considering the computational load, the larger the blocksize, the lower the detection complexity, except for a possible time location refinement. Thus, some compromise is needed, and a good solution was found using N = 4000 samples, which corresponds approximately to 90 ms. Figure 3.2 shows a rather difficult situation, with a very short gunshot sound, standing half a decibel above the level of the disturbing gaussian white noise.

3.2.2 Windowing and Normalization of the Power Sequence

At every new block k, a new power value $e(k)$ is obtained, and the power sequence is processed through a rectangular window, to retain only the L more recent values (indexed $j = 0,...,L-1$):

$$e_{win}(j,k) = e(i), \quad i = k - L + 1,...,k \tag{3.2}$$

In order to keep a significant enough portion of signal, L is chosen to be 30, representing approximately 3 seconds. Then, those 30 values are normalized, transforming the dynamic of the windowed sequence to the range [0, 1]. Considering the k^{th} window, and removing the index k for reading simplicity, the resulting normalized window becomes:

$$e_{norm}(j) = \frac{e_{win}(j) - \min_j(e_{win}(j))}{\max_j\left(e_{win}(j) - \min_j(e_{win}(j))\right)} \qquad j = 0,...,L-1 \tag{3.3}$$

The normalization is actually the key of the detection method. When the signal is rather stable, with a constant or slightly variable level, all values in e_{norm} are spread between 0 and 1, with a rather important standard deviation, near 0.3. On the opposite, once a significant pulse occurs, the last value of the window reaches 1, while the previous ones are suddenly clustered near to 0. As a consequence (see Figure 3.3), the standard deviation falls drastically below 0.1, the more abrupt the increase in energy, the lower the standard deviation. To some extent, this behavior ensures the insensitivity to a possible positive or negative tilt of the background noise level (see Figure 3.4), because of the low energy difference between $e(k)$ and $e(k+1)$.

3.2.3 Variance Thresholding

This normalization operation turns the standard deviation of the windowed sequence into a very efficient criterion for detecting a sudden increase in the signal power level. A thresholding operation can trigger a detection flag when the standard deviation gets below a given value. The variance, which is the square of the standard deviation, can be identically used, saving the square root evaluation:

$$var(k) = \frac{1}{L-1} \sum_{j=0}^{L-2} \left[e_{norm}(j,k) - \overline{e}_{norm}(k) \right]^2 \tag{3.4}$$

Practically, the last value of the window, that reaches 1 when the energy increases significantly, is not included in the determination of the variance $var(k)$. This gives rise to a lower variance value when a pulse is present, making the detection easier. Consistently, $\overline{e}_{norm}(k)$ represents the mean of the first L-1 normalized value in the window.

A few examples of the detection system behavior are provided in Figure 3.3 and Figure 3.4, using 0.15 as the standard deviation threshold. Figure 3.3 illustrates the detection process applied on the signal shown in Figure 3.2, supposing that the pulse present at time 4.6 s. occurs during block index k.

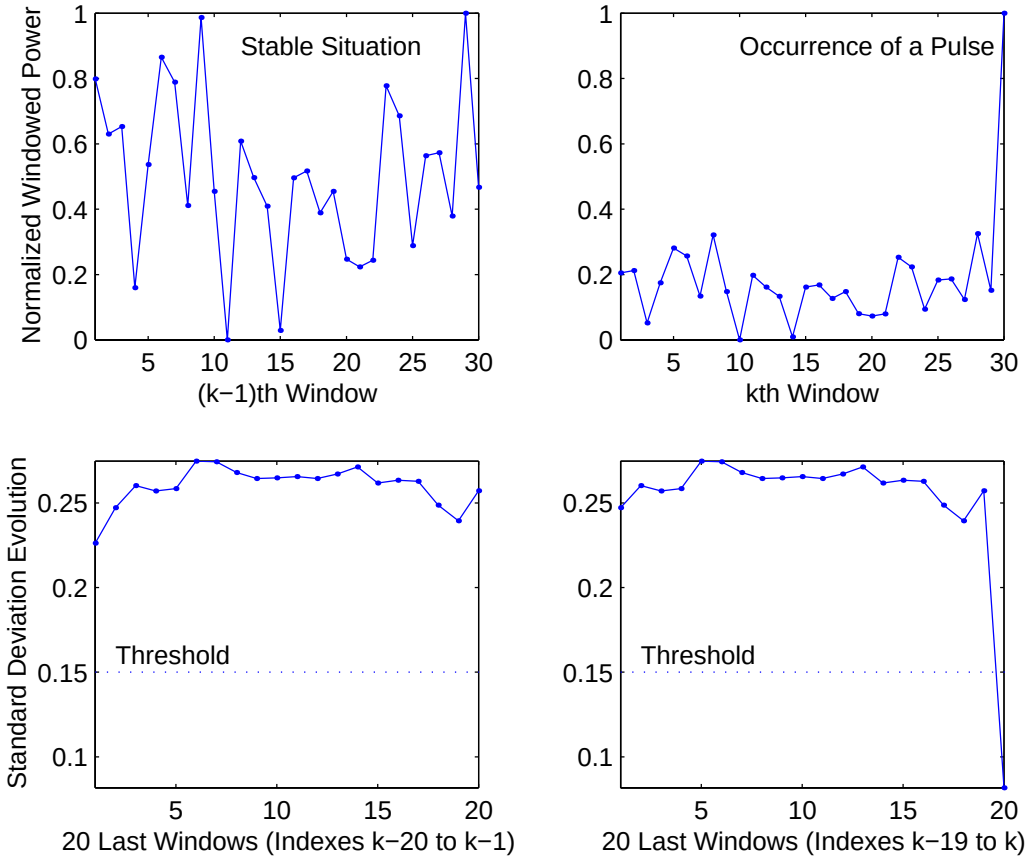


Figure 3.3: *Illustration of the detection process. Normalized power windows $e_{norm}(j,k)$ (on top) and evolution of the standard deviation var(k) over the 20 previous normalized windows (below). Two typical situations are represented: just before (at left) and just at the time (at right) when a pulse occurs: the standard deviation obviously falls below the threshold (shown in dotted lines), even in this rather difficult situation, where the pulse level is just 0.5 dB above the background white noise level.*

Figure 3.4 represents situations of both increasing and decreasing level of background noise.

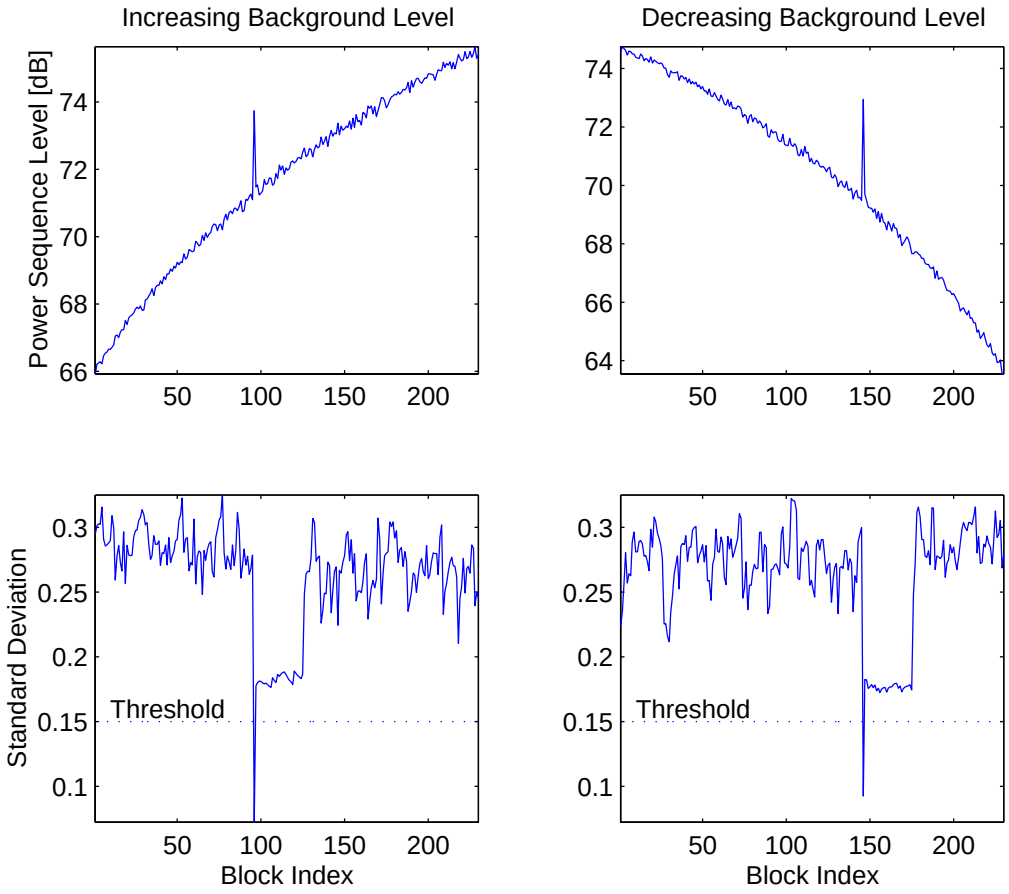


Figure 3.4: *Power sequence and standard deviation for increasing (at left) and decreasing (at right) background gaussian white noise level. The threshold is shown in dotted lines.*

Considering this figure, one can observe that the variance sharply decreases during the time the pulse is present in the window. This effect is particularly strong when the pulse is very short. As a consequence, the window length L has to be carefully chosen. It has to be high enough to ensure reliable values of the standard deviation. On the other side, the window must not be too long, because in some cases, a positive detection flag may still be produced as long as the pulse is present in the window. This means that during the L-1 blocks following the pulse attack, the system could not be able to signal the possible presence of a new critical event. This point is not a problem and can be encompassed in the classifier stage, as far as the window is short enough. As already mentioned, L=30 is a good compromise.

3.2.4 Detection of Slowly Varying Sounds

In every successive window *k*, studying the evolution of the last value $e_{norm}(L-1)$ can provide interesting information about the tilt of the sound level. As shown in Figure 3.5, a mean near to 1 expresses an increasing energy level. On the opposite, if the mean is near to 0, the level must be decreasing. In stable condition, the values are spread between both limits, and the mean is around 0.5. This property could be used for detecting different kind of slowly varying sounds, like vehicles coming and passing by for example. A threshold could be applied, detecting an increasing sound level when $e_{norm}(L-1)$ is systematically above 0.5 over a recent window, or detecting a decreasing level if those recent values are always below 0.5. In another way, the threshold can be affected to the mean value over the considered window.

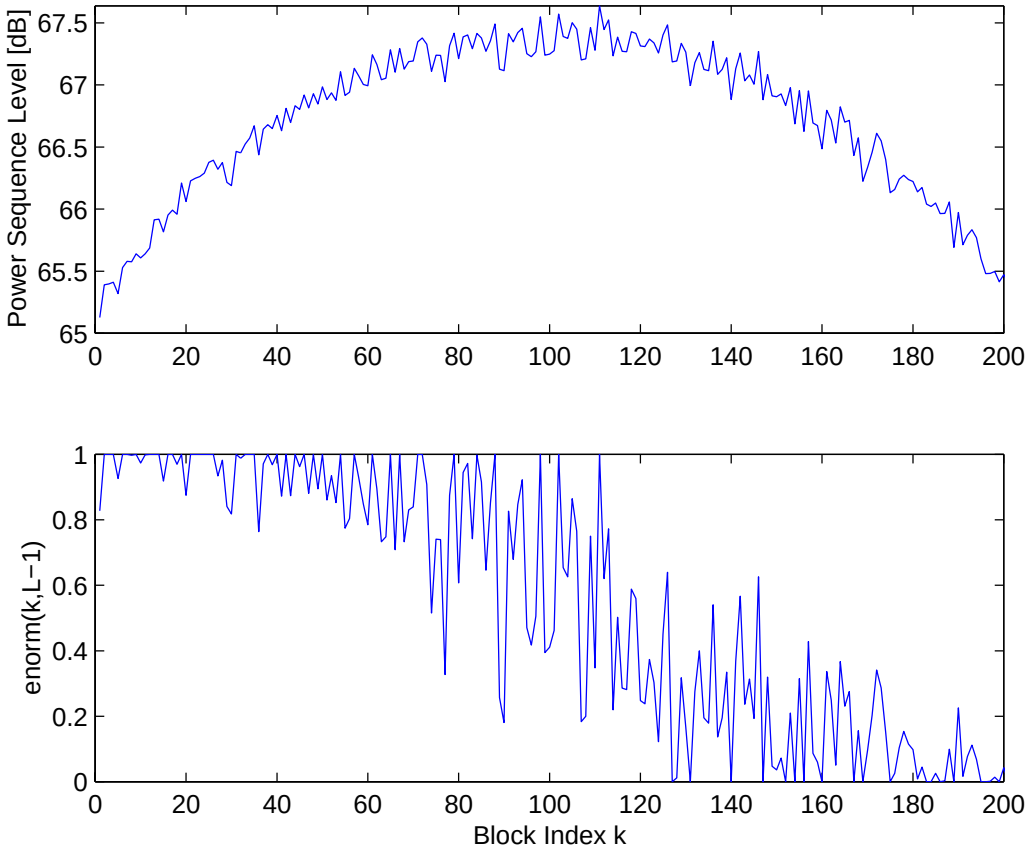


Figure 3.5: *Power sequence e(k) and evolution of the corresponding* $e_{norm}(L-1, k)$ *values, in the case of an increasing and then decreasing sound.*

3.2.5 Threshold Optimization

This detection method enables a tuning of the sensitivity, applying some slight deviations on the threshold value. Ideally, the threshold should be selected to optimize the false/missed detection compromise. However, Figure 3.6 shows that the optimal threshold changes with the signal-to-noise ratio.

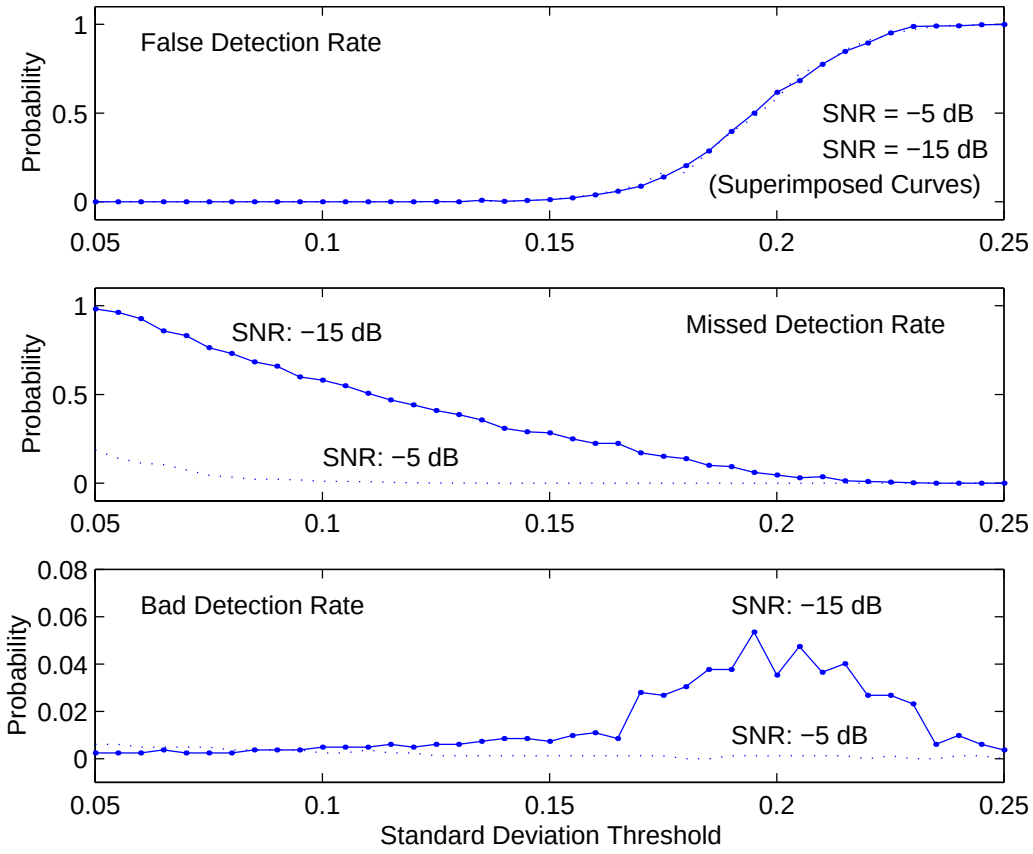


Figure 3.6: *Variation of false, missed, and bad detection rates, for threshold values in the range 0.05 to 0.25. The measures involve the medium database, and two different gaussian white background noise levels are considered: -5 and –15 dB. The random white noise is re-generated for each pulse.*

In fact, if a pulse is present, the variance decrease is less important in case of high noise level. As a consequence, the probability of missing a pulse increases when the SNR gets lower, what is not surprising. On the opposite, the false detection rate is not affected by low SNR. Only the temporal precision of detection is concerned with high level of background noise. After some experiments involving the medium database, a globally satisfying value

for the standard deviation threshold was selected to be around 0.15. This ensures the false detection rate to be very low, and minimize the missed detection risk when the SNR decreases. Otherwise, the threshold could be made adaptive to the input noise level, being lower (below 0.1) for clean signals, and reaching 0.18 in very noisy situations.

3.2.6 Performance

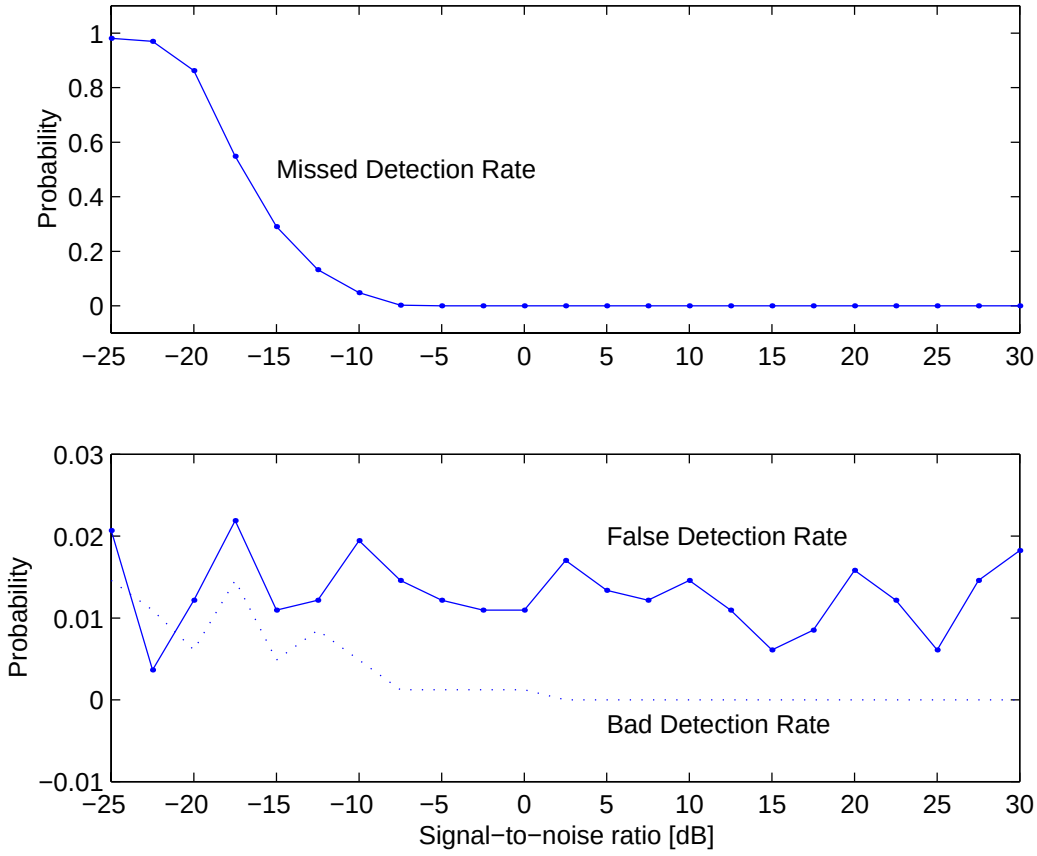


Figure 3.7: *Optimized performance of the detection method: missed, false, and bad detection rates, for a standard deviation threshold of 0.15, for gaussian white background noise. Measured using the medium database.*

The performance of the detection scheme described in this section can be illustrated in Figure 3.7. The measures are done using each signal of the medium database, adding a randomly generated gaussian white noise to it. The noise is different for each pulse. A missed detection situation is obtained when no pulse is signaled. If the pulse is signaled, the position of the flag is

verified, and if an error larger than 1 second is found, a bad detection situation is declared. For false detection rates, the pulse is not added to the background noise, and if some detection flag is signaled, the false detection situation is encountered.

It can be observed from Figure 3.7, that the missed detection rate becomes critical for SNR below –15 dB, while the false detection rate always remains below 2.2%. Bad detection rates are generally equal to zero, except under 0 dB. They can reach 1.5% at –20 dB. Those results are satisfying, considering the target application, that is the detection pre-processing in a sound classification system. In effect, recognition algorithms start to have severe difficulties for SNR values that get below -10 dB. Consequently, a more robust detection method would not be more helpful, at system level.

In conclusion, this simple method presents good detection performance, with the capacity of immediate reaction to the power variations. Thus, the maximum overall delay is given by the block length N (~100 milliseconds). One possible drawback could be the incapacity to detect successive critical events. However, this can be managed at system level, in the design of the recognition stage, by adapting the considered duration of signals to classify.

3.3 Median Filtering Techniques

3.3.1 Principles

A median filter is a non-linear operator, often used in noise removal applications when traditional linear filtering is not appropriate [Embr91]. In situations involving impulsive noise annoyance, a traditional linear filtering can degrade the signal of interest, while a median filter process offers the possibility to remove pulses without touching sharp attacks of the signal, as shown on Figure 3.8.

In its causal version, given an input sequence of numerical values *e(k)*, the median filtering process at the k^{th} sample consists in sorting it together with the L-1 past values (increasing order), to provide the median (or central) value *mf(k)* as filter output:

$$mf(k) = MED_{i=k-L+1}^{k} \; e(i) \tag{3.5}$$

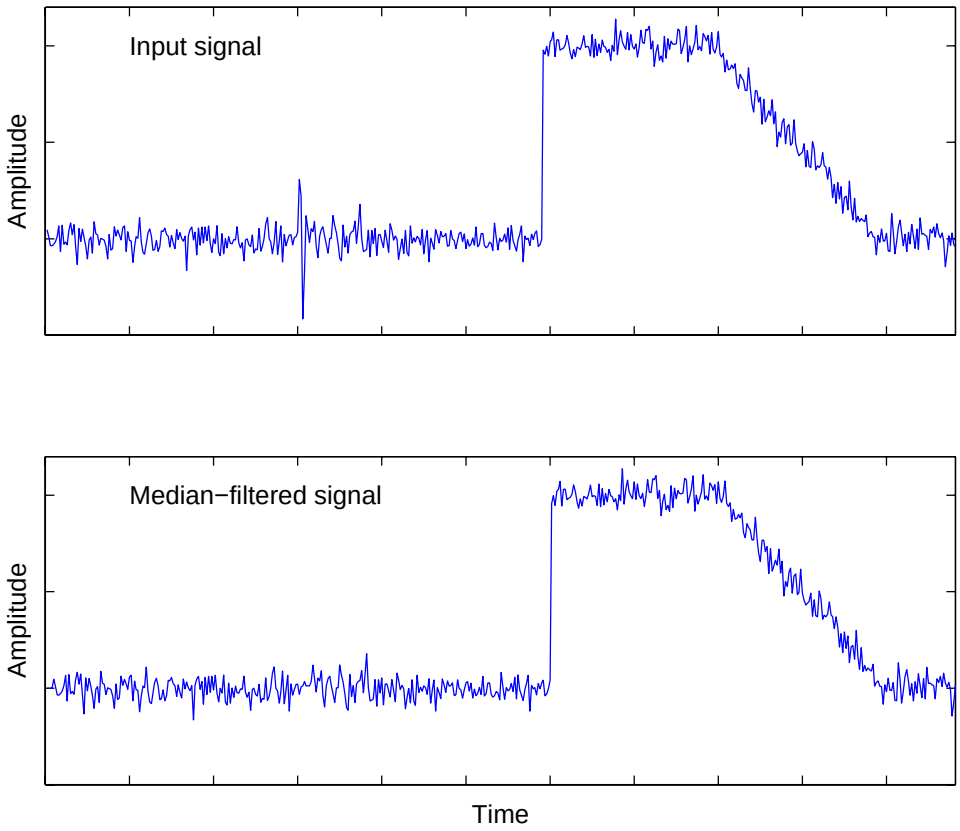


Figure 3.8: *Illustration of a median filter process: A disturbing spike is removed, without affecting the sharp attack of the signal.*

In the above expression, *MED* is the median operator, which sorts the L samples, and gives back the middle value. If $L = 9$, for instance, the output of the filter will be the fifth sample. The parameter L represents the median filter tap number, giving rise to a delay d:

$$d = (L-1)/2 \tag{3.6}$$

A convention must be fixed when working with even values of L. In this study, to minimize both complexity and delay, the highest ranked value among the two central positions would be selected. Then, d becomes $L/2$-1.

In most situations, the median operator has the interesting property of totally removing spikes, whose duration is lower than $L/2$. It is therefore

important to determine a suitable value for this parameter L, depending on the type of pulse to remove, and possibly taking into account some delay constraints.

Sometimes, it can also be useful to affect only the pulses whose amplitude is above a given value. In this case, a *conditional median filtering* is used, and a threshold value *th* is introduced as second parameter. In this case, after sorting the L samples, the absolute difference between *mf(k)* and the corresponding input value *e(k-d)* is compared with *th*. If the difference is larger than the threshold, *mf(k)* is provided as the conditional median filter output *cmf(k)*. Otherwise, the input value *e(k-d)* is kept unchanged:

$$cmf(k) = \begin{cases} mf(k) & if \ \left| mf(k) - e(k-d) \right| > th \\ e(k-d) & otherwise \end{cases} \tag{3.7}$$

This conditional approach was used in the example of Figure 3.8, to keep unchanged the stable noise component. The causal implementation of this conditional median filter is illustrated on Figure 3.9.

3.3.2 Application to pulse detection

The median filter technique can easily remove pulses whose duration is lower than half the order and whose amplitude exceeds the mean background noise level by more than the median threshold. If the power sequence $e(k)$ of section 3.2.1 is considered as the input of the filter, its pulses can be removed, provided that good values are selected for L and *th*. Then, as proposed in [Rice98], subtracting the filter output from its delayed input

$$p(k) = e(k-d) - cmf(k) \tag{3.8}$$

gives rise to a new sequence $p(k)$, preserving the pulses of interest only. In this way, a detection process can be applied to the sequence $p(k)$, searching for remaining pulses and setting the detection flag on, each time $p(k)$ is not zero.

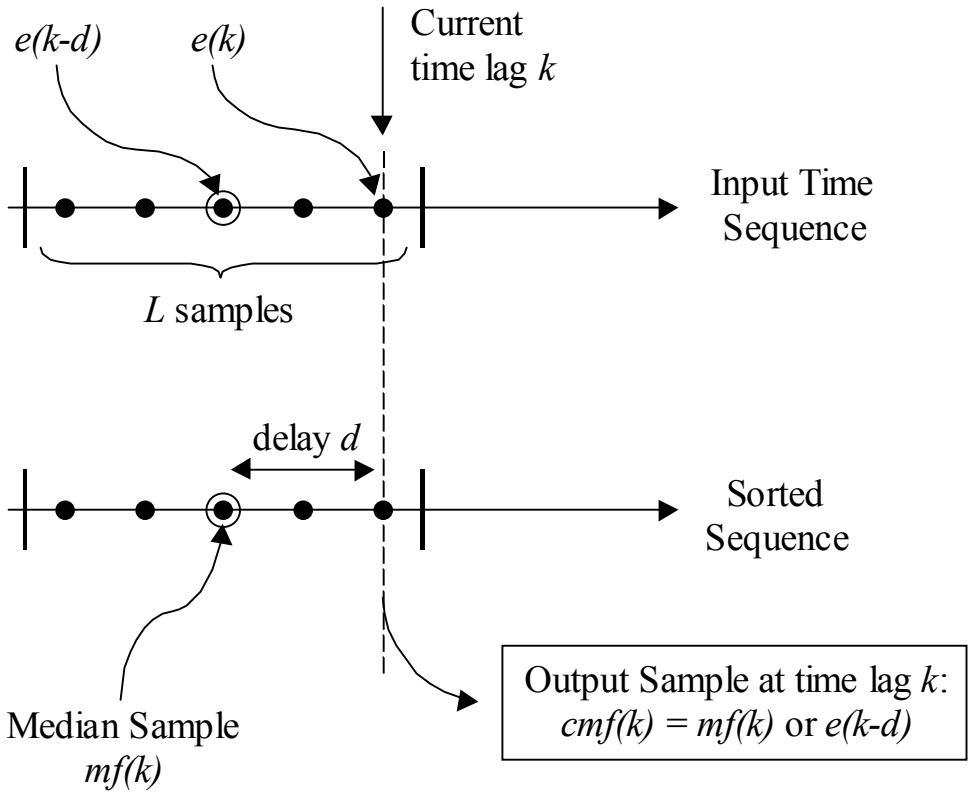


Figure 3.9: *Conditional median filter: Illustration of a causal implementation, with L=5.*

To help understanding this principle, Figure 3.10 shows the different signals involved in the median filter detection for the gunshot pulse of Figure 3.2, which stands 0.5 dB over the background level. One can observe the resulting delay d on the filtered sequences $mf(k)$, $cmf(k)$ and $p(k)$. The sorted sequence $mf(k)$ is rather flat and can be considered as a good way to estimate the mean background level. Otherwise, $cmf(k)$ represents the background noise, and can be used to extract some statistical properties. Finally, it must be clear that $p(k)$ is identically zero, except during the pulse detection. Thus, the only condition for signaling a critical event is that $p(k)$ is not equal to zero. The value of $p(k)$ gives an absolute indication about the importance of the pulse.

This method is considered as one of the most efficient pulse detection technique, and is widely used, especially when both the background noise and the potential pulses possess well-known and stable characteristics. In more general cases, when the duration of the pulses or the background level may

vary, it is more difficult to find optimized fixed values for the filter tap number L and the threshold th. Let's study what happens in those situations.

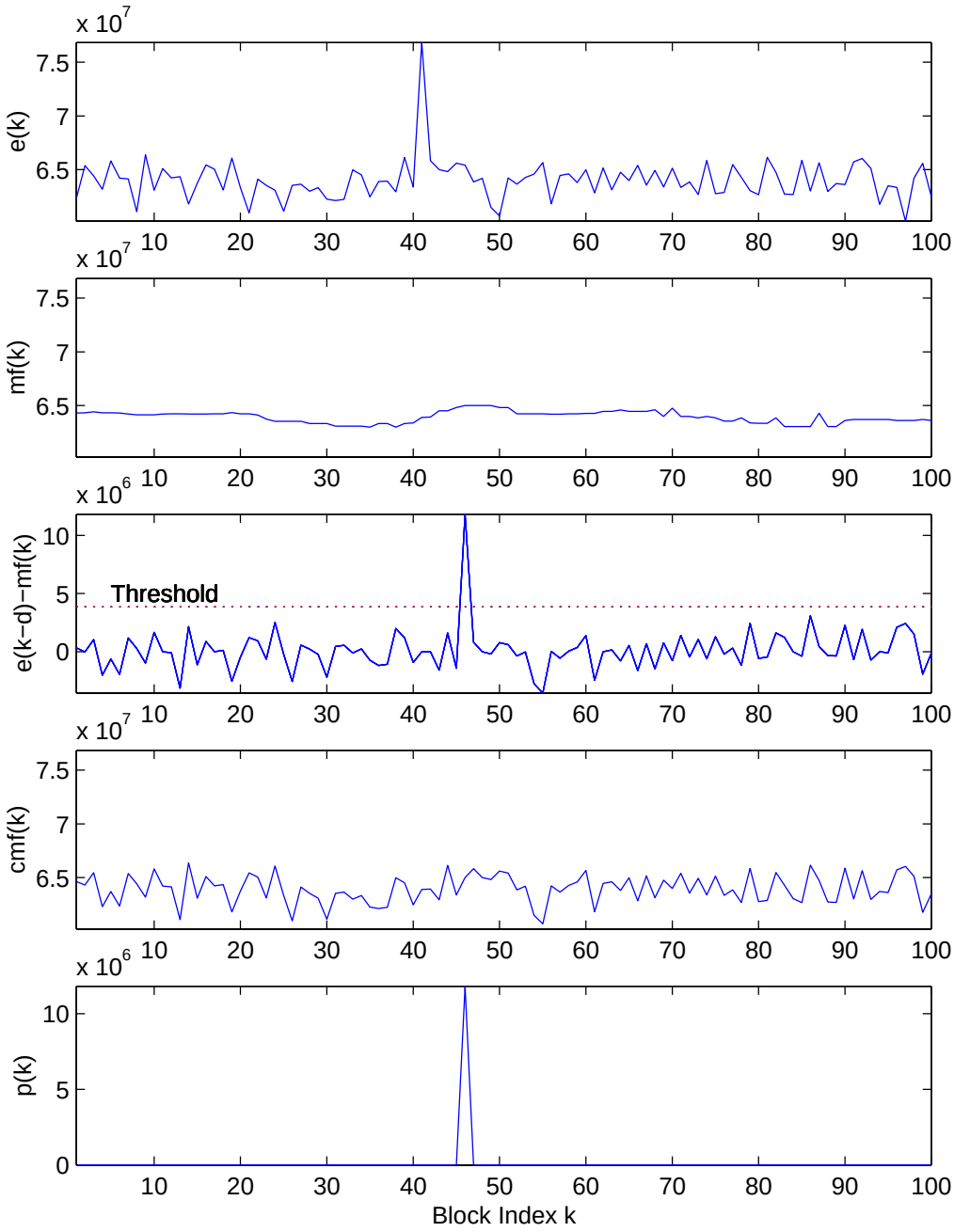


Figure 3.10: *Illustration of a detection based on the median filter technique, for the signal in Figure 3.2 (SNR=0.5 dB), with L = 11 and th = 3.86 10^6. The threshold is shown in dotted lines.*

3.3.2.1 Variation of pulse duration

When the pulse duration gets over $L/2$, part of the pulse appears in $mf(k)$, which increases significantly instead of being flat. Nevertheless, this situation is not too much disturbing, because the difference between the input pulse and $mf(k)$ is generally large enough, to let sufficient components appear above the threshold *th* (see Equation (3.7)). Thus, the final output $cmf(k)$ is still different from the input pulse, while not perfectly flat. It follows that $p(k)$ is not zero, and the pulse can be detected all the same. The only drawback is that the sensitivity of the method may be reduced, having consequences when the pulse level is just above the background noise.

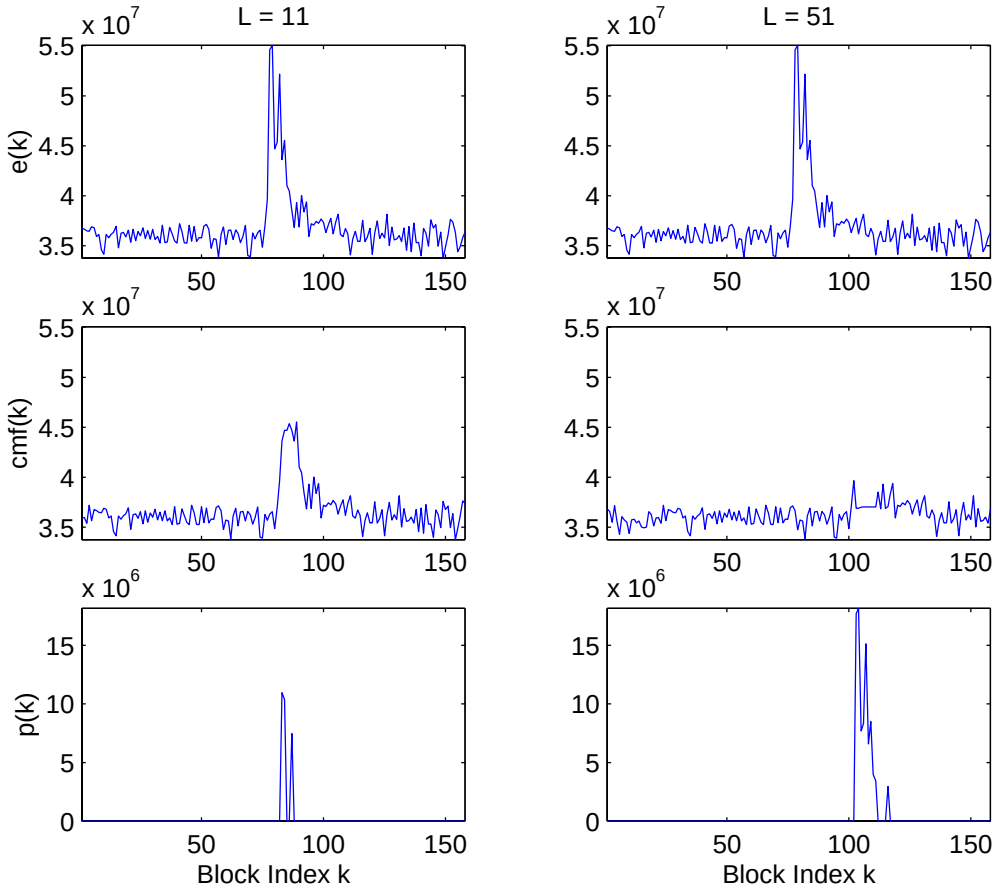


Figure 3.11: *Application of two different tap numbers L, for filtering the same power sequence (SNR=2dB). On the left, L is not high enough, cmf contains some pulse components, and the event is only partially detected. On the right, L has been raised, cmf is flat, and the detection is clearer. In extreme cases, when the SNR is lower, the pulse on the left case would not be detected anymore.*

Figure 3.11 illustrates this situation, showing the detection impairment when L is not large enough. Increasing L can not always be a satisfying solution, because the delay is linearly rising with L. For instance, if the detection flag of an event is required to be half a second after its starting time, the corresponding value of L cannot be higher than one second, that is L=11 according to section 3.2.1. Considering that an explosion or a human scream can be 5 or 6 seconds long, the compromise is inevitable.

3.3.2.2 Variation of background level

When the background noise level changes, the threshold *th* can be adapted, using recent values of the noise power level. One solution to keep the threshold up-to-date is to apply a second median filter on the input sequence, this time using a long-term tap number L_a corresponding to several seconds. In this case, the corresponding sequence $mf_a(k)$ estimates the mean value of the input, being sure not to take any pulse component into account. Furthermore, if the threshold th_a is selected equal to the previous value of *th*, $th(k\text{-}1)$, the difference between $cmf_a(k)$ and $mf_a(k)$ can be used to measure the background noise variations (see Figure 3.12). In this way, the new threshold *th(k)* is recursively adapted, being set just above the maximum values of the background noise. For example, the threshold *th* can be chosen 50 percents over the maximum value in a recent window, that is:

$$th(k) = ov \cdot \max_{i=k-D+1}^{k} \left\{ cmf_a(i) - mf_a(i) \right\} \qquad \text{with } ov = 1.5 \tag{3.9}$$

The *ov* factor can be tuned and is inversely proportional to the sensitivity of the detection scheme. The larger the concerned window length D, the slower the reaction to background noise modifications.

Figure 3.12 illustrates the threshold adaptation, for an increasing background situation on the left, and a decreasing situation on the right. Then, Figure 3.13 illustrates the application of the obtained threshold for the detection of potential pulses in the same power sequence. It can be seen that the presence of the pulse doesn't affect the threshold determination. However, this could happen in case of a long pulse. Typically, the measure of the threshold must at least involve 5 seconds of past signal. Once again, L_a is a compromise, because if it is too large, the adaptation will be importantly delayed. In fact, L_a sets the limit between what is considered as a pulse (in which case, the threshold must not be affected) and what is considered as a temporary (but long enough) increase of the background noise. In this situation, the threshold is adapted.

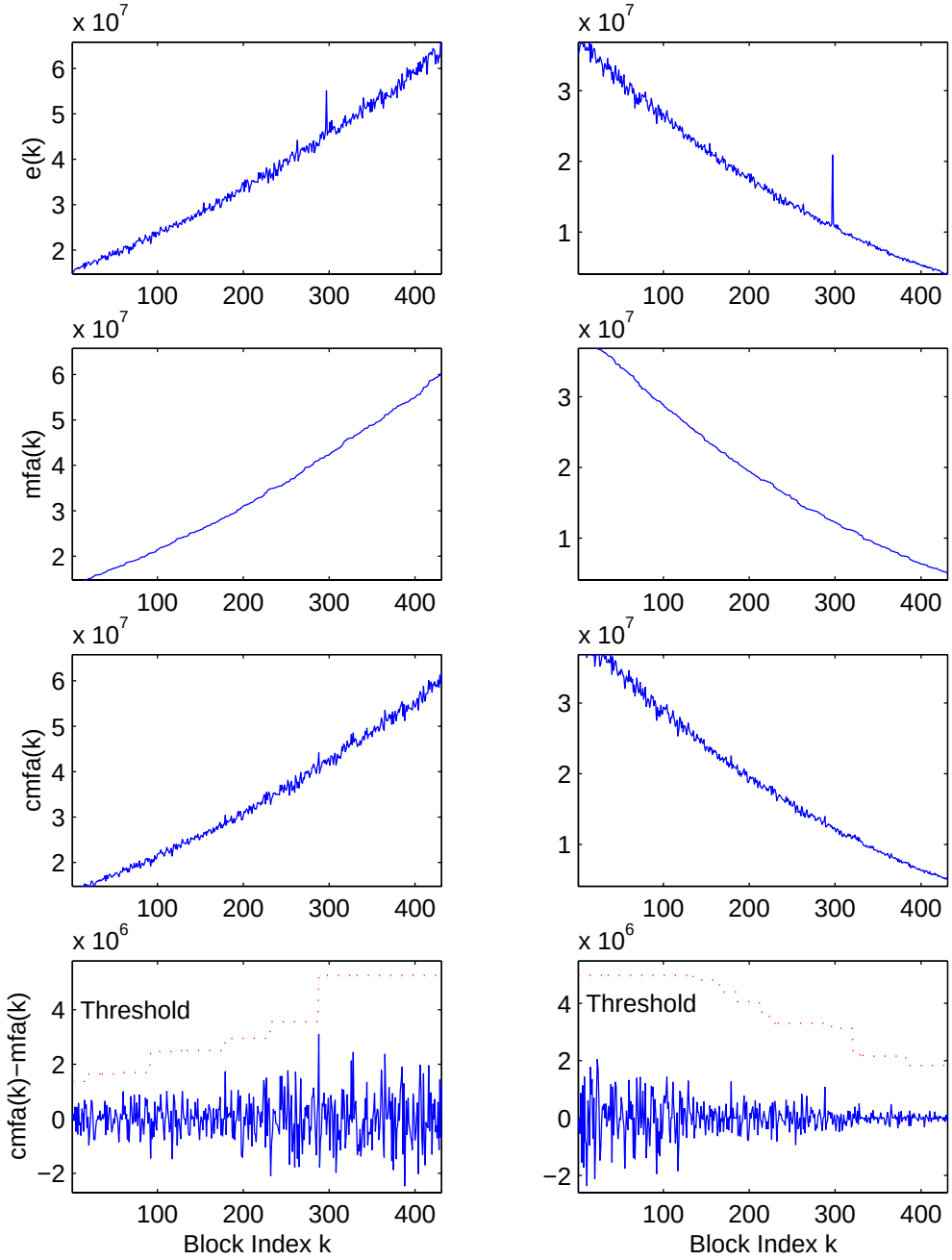


Figure 3.12: *Threshold (dotted lines) estimation using a long-term median filtering of the recent power sequence. An increasing background situation is shown on the left (SNR=1dB), and a decreasing situation on the right (SNR=3dB). A pulse is present near to the end of the sequence. The filter tap number L_a is 51, and the threshold is chosen to be 70% above the maximum value of the background noise in a recent window of duration D = 204 blocks.*

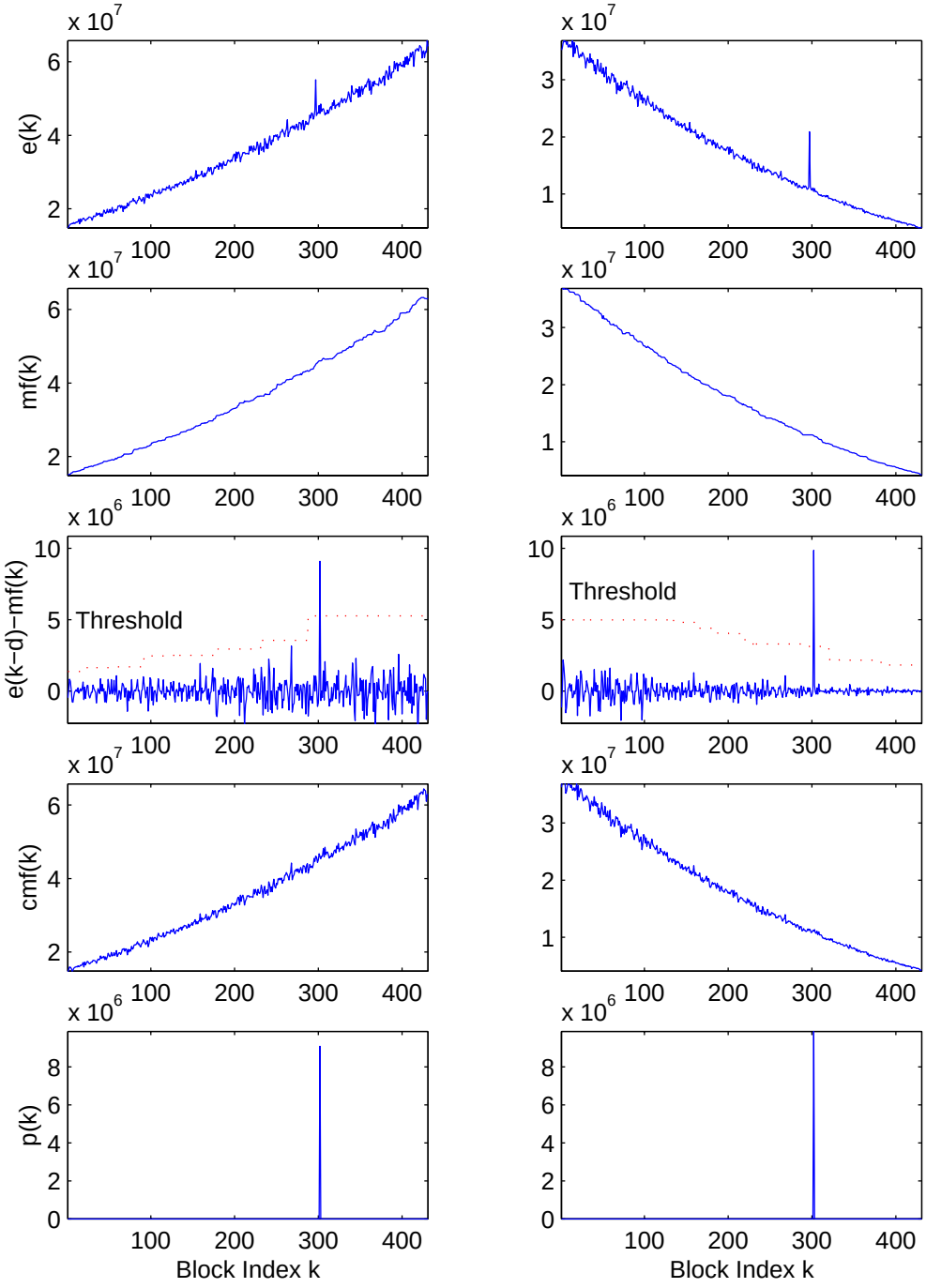


Figure 3.13: *Application of the detection thresholds (dotted lines) determined in Figure 3.12, to the same power sequences e(k), using the median filter with L=11.*

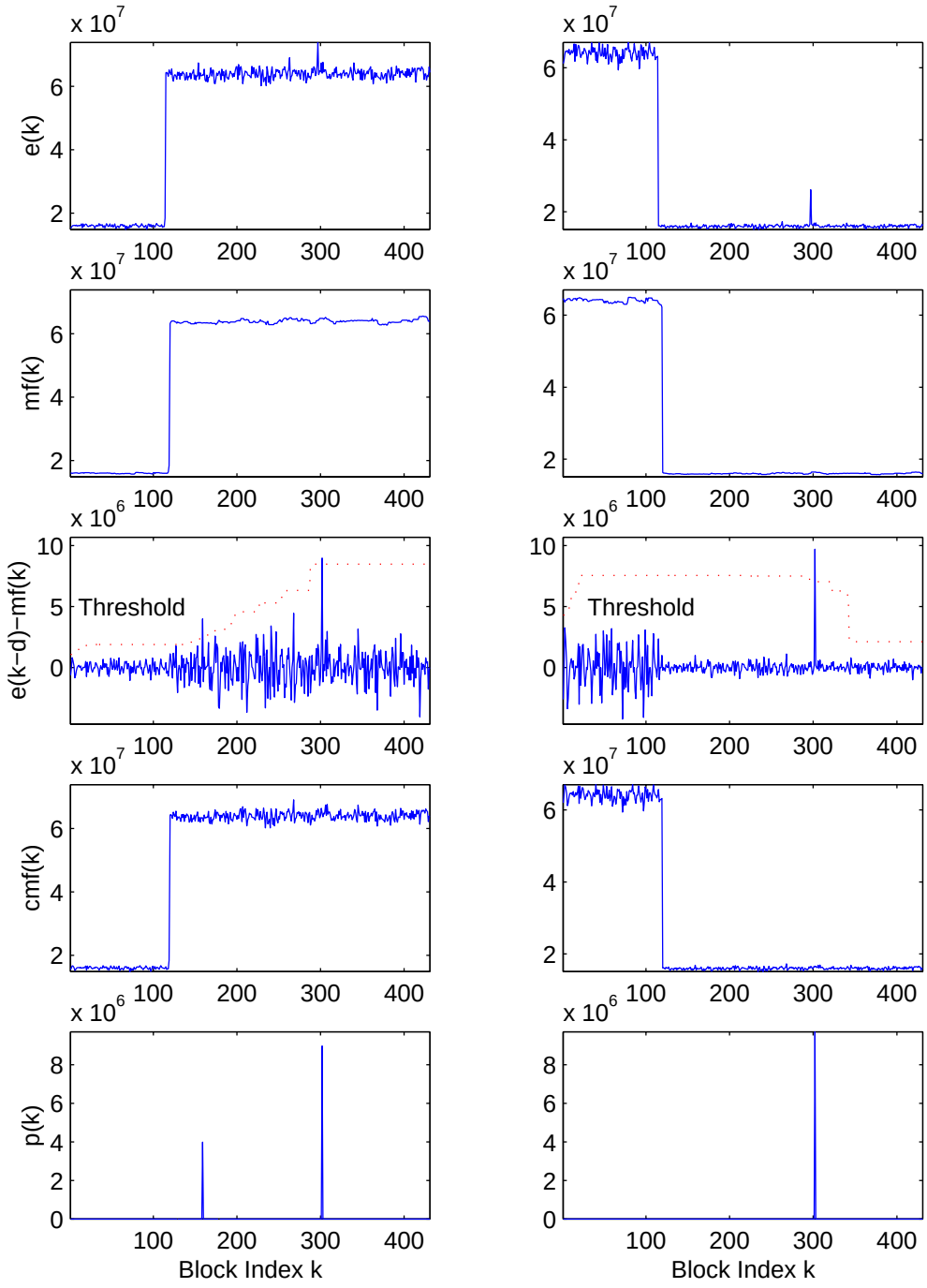


Figure 3.14: *Detection of pulses near an abrupt modification of background noise. At left, increased level (SNR=1dB) and at right, decreased level (SNR=3dB). The adaptation of the threshold (which is shown in dotted lines) uses the same parameters as in Figure 3.12.*

It should be noted that a delayed adaptation has two different kinds of consequences, as illustrated on Figure 3.14: at the beginning of an increasing noise, the threshold is too low, and the event is signaled, what is benefic. On the opposite, when the noise intensity starts to decrease, the old large threshold stays in effect for a few seconds, preventing the system to signal a possible little impulsive sound occurring during this period. The effect is also observable in Figure 3.12, in which the threshold is higher when the noise level decreases than when it increases. This phenomenon can be overcome in particular cases, measuring the tilt or the sudden power decrease, and adapting the threshold accordingly, using a temporary alteration of the sensitivity.

3.3.3 Performance

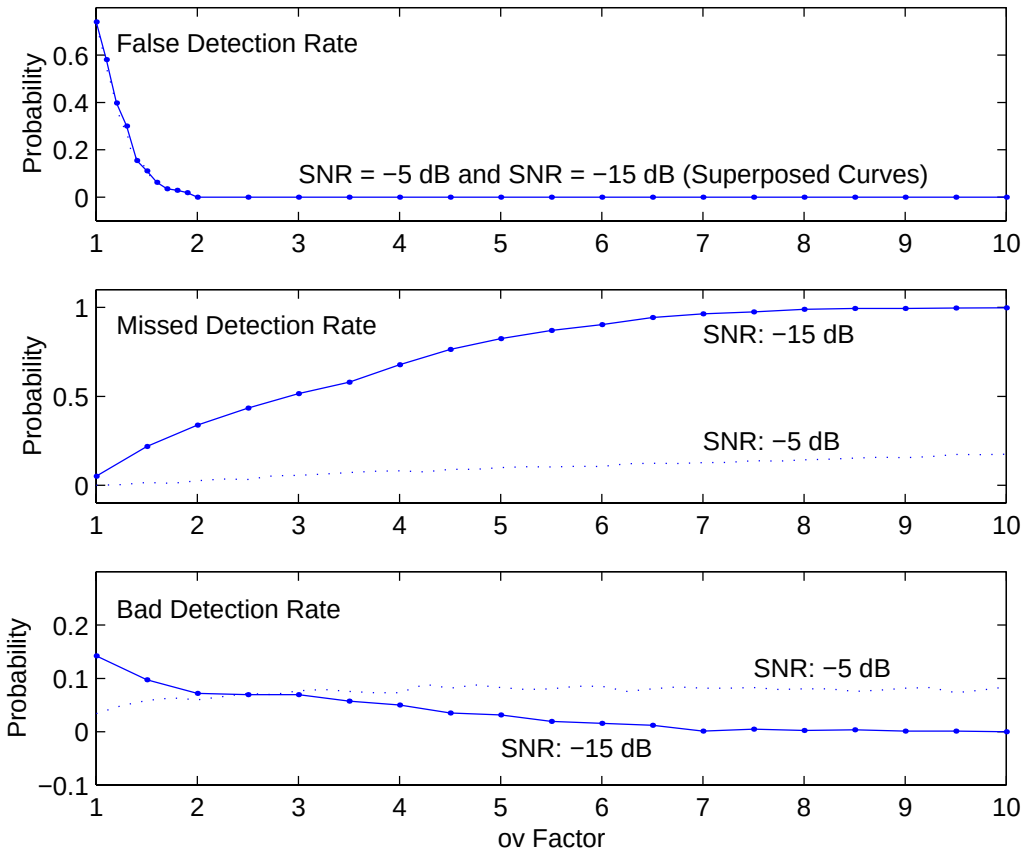


Figure 3.15: *Variation of false, missed, and bad detection rates, for ov values in the range 1 to 10. Two different gaussian white background noise levels are considered: -5 and –15 dB. Measured using the medium database.*

The performance of this median filter-based detection method is slightly reduced, compared to the one of the normalization method presented in section 3.2. At –5 dB, no good solution can be found for the sensitivity, though this was still possible for the previous technique. Furthermore, bad detection rates are more important. As shown on Figure 3.15, the best sensitivity compromise seems to correspond to *ov*=2.0. Figure 3.16 uses this value to illustrate the detection performance.

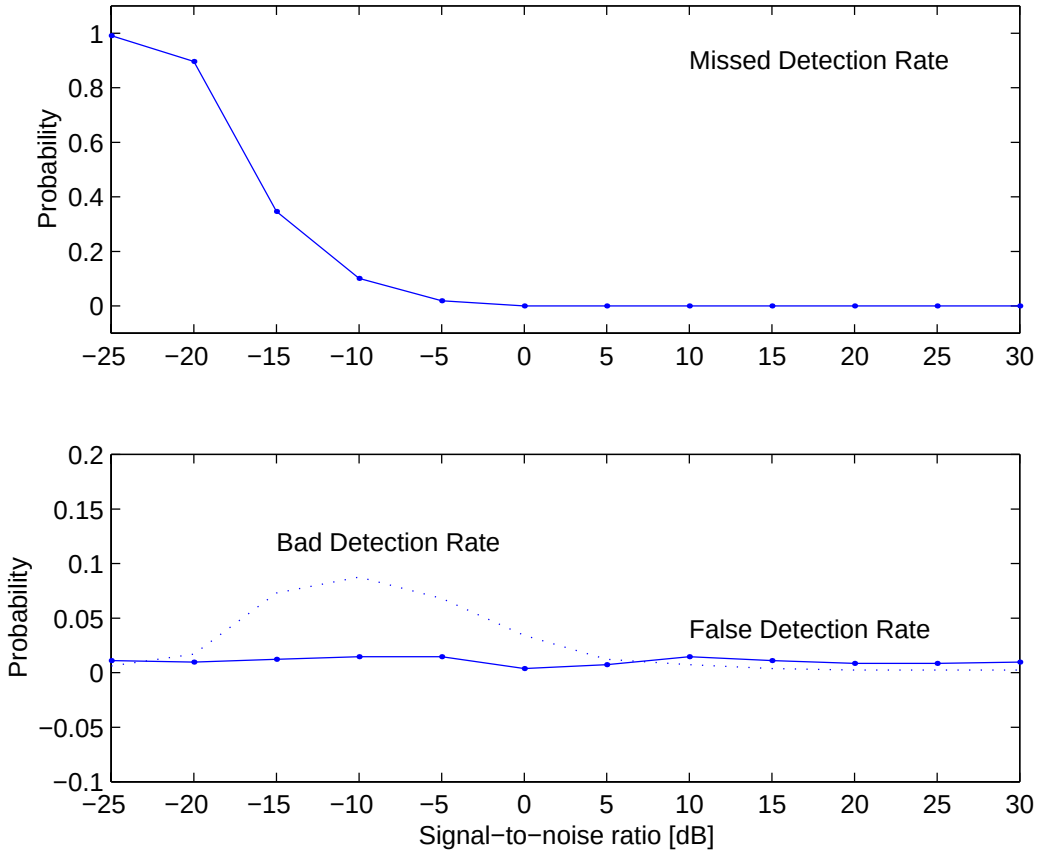


Figure 3.16: *Optimized performance of the detection method: missed, false, and bad detection rates, for ov=2.0, and for gaussian white background noise. Measured using the medium database.*

The advantage of this median filter method over the previous one is that the presence of the pulse is generally detected during all its duration, and not only at the attack. In our application, this added information is useful, to know the duration of the signal to consider at subsequent analysis and recognition stages. However, one important drawback of the technique is the delayed reaction to impulsive events, what could be critical in some dangerous

situation. Furthermore, the adaptation of the threshold is not immediate, and can be problematic during variation of the background noise.

To overcome this delay problem, a derived method is proposed in the following section, replacing the central median filter by another simple thresholding, directly applied on the power sequence. Only the threshold estimation and adaptation is based on a long-term median filtering, as described in paragraph 3.3.2.2. In this way, the detection can be near instantaneous.

3.4 Detection Using a Threshold on the Power Sequence.

This section simplifies the median filter detection scheme, and directly applies a threshold on the power sequence $e(k)$ of paragraph 3.2.1. A detection flag is signaled when the power exceeds the threshold. This threshold th is derived considering both current mean level m and standard deviation std of the background noise. Taking a factor ov_2 into account, which is inversely proportional to the sensitivity, the expression for th is as follows (the subscript 2 is here to mark the difference with the ov factor of the previous section):

$$th = ov_2 \cdot std + m \tag{3.10}$$

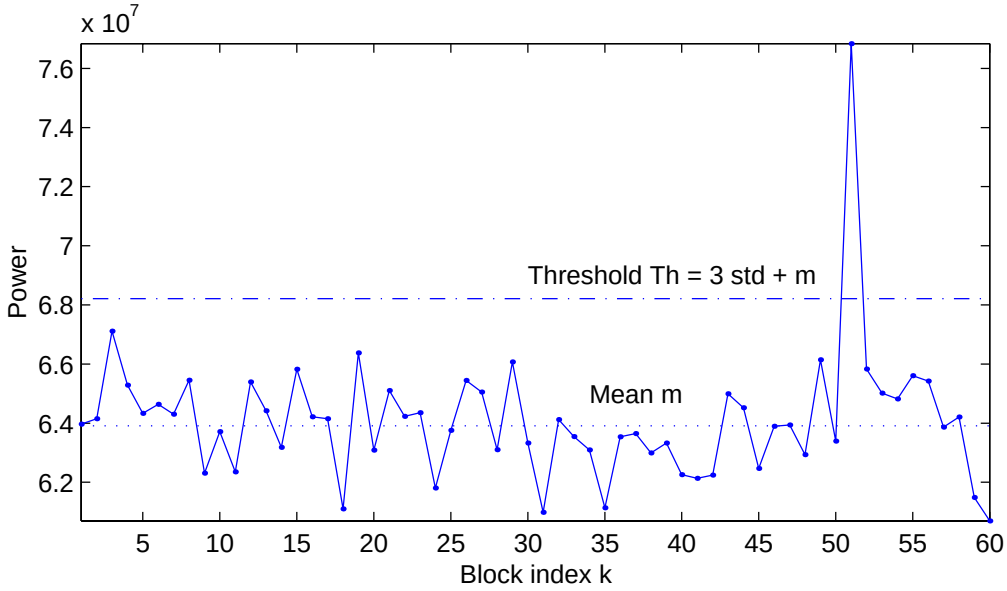


Figure 3.17: *Illustration of the power sequence thresholding, with* $ov_2 = 3$.

This instantaneous technique of detection is appropriate in constant background noise environments. An example is provided on Figure 3.17, with an ov_2 factor of 3.

When the background noise changes, the threshold has to be adapted. This would require a running evaluation of *m* and *std*. In fact, comparing with Figure 3.10, it can be observed that the new threshold *th* is equivalent to the median filter threshold of previous section, to which the mean level *m* of the background noise is added. Therefore, the threshold adaptation technique described in section 3.3.2.2 can be used all the same, adding $mf_a(k)$ to the updated median filter threshold (see Figure 3.12). In another way, direct estimations of *m* ($mf_a(k)$) and *std* (using $cmf_a(k)$ - $mf_a(k)$) with a long-term median filtering of the past power sequence would be possible too.

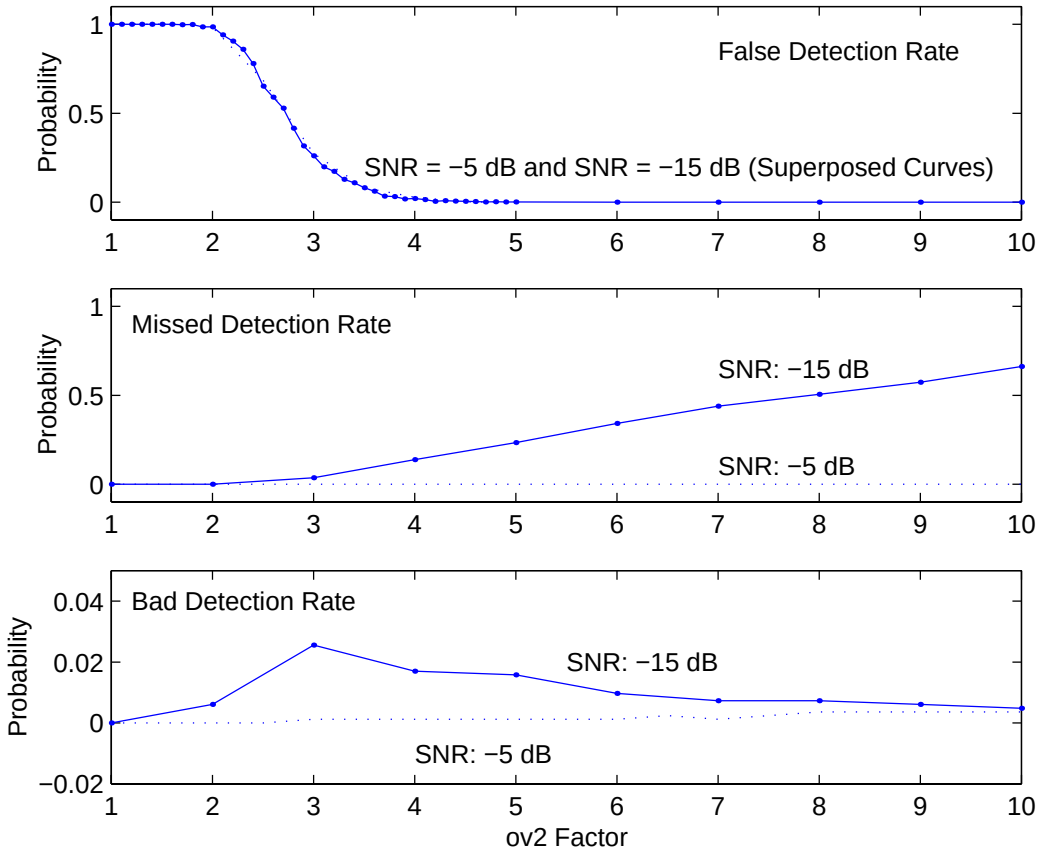


Figure 3.18: *Variation of false, missed, and bad detection rates, for ov_2 values in the range 1 to 10. Two different gaussian white background noise levels are considered: -5 and –15 dB. Measured using the medium database.*

Once more, the filter tap number L_a must be large enough to allow most impulsive components to be eliminated, but not too important to limit the update delay.

This technique exploits the advantages of the median filter scheme, without being highly concerned with its delay drawback. Only the adaptation of the threshold is delayed, with the implications described in section 3.3.2.2. The performance of the method is shown below:

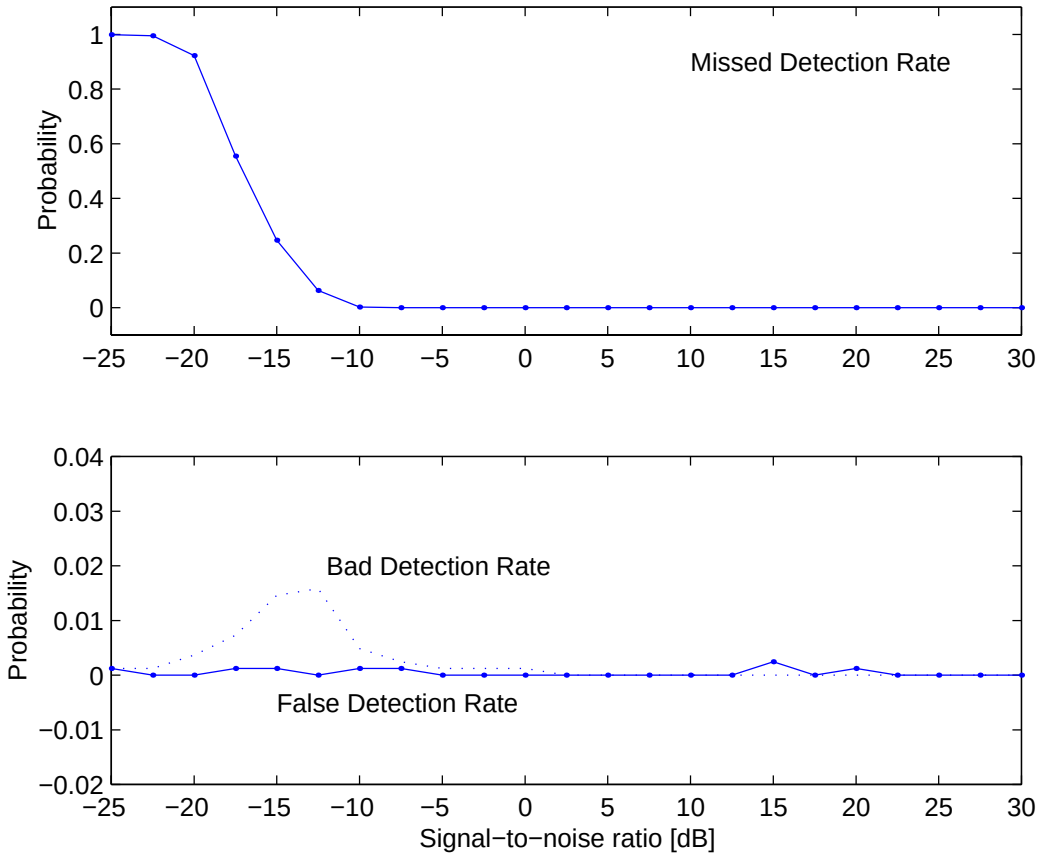


Figure 3.19: *Optimized performance of the detection method: missed and false detection rates, for ov_2=5, and for gaussian white background noise. Measured using the medium database.*

It can be seen that good detection results are present again for SNR higher than –5 dB. The bad detection rates are very low. It must be noticed that the ov_2 and ov factors can not be directly compared, because their definitions are different. A globally satisfying value for ov_2 was found to be 5.

3.5 Conclusions

In this chapter, three detection methods were presented. The first technique, which is based on the normalization of successive windowed power sequences, is aimed to be very simple, with the lowest mathematical complexity. It offers the possibility to immediately signal a possible impulsive sound. The obtained performance turns to be slightly better than the performance of more advanced techniques, that exploit the median filter concept. As a drawback, this method would not be able to signal successive events, appearing one after the other in a very short duration.

The median filter technique, discussed secondly, is more complex because of the sorting operation that is involved. Of course, sorting the successive power sequences can be simplified, performing a simple update of the sequence, removing the oldest power value, and correctly ranking the incoming one. However, even if the transition domain which suppresses both false and missed detection probabilities is generally larger than in the previous method (compare Figure 3.6 and Figure 3.15), the performance more rapidly decreases when the background noise is important. Furthermore, bad detection rates are rather important, and, as observed in Section 3.3, the major inconvenience of the median filter is the involved delay.

To prevent this annoying time delayed detection, a third method was finally proposed. This last solution uses the median filter only to remove the pulses and provides a good way to update the threshold, which is applied directly on the power sequence. This technique appears as the best compromise, showing very low false, missed and bad detection rates, even better than those provided by the traditional median filter concept.

Before closing this chapter, it must be mentioned that some more experiments on detection are presented in [Dufa00]. In this article, a slightly different method is presented, using a median filter, and normalizing its output. Then, an adaptive threshold is used, based on the past sequence variance. In fact, this method is a mixture of the first and second techniques presented in this chapter, showing identical performances. However, its mathematical load and memory requirements are more important, because of a long-term-based threshold adaptation.

In the next chapters, the problem of recognizing the detected pulses is addressed, starting by describing possible ways to analyze the signal.

4. Signal Analysis

After reminding the most important part played by the signal analysis stage in a recognition system, this chapter introduces the basic principles of features extraction. The difficult job of features selection and optimization is discussed.

Then, several impulsive sound analysis schemes are presented. The traditional short-time Fourier transform is first considered, providing a uniform time-frequency parameterization of the signal energy. Another non-uniform frequency segmentation is envisaged too. Finally, several analysis methods are considered, inspired from the speech recognition community. Cepstral and Mel-frequency cepstral coefficients are presented, as well as techniques exploiting behavioral models of the human ear.

4.1 Features Extraction

The front end of a recognition system is usually composed of an analysis stage to extract specific features from the signal to be classified. Those features must provide the classifier with a smart information, being able to discriminate between the different involved classes.

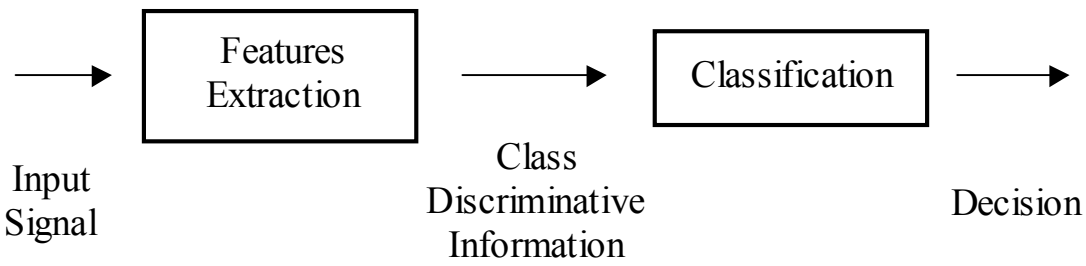


Figure 4.1: *Main steps during the recognition process.*

The features extraction [Kil96] is the most important part of a recognizer. If the features are ideally good, the type of classification architecture won't have much importance. On the opposite, if the features cannot discriminate between the concerned classes, no classifier will be efficient, as advanced as it could be. In practical situations, features always present some degrees of overlap from one class to the other. Consequently, it is worth using good and adapted classification architectures. It is important to realize that both blocks of Figure 4.1 are interdependent and should be designed, taking account one of the other.

Ideally good features should present the following properties:

- They have to emphasize the differences between classes (class separability), exploiting dimensions in which those differences are clearly observable.
- They have to be robust to noise disturbance, preserving the class separability as far as possible.
- Their intra-class variances should be minimal, and their inter-class means well separated.
- The features number, or dimension, has to be sufficient, but limited. Too many features can turn the system into confusion, according too much importance to particularities of intra-class signals, forgetting general aspects. This fact is the well-known "Curse of Dimensionality" [Couv97], which is strongly emphasized when few examples are presented to the classifier at learning time. Furthermore, the larger the dimension, the more dispersed the features, and in the end, the more complex the required classifier architecture. Adding useless features is equivalent to adding noise to the signal.
- A high correlation between features should be avoided, as much as possible. In some cases, a slight amount of correlation can be benefic, one extracted feature reinforcing the other.

Those requirements are difficult to be satisfied all together, and the feature selection task turns to be a Chinese puzzle. However, some guidelines do exist, to help performing the job. Features analysis and optimization is possible, after a pre-selection. In fact, the features extraction block often integrate the following successive operations:

- *Signal Pre-processing*: Removal of some potentially disturbing noise, for example using a filtering process*;*
- *Signal Projection*: Transformation of the input signal into another domain, in which class discriminative parameters can be more easily extracted. A typical projection example is the Fourier transform, which allows an efficient discrimination between frequency contents.
- *Features Extraction*: Calculation of *N* potentially interesting features.
- *Features Optimization*: Normalization of the features and derivation of a subset of *M* features, with better discriminative power.

The last step, *Features Optimization,* is an important subject, extensively developed in [Kil96]. The next sections will present a summary of the main available techniques.

4.1.1 Features Normalization

The goal of this preliminary normalization is to avoid bad numerical behavior during the classification stage, resulting in the so-called *ill-conditioning* situations. Actually, very low numerical values can be encountered when important scale differences are present between different dimensions of the features vector. As a consequence, some recursive algorithms used to learn the class properties may not converge. This problem is especially increased when the features number is too large or when an important correlation exists between features.

Supposing that one *D*-dimensional features vector is obtained at each observation time:

$$\vec{f} = \begin{bmatrix} f_1 & f_2 & \dots & f_D \end{bmatrix}^t \tag{4.1}$$

then, for *K* observations, the features can be presented in the form of an observation matrix of dimension *D* x *K*:

$$F = \begin{bmatrix} \vec{f}_1 & \vec{f}_2 & \dots & \vec{f}_K \end{bmatrix} \tag{4.2}$$

This set of observations can be used to estimate the global mean vector $\vec{\mu}$ and the global covariance matrix *C* of the features:

$$\vec{\mu} = [\mu_1 \quad \mu_2 \quad \ldots \quad \mu_D]^t \tag{4.3}$$

$$C = \begin{bmatrix} c_{11} & c_{21} & \ldots & c_{D1} \\ c_{12} & c_{22} & \ldots & c_{D2} \\ \ldots & \ldots & c_{ij} & \ldots \\ c_{1D} & c_{2D} & \ldots & c_{DD} \end{bmatrix} \tag{4.4}$$

with the mean elements:

$$\mu_d = \frac{1}{K}\sum_{k=1}^{K} \vec{f}_k(d) \quad \text{for each features dimension } d = 1,...,D \tag{4.5}$$

and the covariance values (unbiased estimation):

$$c_{ij} = \frac{1}{K-1}\sum_{k=1}^{K}\left(\vec{f}_k(i) - \mu_i\right)\left(\vec{f}_k(j) - \mu_j\right) \quad \text{for each dimension pair } (i,j) \tag{4.6}$$

Each element of $\vec{\mu}$ is the mean in the corresponding features dimension, and the diagonal elements c_{ii} of the symmetric covariance matrix represent the variance σ_i^2 in the i^{th} features dimension. The normalization consists in scaling the features range in each dimension *d*, to get zero mean values and unit variances:

$$\vec{f}_k^{\,norm}(d) = \frac{\vec{f}_k(d) - \mu_d}{\sigma_d} \tag{4.7}$$

The normalization involves all the observations, without taking care of their class membership. In this text, all the mentioned features will be considered as normalized using the above expression. It can be noticed that in this case, the features covariance matrix and correlation matrix are equal, because of the zero mean.

4.1.2 Optimization Using Features Selection

As explained above, better classification results are obtained using a limited number of non-correlated features that maximize the class separability. In this perspective, the features optimization can simply consist in a selection of the *M* best features among *N* previously extracted ones. A performance criterion

is then needed to estimate the quality of the features. This criterion can be the recognition performance itself, involving a database to test the system, and the whole classification process. Of course, the selection must be done once, before any use of the system. All features combinations could be tried, but more reasonably, simplified protocols like the *Sequential Forward Selection (SFS)* recursion [Couv97], the *Viterbi* algorithm [Vite67], or the sub-optimal *Add-on* algorithm [Gold76] should be used. All those methods consist in adding one feature after the other until M features are collected, and keeping at each step the new feature that maximizes the performance criterion. The opposite procedure, the *Sequential Backward Selection (SBS)* is also possible, starting with all features, and removing the bad ones, one after the other (*Knock-out* algorithm, [Samb75]).

Those features selection techniques are highly complex and time-consuming. However, they are the only way to really extract the best features, and perform a multi-dimensional optimization. Otherwise and more simply, a ranking of each one of the N features can be done, using criteria like the *Fisher Discriminant Ratio (FDR)*, or the *Multimodal Overlap Measure (MOM)* [Kil96]. Those methods allow estimating the capability of each feature to discriminate between the classes, by measuring the overlap of their PDF (Probability Density Function) from one class to another.

For example, the FDR that is well fitted to gaussian PDFs, can be determined for each feature:

$$FDR = \frac{\sum_{i=1}^{N_c}\sum_{j=1}^{N_c}\left(\mu_i - \mu_j\right)^2}{\sum_{i=1}^{N_c}\sigma_i^2} \tag{4.8}$$

In each feature dimension, it expresses the ratio between the distance separating two classes i and j, and their variances. In the above expression, contributions from all N_c classes are cumulated.

In this way, the M features with the best non-overlapping properties can be selected. The drawback with those criteria is that they generally do not integrate the correlation between features, the measure being individual. More concretely, the obtained recognition results are said to be 8% lower than with a multi-dimensional optimization [Kil96]. To some extent, more advanced criteria can be used when a high degree of correlation exists between features (see *Generalized Rayleigh Quotient*, [Kil96]).

4.1.3 Optimization Using Linear Transformations

Another method to determine optimized features consists in performing linear transformations of the original features space. Such transformations can both decorrelate the features or improve their class discriminative capability. The goal is to do a change in the coordinate axes to get the maximum orthogonality between features, or respectively to increase a separability criterion. The transformed features can be ranked according to this criterion, allowing an optimized reduction: useless dimensions are first removed (increasing classification performance), and then, if reduction is continued, the decrease of performance due to further feature removal is minimized.

Supposing the features are zero-mean (normalized using (4.7)), or if the features PDF are gaussian, no correlation between two features can be identified as statistical independence, what implies zero covariance values [Deco84]. Thus, the decorrelation can be done with an orthogonal transformation of the features vector, getting rid of the non-zero off-diagonal elements in the features covariance matrix (diagonalization):

$$\vec{f}_{tr} = A\vec{f} \tag{4.9}$$

In this expression, $\vec{f}_{tr}$ is the transformed features vector, and the transformation matrix A is obtained using a singular value decomposition of C (Karhunen-Loève expansion [Theo98]). This expansion performs a change of coordinate that makes the principal axis (coordinate $\vec{e}_1$) of the new space parallel to the direction of the maximum feature variance.

$$A = \left[\vec{u}_1 \quad \vec{u}_2 \quad \ldots \quad \vec{u}_D\right]^t \tag{4.10}$$

The $\vec{u}_i$ are the eigenvectors of the covariance matrix C, sorted from the highest to lowest corresponding eigenvalues λ_i. The largest eigenvalue corresponds to the feature with the largest variance. In this way, a reduction of features dimension can be done in the new space, only keeping the most relevant features that are supposed to correspond to the larger variances. This operation is called the *Principal Component Analysis (PCA)* [Joll86]. If D_r is the reduced dimension, the matrix A is (D_r x D)-dimensional, and the reduced features are obtained with (4.9). According to [Swet96], D_r can be selected as:

$$\frac{\sum_{i=D_r+1}^{D} \lambda_i}{\sum_{i=1}^{D} \lambda_i} < 5\% \quad \text{with} \quad \lambda_1 > \lambda_2 > \ldots > \lambda_D \tag{4.11}$$

This method allows to remove correlation between features, and to reduce their number. This is a positive point, but all the above process is still class-independent, and de-correlated features do not imply good class separability! To take the separability into account, another transformation can be next performed: a generalization of the FDR separability criterion (4.8) is used, maximizing the ratio between inter-class distances and intra-class variances. The *Fisher Covariance Matrix* J_{wb} is defined as the division of an *inter-class* scatter matrix S_b by an *intra-class* scatter matrix S_w. The singular value decomposition is then applied on the Fisher covariance matrix, aligning the principal axis in the direction of the maximum separability. This technique is called the *Linear Discriminant Analysis (LDA)*. It allows an important reduction of the features dimension, offering equal or better performance.

$$J_{wb} = S_w^{-1} S_b \tag{4.12}$$

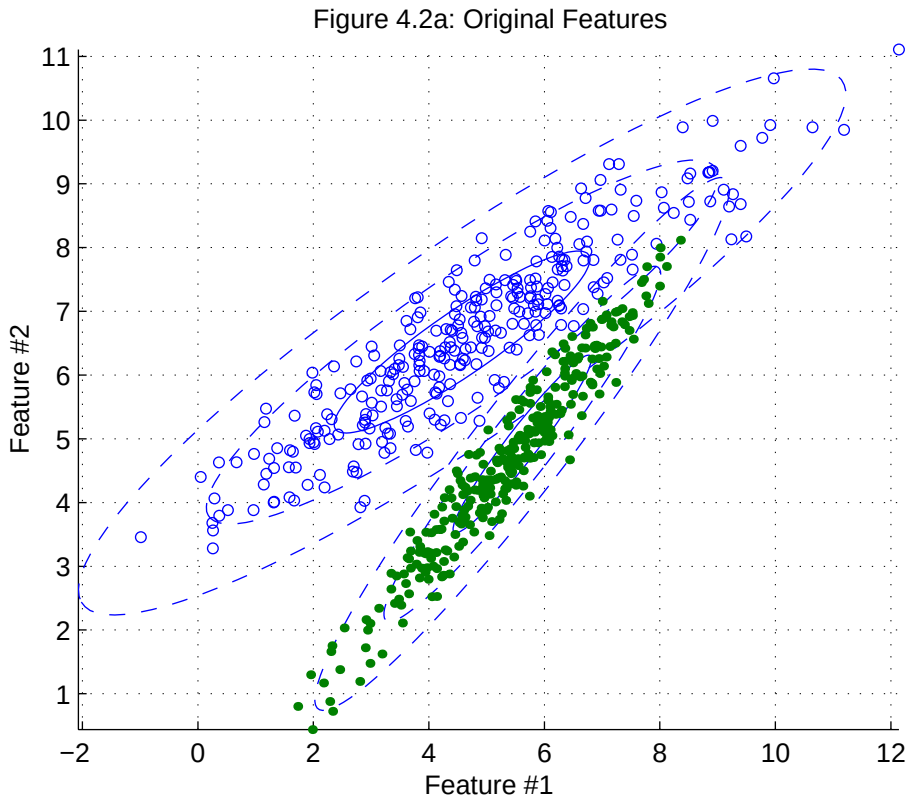
with:

$$S_b = \sum_{i=1}^{N_c} (\vec{\mu}_i - \vec{\mu}_0)(\vec{\mu}_i - \vec{\mu}_0)^t \tag{4.13}$$

$$S_w = \sum_{i=1}^{N_c} C_i = \sum_{i=1}^{N_c} \left[\frac{1}{K_i - 1} \sum_{k=1}^{K_i} \left(\vec{f}_k^{\,(i)} - \vec{\mu}_i \right) \left(\vec{f}_k^{\,(i)} - \vec{\mu}_i \right)^t \right] \tag{4.14}$$

In the above expressions, the *d*-dimensional vectors $\vec{\mu}_i$ and $\vec{\mu}_0$ respectively hold the i^{th} intra-class means and the global means involving all classes, while C_i is the i^{th} intra-class covariance matrix. K_i observations $\vec{f}_k^{(i)}$ (k = 1, …, K_i) are involved in each class *i*. Unfortunately, because of the non-symmetric properties of J_{wb}, solving the eigensystem may not be guaranteed. Then, a hot trick is often used, performing the singular value decomposition on two symmetric matrices derived from S_b and S_w (see Appendix A).

Figure 4.2 shows the difference between PCA and LDA transformations. The illustration involves bi-dimensional features, discriminating between two classes. Figure 4.2a shows the initial coordinate axis, while Figure 4.2b and Figure 4.2c illustrates the new positioning, aligning the principal axis according to the maximal variance and separability, respectively. In the next chapter, both transformations method will be applied to sound classification and compared. Both transformations can be applied successively, to optimize the reduction. The inconvenience of such techniques is their mathematical complexity. However, balanced with the gain coming from the dimension reduction, the overall processing time can be lower for recognition.



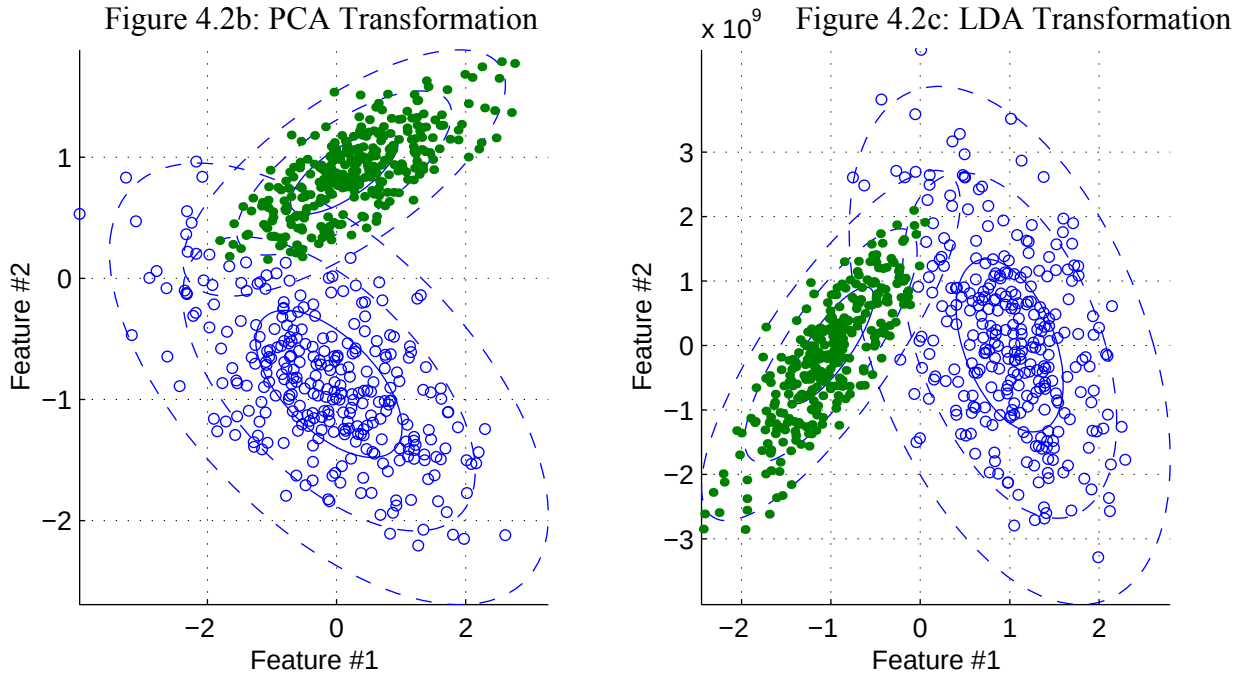


Figure 4.2: *PCA and LDA features transformation for 2 classes and bi-dimensional features. No reduction is applied for comparison purpose.*

4.2 Spectrogram Features

The traditional short-term Fourier transform [Rabi78] is a well-fitted method for the analysis of signals like impulsive sounds, whose frequency spectrum varies in time. The time-frequency analysis proposed in this section consists in calculating the energy spectrum of the signal for each successive time frame. The frames have a duration of N samples, N_{ov} of them overlapping with the previous frame. To reduce computation, a modified Hanning window (Figure 4.3) is applied on each frame. Both Hanning flanks are kept apart by a constant unit zone, allowing the overlap ratio to be lower than 50%. When the signals are sampled at 44.1 kHz, typical frame designs are N = 512, 1024 or 2048 samples (~12, 23 and 46 ms respectively), and N_{ov} = N/4 or N/8.

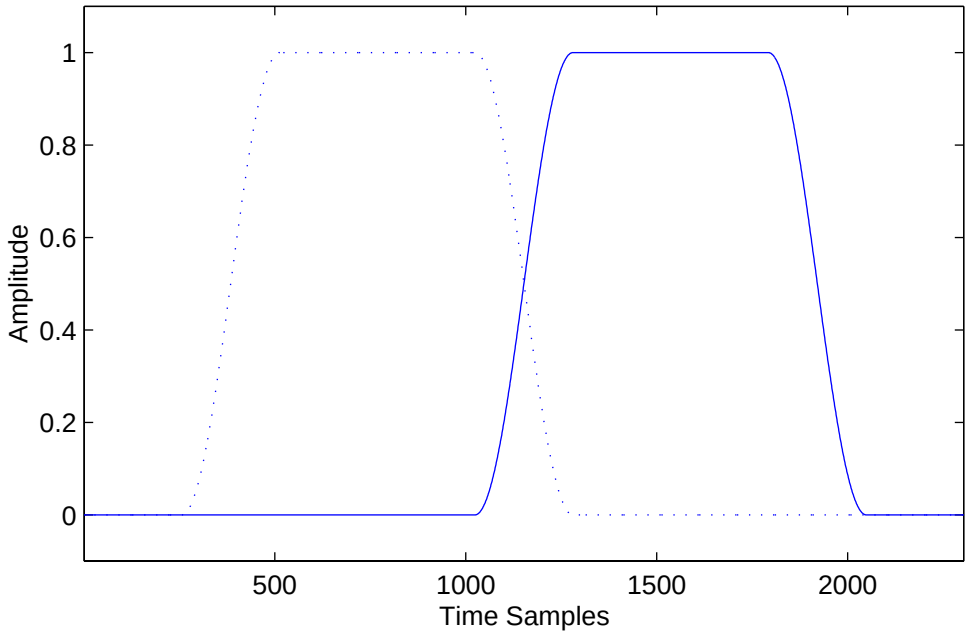


Figure 4.3: *Modified Hanning windows for N=1024 and N_{ov} =N/4.*

Each energy spectrum is made of N/2+1 values over the unilateral frequency axis (0 -> 22050Hz) . Those values are regrouped and averaged in N_b frequency bands, to build a low-resolution energy spectrum. The band structure can be uniform, consisting of 5, 10, 20 or 50 bands, for example. In another way, non-uniform distributions (Bark spectrogram) can be considered, simulating the human ear critical band analysis described in Sub-section 4.4.1 [Perc00]. Figure 4.4 shows a spectrogram example for a closing door sound, in the case of uniform frequency decomposition. Normalization by the maximum encountered value is recommended.

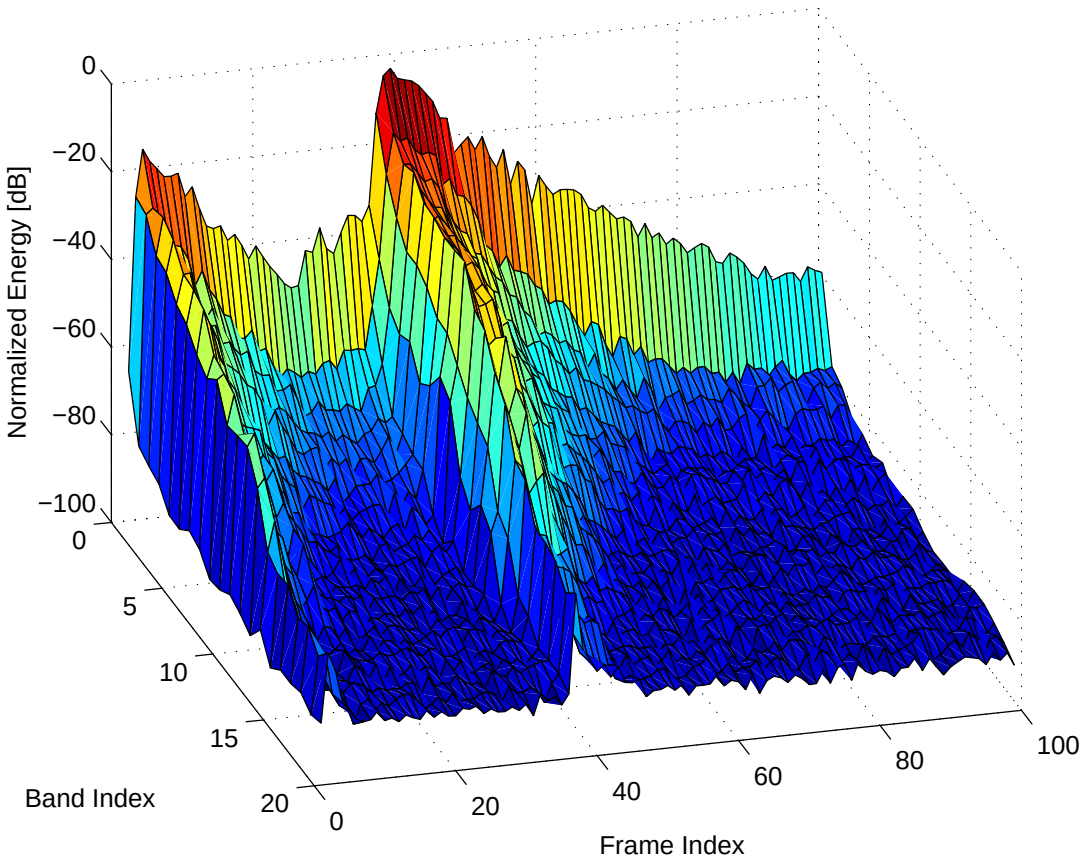


Figure 4.4: *Spectrogram representation of a slowly closing door, with N=512, N_{ov} =N/8, and N_b =20 uniform bands.*

The final features set can be selected in two ways. A first solution (that will be referenced to as *spectrogram type A*) consists in regrouping the spectral information extracted from several successive frames, and concatenating it to build one large features vector. The concatenation can be done, for example, during one second. This is a real spectrogram that integrates the temporal evolution in the features. Then, it can be guessed that a rather simple classifier could be associated to this features scheme. However, if N_f frames are taken together, the number of dimensions by features vector increases to $N_f N_b$, and a reduction process is recommended. From an implementation point of view, this method may require a lot of memory, depending on the number of concatenated frames.

The principle of the second solution (*spectrogram type B*) is to consider each frame individually, providing N_b dimensions in each observation vector. Then, a classification result will appear for each frame, and the balance will

have to be determined for the whole signal. This technique is appropriate in case of immediate recognition requirements and needs not as much memory as the first method. However, no temporal evolution information is included in the features. As a consequence, its use may be better adapted to more advanced classifier architectures that integrate temporal capabilities.

Hybrid solutions of both method A and B are sometimes used. The idea is to consider each frame individually (method B), adding some extras dimensions to the features vector, for incorporation of temporal evolution. As an example, the extra-features could represent the energy evolution in some specific spectral band, during several previous frames.

4.3 LPC and Cepstral Features

4.3.1 LPC Coefficients

In the speech processing community, the analysis of the speech signal often involves the LPC (*Linear Prediction Coefficients*) model. This section won't go into much detail about LPC, the reader being invited to consult the large published literature, as [Rabi93] for example. However, in a few words, this analysis model consists in a linear all-zero analysis filter, which is adaptively designed to separate between two main components of the speech signal:

- The *excitation*, output of the LPC analysis filter, which represents the air pressure waveform resulting from the vibration of the vocal cords that are excited by the air expulsed from the lungs. This excitation signal is like a white-shaped noise, to which the periodic pulses induced by the cord vibrations are superposed. Consequently, the pulse period or *pitch* can be derived from the excitation signal (*pitch detection* [Rabi93]).

- The *spectral envelope* of the speech, induced by the shape of the vocal tract. In fact, the vocal tract is represented by the inverse analysis filter (synthesis filter), whose input is the excitation. The spectral envelope information is hold in the coefficients of the adaptive filter, the larger the number of coefficients, the more precise the envelope.

The LPC filter coefficients are derived for each overlapping (1/3) hamming windowed frame (20 or 30 ms) of speech signal, using a short-term prediction $p_r(k)$ of the speech samples $x(k)$. A minimization of the prediction error $e(k)$ is

performed, leading to the well-known autocorrelation equations, solved using the Levinson-Durbin recursion [Osha00], which provides the prediction coefficients a_i in each frame. The prediction error (that is the excitation, sometimes also called *residual*) can be expressed as below, where p indicates the number of filter taps:

$$e(k) = x(k) - p_r(k) = x(k) + \sum_{i=1}^{p} a_i x(k-i) = \sum_{i=0}^{p} a_i x(k-i) \qquad (4.15)$$

In speech or speaker recognition problems, the number of filter taps is chosen between 8 and 12. It must be sufficient but not too high, because the envelope representation would become too fine, including redundant pitch information that could potentially disturb the classification performance. The LPC coefficient features are well adapted to speech recognition, apparently providing better results than the time-frequency analysis, as presented in the previous section or using a filterbank. For impulsive wide-band signals, they could be an alternative (or a complement) to the short-term Fourier method, provided that the LPC order is increased to 15 or 20, allowing a more fine analysis of the spectrum evolution.

4.3.2 Cepstral Coefficients

Compared to LPC, the cepstral transformation is an improved and more concrete way to isolate the two components of the speech signal [Dell93]. After evaluating the amplitude spectrum of an input signal $x(k)$, the real cepstral transformation $C_x(n)$ is defined as the inverse Fourier transform of the amplitude logarithm,

$$C_x(n) = F^{-1}\left\{\log\left|F\left\{x(k)\right\}\right|\right\} \qquad (4.16)$$

where F represents the Fourier transform operator. This transformation is very interesting for speech, because in the cepstral domain, the excitation and vocal tract components are linearly combined. Indeed, in the speech production model, the excitation is convolved with the impulse response of the inverse LPC filter (vocal tract synthesis filter) to produce speech. When transposed in the frequency domain, this convolution becomes a simple product, and taking the logarithm turns it into an addition! Then, the linear Fourier transform (or its inverse) can be taken once again, to analyze the "frequency" content of the log spectrum! Here lies the advantage of the cepstral transformation, because

for speech, the envelope shows slowly varying spectral properties. On the opposite, the excitation of voiced speech is made of important spectral variations (pitch and harmonics), that will rather appear in the higher part of the cepstral axis. A more easy discrimination between excitation and envelope components is then possible.

In speech recognition, for computational reasons, only the first cepstral coefficients (15 or 20) are usually kept, corresponding to the envelope information. When used, the excitation information is rather extracted using traditional LPC technique. In this case, a simpler derivation [Rabi93] is available to calculate the first N_c cepstral coefficients from the LPC filter coefficients. N_c cepstral coefficients $CC(i)$ can be derived at each time frame. The expression (4.17) illustrates the direct derivation using the definition [Davi80]:

$$CC(i) = \sum_{n=0}^{N-1} X_n \cos\left(\frac{\pi i n}{N}\right) \qquad \text{for } i=1,\ldots, N_c \tag{4.17}$$

N represents the frame length or number of spectral points, and X_n stands for the log-amplitudes corresponding to frequency n. The resulting set of cepstral coefficients is generally smoothed using a sinusoidal window:

$$cc'(n) = cc(n)\ w(n), \qquad \text{with} \qquad w(n) = 1 + \frac{N_c}{2}\sin\left(\frac{\pi n}{N_c}\right) \tag{4.18}$$

This windowing operation de-emphasizes both first coefficients that are sensitive to the spectral tilt, and last coefficients, more sensitive to noise.

From a performance point of view, speech recognition systems based on cepstral features are said to be slightly better and more robust than those based on LPC [Davi80]. The final windowing takes an important part in this fact, as well as the *Delta Cepstral Coefficients*. Those coefficients can be computed as the difference between cepstral coefficients of the current and past frames, including temporal evolution information in the features [Dell93].

For impulsive wide-band sounds, cepstral features may be interesting, being able to decorrelate slowly and fast varying components of the spectrum. However, a more important part of the axis may have to be considered (~50 coefficients), to take account of the more distributed variability of spectral components. This requirement would increase the computational load,

compared to speech conditions. As an illustration, for impulsive sounds, rather flat envelope spectrum can often appear during the short duration of the attack, giving rise to large components in the lower part of the cepstral axis. After the attack, on the opposite, periodic signals are sometimes encountered, and components in the highest part of the cepstral do appear.

4.4 Features From Human Ear Models

4.4.1 Mel-frequency Cepstral Coefficients

In the psychoacoustic approach of human ear modeling, the *mel-frequency* or similarly *Bark* scale corresponds to the human perceptual frequency scale, as measured by Stevens [Stev40]. The mapping between the linear frequency axis and the human scale seems to be linear till 1 kHz, and gets logarithmic above this value.

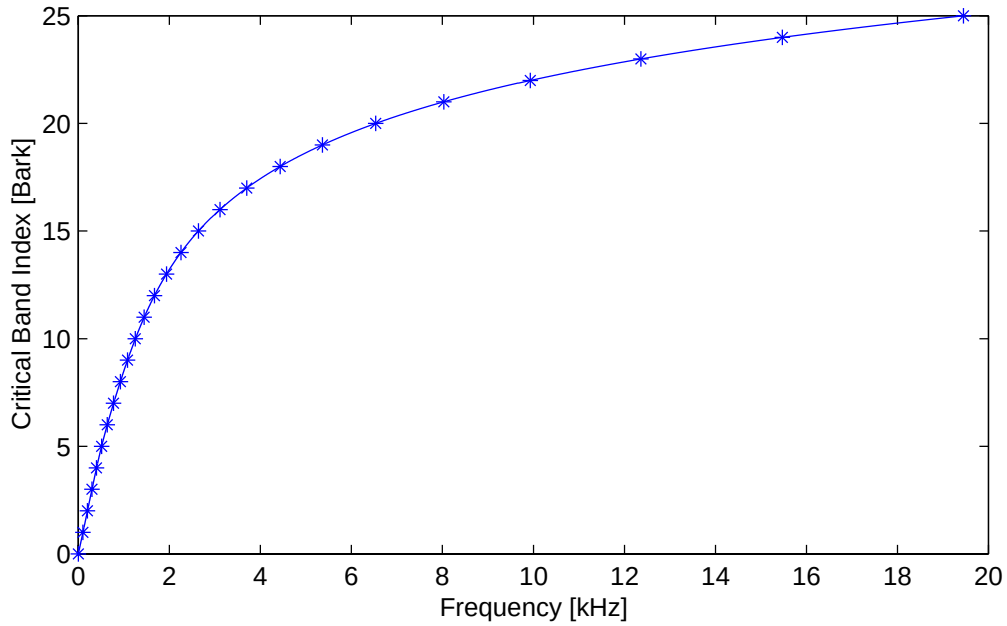


Figure 4.5: *Linear frequency to Bark scale mapping. The stars represent the borders of the critical bands, corresponding to integer Bark values.*

It can be expressed as a function of the linear frequency *f* in kHz [Kran94]:

$$f_{bark} = 13\arctan(0.76f) + 3.5\arctan(f^2/56.25) \qquad (4.19)$$

In fact, the bark scale splits the frequency axis in N_{cb} = 25 successive *critical bands* of increasing bandwidth. The bark is the critical band index, starting from the lowest frequencies. The critical bands were designed by performing the following experiment: people were asked to listen to a narrowband noise whose bandwidth was progressively increased, while keeping its central position and level unchanged. Surprisingly, during the bandwidth modification, the perception of the sound stayed identical till some limit was reached, depending on the noise central position. Therefore, this limit was used to set the width of the critical bands for the considered central frequency. A decomposition in 25 adjacent critical bands [John88] was arbitrarily decided for easier use, as represented in Figure 4.5 and specified in Appendix B. The critical bands are closely tied to the well-known *masking effect* [Shar70]. In fact, the whole amount of energy present in one critical band influences the perception of any one of its components, sometimes making it imperceptible.

The idea of the Mel-Frequency Cepstral Coefficients (MFCC) is to distribute the cepstral coefficients according to the critical bands, instead of the traditional linear distribution. For that, a critical band decomposition of the current frame spectrum is done using "critical band filters" [Davi80]. The log-amplitudes of the components in each band are added to provide N_{cb} resultant values Y_k. Then, a complete spectrum is rebuilt, placing zeros at indexes that do not correspond to the N_{cb} critical band central frequencies. The inverse discrete Fourier transform can finally be calculated, to get the N_c MFCC coefficients [Davi80].

$$MFCC(i) = \sum_{k=1}^{N_{cb}} Y_k \cos\left[i\left(k - \frac{1}{2} \right) \right] \frac{\pi}{N_{cb}} \qquad \text{for } i=1,\ldots, N_c \qquad (4.20)$$

In this work, the MFCC are derived using C functions provided by Jialong [Jial99]. The mel-frequency cepstral features are known to provide important performance improvements, compared to the linear cepstral coefficients [Dell93]. More generally, the use of human ear models seems to bring an improvement on the performance of speech recognition system, especially in presence of important background noise level. In the following section, more advanced auditory models are presented, to be later tested for impulsive sound recognition.

4.4.2 ERB Filter Bank and Meddis Hair Cell Models

In more advanced models of the human ear, the full auditory system is considered and studied. However till today, the complete auditory path is still not clearly understood and the elaborated models mostly rely on experiments. A lot of studies have been reported, and keep being proposed [Kim99]. It seems that beside the problem of impossible direct analysis of the auditory mechanism, difficulties come from the successive non-linear transformations that are involved, and by their large number of tunable parameters.

In a few words, sound waves are transformed into mechanical vibrations in the outer ear, at the eardrum. The little bones of the middle ear convert those vibrations into liquid pressure variations in the inner ear, that finally create traveling waves on the *basilar membrane* of the cochlea [Yang92]. The membrane moves in a transversal way, according to the energy of the incoming sound [Flet40]. Low frequencies result in movement at the beginning of a hypothetically unrolled basilar membrane, and higher frequency responses appear farther in the cochlea.

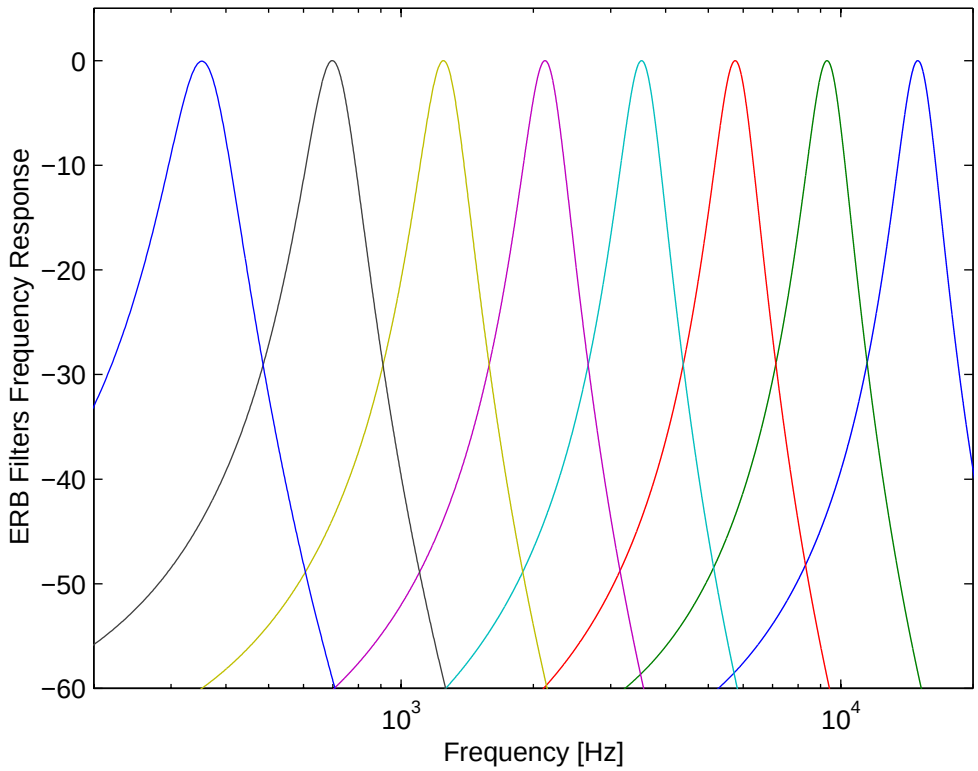


Figure 4.6: *Gammatone filterbank, with M=8.*

The frequency-to-position mapping is logarithmic, and shorter and shorter portions of membrane are dedicated to increasing frequency components. As represented on Figure 4.6, the *gammatone* filterbank made of *M cochlear filters* [Slan94] is a widely accepted model of the basilar membrane behavior. Each cochlear filter represents the frequency selectivity at specified locations. The number of involved filters (that is the resolution) can vary between 20 and 200, from one model to another. In this work, the *ERB (Equivalent Rectangular Bandwidth)* filterbank defined by Patterson and Holdsworth [Slan93] will be used.

Then, the movement of the membrane is translated to electrical stimulation in auditory nerve fibers. *Inner hair cells* uniformly disposed along the basilar membrane are excited by the bending displacement of attached filaments called *cilia*. Those cilia are little sensors that move according to the flow of the inner ear fluid resulting from membrane displacements. In the inner hair cells, the transduction to electrical activity seems to be a three-step non-linear process. At first, the moving cilia create ionic flows through several non-linear channels in the cell. Then, each ionic flow generates an electrical potential across the cell membrane. Finally, several nerve fibers convey the resultant electrical potential of the cell to the central auditory system, in the form of trains of nerve spikes. The more intense the nerve activity, the higher the spikes frequency.

Global models for those three steps have been elaborated, sometimes taking into account the well-known *lateral inhibition* effect of neural networks [Yang92]. In those models, acoustical stimuli are represented by the nerve *firing rates* associated with each cochlear filter. The firing rates can be estimated by zero-crossings measures. Many interesting auditory models such as the Seneff's Auditory Model [Sene84], the *EIH (Ensemble Interval Histogram)* [Ghit94], or the more recent ZCPA *(Zero Crossing with Peak Amplitudes)* [Kim99] have been published. In this work, the popular *Meddis Hair Cell* model will be used to test the sound recognition potentiality of such hearing-related features. The detailed development of this model would be out of this work's scope, but its description and implementation can respectively be found in [Medd86] and [Medd90]. However, it must be kept in mind that the final auditory analysis stage in the brain is totally unknown, and consequently, the model is not complete. Only the peripheral auditory stage is available. It is expected that an important features analysis (reduction and selection) is performed in our brain.

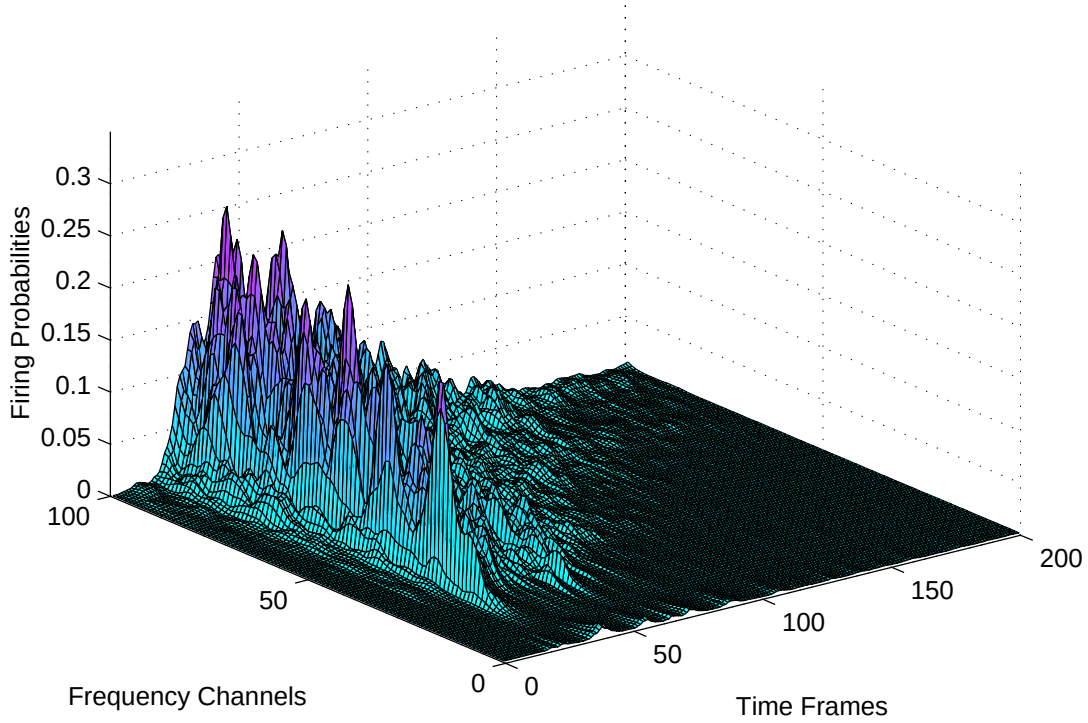


Figure 4.7: *Representation of the nerve firing probabilities for the closing door of Figure 4.4 (second part). A decimation by 50 of the time scale is applied.*

Figure 4.7 shows the *cochleogram* representation of the nerve firing probabilities obtained using the Meddis Hair Cell. The implementation uses the Matlab Auditory Toolbox, provided by M. Slaney [Slan94]. The illustration concerns the second part of the closing door illustrated in Figure 4.4. One firing rate is obtained for each cochlear channel and for each time sample. To decrease this huge number of features, a low-pass filtering of the probabilities is applied in time, followed by a decimation involving a factor of 100 for example.

4.5 Conclusion

This chapter has shown that many different and interesting features extraction schemes could be used at first sight, as a front end of the sound recognition process. However, all those methods surely not present the same

properties needed to build a good recognizer. Some of them, for example, may be better adapted for the robustness of the system.

In the next chapters, classification architectures will be studied and applied to the sound recognition problem. The efficiency of the analysis methods and optimization techniques presented in the previous sections will be evaluated and compared.

5. Recognition

This chapter presents the core of the recognition system. Classification architectures are presented, and applied to the sound recognition problem.

Three types of parametric classifiers are considered in the following sections. At first, the Bayesian classifier is described, modeling the features PDF with a normal distribution. Secondly, the approximation PDF is refined using mixtures of Gaussians (GMM). Then, the popular Hidden Markov Model (HMM) is experienced, taking time evolution into account in the recognition architecture. Finally, neural network solutions are investigated, involving a Multi-Layer Perceptron (MLP).

The performance of those classification methods is compared, when driven by the various features schemes described in the previous chapter. The three sound databases presented in the beginning of this text will be used in this purpose.

5.1 Pattern Recognition

In *Pattern recognition* problems, the goal is to discriminate between features representing different classes of interest. Two kinds of situations can be encountered. In the first one, no example signals of the involved classes are available before the system to be in operation. Then, an automatic clustering of the incoming features must be performed by the system, using a so-called *unsupervised learning*. Popular clustering methods like the *k-means Nearest Neighbors* does exist and are extensively developed in [Theo98].

In this work, however, the second situation is rather considered, where a sufficiently important and representative set of training sounds is available before use, to let the system learn the properties of the different classes. This *training set* must be carefully chosen for each class. If so, *supervised learning* schemes generally performs better than clustering recognition in complex

problems, even though those last methods show an interesting adaptation capability to variations of encountered signals.

In supervised learning, different recognition schemes are possible, depending on the possibility of estimating the features statistical distribution. *Non-parametric classifiers* can be used when a concrete measure (histogram) of the features PDF is possible. On the opposite, in *parametric classifiers* a model of the distributions is first selected, like the normal density for example. Each class is then represented by the parameters of the PDF curve, which are learned during training stage. Non-parametric classifiers more generally used, because of their lower computational complexity.

Otherwise, conceptually different approaches do exist, without directly considering the features distributions. In such schemes, the goal is to define linear or non-linear *discriminant functions* that best separate the involved classes in the features space. The maximization of a discriminant criterion using least-squares procedure is performed at training stage. Equivalent interpretations of both parametric and discriminant functions approaches can be formulated, as mentioned in [Couv97]. In the same way, the main schemes of the neural network philosophy (*perceptron*, *radial basis* and *Hopfield* networks) can be compared to particular traditional pattern recognizers.

In this text, three types of parametric classifiers will be considered. The sound class features will be successively modeled with: one normal distribution (Bayes classifier), a mixture of several weighted normal distributions (GMM), and using the Hidden Markov Model (HMM). In this last case, several states do exist to model the time evolution, and each state is associated with one normal distribution. Finally, the *multi-layer perceptron* (*MLP*) neural network approach will be considered as a fourth classifier architecture, for comparison.

5.2 Bayesian Classifier

The Bayesian classifier is one of the simplest parametric recognizer [Theo98]. Supposing that a one-dimensional feature x is received from the analysis stage, the recognizer will try to attribute it to one of N_c classes noted ω_i, with $i = 1,\ldots, N_c$. The *Bayes decision rule* [Couv97] says that the feature must be associated with the class that possesses the maximum *a-posteriori* probability, in the purpose of minimizing the global classification error. Let's define the involved probabilities:

- After having observed the feature *x*, the *a-posteriori* probability $P(\omega_i \mid x)$ is the probability that *x* belongs to the class ω_i.

- The *a-priori* probability $P(\omega_i)$ is the probability that the class of the next feature *x* to come out will be ω_i. This parameter can be estimated before use of the system, after considering the number of observation occurrences n_i in each class:

$$P(\omega_i) = \frac{n_i}{\sum_{j=1}^{N_c} n_j} \tag{5.1}$$

- Supposing that the feature to come out will belong to the class ω_i, the probability density function $p(x \mid \omega_i)$ associated to the class ω_i represents the probability for this feature to take the value *x*. $p(x \mid \omega_i)$ is the feature PDF, measured or modeled at training stage.

All those probabilities are connected by the well-known Bayes rule that can be used to determine the *a-posteriori* probabilities of *x* for each class ω_i:

$$P(\omega_i \mid x) = \frac{p(x \mid \omega_i) P(\omega_i)}{\sum_{j=1}^{N_c} p(x \mid \omega_j) P(\omega_j)} \tag{5.2}$$

Then, the Bayes decision rule can be expressed as:

$$\omega(x) = \omega_i \quad \Big| \quad i = \arg\left\{ \max_i P(\omega_i \mid x) \right\} \tag{5.3}$$

or identically, using (5.2) and removing the constant term at denominator:

$$\omega(x) = \omega_i \quad \Big| \quad i = \arg\left\{ \max_i \left[p(x \mid \omega_i) P(\omega_i) \right] \right\} \tag{5.4}$$

Furthermore, in the special case of identical *a-priori* probabilities from one class to the other, it comes:

$$\omega(x) = \omega_i \quad \Big| \quad i = \arg\left\{ \max_i p(x \mid \omega_i) \right\} \tag{5.5}$$

If it is next supposed that the features are normally distributed, with a mean value μ_i and a variance σ_i^2 for each class,

$$p(x|\omega_i) = p_g(x|\omega_i) = \frac{1}{\sqrt{2\pi}\ \sigma_i} \exp\left(\frac{-1}{2\sigma_i^2} \left(x - \mu_i^2 \right) \right) \tag{5.6}$$

then, the gaussian curve can directly be used in (5.4), to determine the membership class of the feature *x*.

The same approach can be done when the extracted features are *D*-dimensional vectors $\vec{x}$. In this case, if each component feature is supposed to have a gaussian distribution, all *D* distributions can be merged in the same joint *multivariate gaussian* PDF:

$$p_g(\vec{x}|\omega_i) = \frac{\exp\left[-\frac{1}{2} (\vec{x} - \vec{\mu}_i)^t C_i^{-1} (\vec{x} - \vec{\mu}_i) \right]}{(2\pi)^{D/2} \sqrt{\det(C_i)}} \tag{5.7}$$

In (5.7), $\vec{\mu}_i$ and C_i are respectively the mean vector and covariance matrix of the i^{th} class, estimated during the training process, using K_i training vectors $\vec{x}_{k,i}$ in the considered class *i*:

$$\vec{\mu}_i = \frac{1}{K_i} \sum_{k=1}^{K_i} \vec{x}_{k,i} \tag{5.8}$$

$$C_i = \frac{1}{K_i - 1} \sum_{k=1}^{K_i} (\vec{x}_{k,i} - \vec{\mu}_i)(\vec{x}_{k,i} - \vec{\mu}_i)^t \tag{5.9}$$

Then (5.4) becomes:

$$\omega(x) = \omega_i \quad \Big| \quad i = \arg\left\{ \max_i \left[p_g(\vec{x}|\omega_i) P(\omega_i) \right] \right\} \tag{5.10}$$

In this expression, a further simplification can be applied, taking the (monotonic) logarithm of (5.7):

$$\arg\left\{ \max_i \left[p_g(\vec{x}|\omega_i) P(\omega_i) \right] \right\} = \arg\left\{ \max_i \ln\left[p_g(\vec{x}|\omega_i) P(\omega_i) \right] \right\} \tag{5.11}$$

Then, introducing the PDF expression (5.7) and removing the constant terms, the member on the right becomes:

$$\arg\left\{\max_i\left[-\frac{1}{2}\left(\vec{x}-\vec{\mu}_i\right)^t C_i^{-1}\left(\vec{x}-\vec{\mu}_i\right)-\frac{1}{2}\ln\left(\det(C_i)\right)+\ln P\left(\omega_i\right)\right]\right\} \quad (5.12)$$

The expression between brackets is the *Bayes discriminant function* $g_i\left(\vec{x}\right)$ to be maximized. Alternatively, minimizing the *Bayes distance* is defined as:

$$\arg\left\{\min_i\left[\left(\vec{x}-\vec{\mu}_i\right)^t C_i^{-1}\left(\vec{x}-\vec{\mu}_i\right)+\ln\left(\det(C_i)\right)-2\ln P\left(\omega_i\right)\right]\right\} \quad (5.13)$$

If the *a-priori* probabilities are equal, the last term is constant and can be dropped. Sometimes, only the first term is minimized, defining the *Mahalanobis distance* [Theo98]. In this section, the Bayes distance will systematically be used, without taking account of the *a-priori* probabilities. This philosophy will be motivated in the following paragraph.

5.2.1 Example

Figure 5.1 shows an example of classification process. The little database (see Chapter 2) is involved: 82 signals representing 34 doors, 29 explosions and 19 bottle breaks have to be classified using (5.13). The features statistics of each class is modeled by a gaussian distribution (5.7), whose parameters have been learned from a training set made of 35 doors, 29 explosions and 19 bottle breaks, all different than the sounds to recognize. The features are extracted during 1 second, as specified in section 4.2: a spectrogram process (method A) is used, with 1/8 overlapping Hanning windowed time-frames constituted of 512 samples. 5 uniform energy bands are considered in each spectrum, between 0 and 15 kHz. The resulting 475 features are concatenated in one big vector, normalized according to section 4.1.1, PCA-transformed with a reduction to 10 decorrelated features, and finally LDA-transformed, to keep only the two most discriminating dimensions with non-zero eigenvalues. The *a-priori* probabilities are not considered in the classification process. Actually, even if the number of signals to classify is not equally distributed between classes in the present experiment, this balance doesn't correspond to the *a-priori* probabilities that will be encountered in real conditions. In such real case, the $P\left(\omega_i\right)$ values could either be measured, or supposed unknown and considered equal. This last hypothesis will be adopted, keeping some

generality in the provided results. When particular values of *a-priori* probabilities are taken into account, slightly better results should be expected.

Figure 5.1 shows the 2-dimensional reduced features space. Each signal to be classified is represented by one mark. Each class model is localized by elliptic contours centered on the mean position. Three ellipses are drawn, corresponding to 1, 4 and 9 times the variance. Boundaries between classes are illustrated, corresponding to the geometric place where two class discriminant functions are equal. Those curves are quadric curves, hyperboles or ellipses. The figure is performed using the Matlab Pattern Recognition Toolbox [Ahle96].

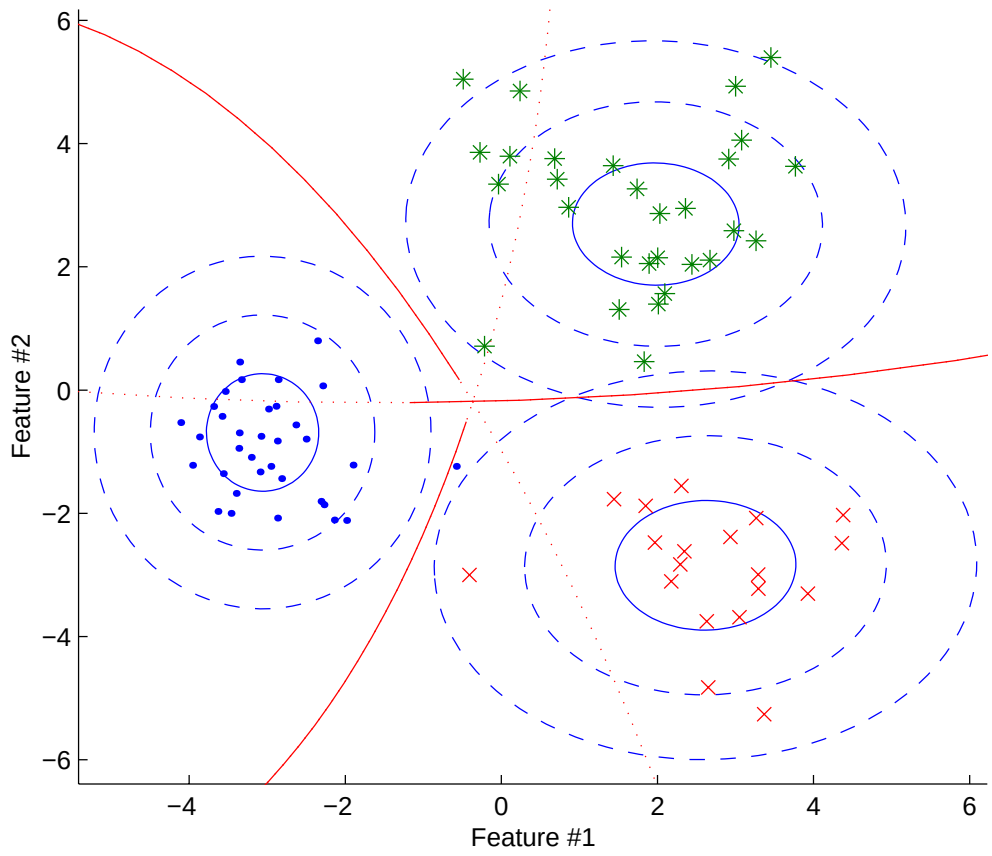


Figure 5.1: *Illustration of a classification process in the features space.*

For each signal, the Bayes distances (5.13) relatively to each class are calculated, and the membership of the signal is attributed to the class with the lower distance. The results can be summarized in the *confusion matrix*. This confusion matrix illustrates the recognition performance in one glance, each

class being associated with one line. If the matrix is diagonal, the correct 100% classification rate is reached. If the non-zero value *A* appears out of the diagonal at index *ij*, it indicates that *A* members of the class *i* were erroneously attributed to class *j*. Here, the confusion matrix for the doors, explosions and bottles breaking sounds is:

$$\begin{bmatrix} 33 & 0 & 1 \\ 0 & 29 & 0 \\ 0 & 0 & 19 \end{bmatrix}$$

In this example, the global classification performance is 98.78%. It can be seen that one signal is positioned just on the other side of the boundary delimiting its class: it's a closing door sound that seems to have properties near to bottle breaks! When examining the three Bayes distance values for each correctly classified signals, one distance always appears much lower (<20%) than the other two ones, indicating the class belonging. On the opposite, for the bad classified signal, two distances are small, and the smaller one is the wrong one. In Chapter 6, such situations will be specially considered, and the signal could be declared as "rejected" because of uncertainty.

5.2.2 Protocols for Performance Measure

In the above example, one signal was wrongly classified, resulting in the performance rate of 98.78%. However, this value depends on the class models built during the training stage, using a subset of the database. If another subset is chosen, the models may be slightly different and other values of performance can be obtained, depending on particularities of the involved signals. Therefore, to prevent the measured performance from being too much sensitive to one particular choice of signals at training, the experience should be re-iterated several times, using different distributions of signals in the *training* and *testing sets*.

Depending on the importance of the available database, different protocols are presented in the literature [Couv97] for measuring the performance of recognition systems. Involving the whole database at training stage would provide most precise models, but the same signals would have to be used for the measures, providing optimistically biased results (*Resubstitution method*). A good solution is the *Leave-one-out method*, involving the whole database at training, one signal excepted. This signal is then used for the measure.

Iterating this scheme until all arrangements are passed provides the most precise results, but the method is highly complex. In the *Cross-validation method*, the database is segmented in several subsets, and an iterative procedure is implemented, each time using one subset for the test and the other ones for training the system.

In this study, a variation of the cross-validation method is used. The database is separated in two equally sized subsets, one being used for training and the other one for testing the model. This operation is iterated several times, each time with a different attribution of the signals in both subsets. The attribution is randomly generated at each new iteration. The number of iterations is generally selected between 10 and 100 times, depending on the complexity of the experiments. Ideally, this number should increase with the size of the database. However practically, when considering the medium or larger databases (see Chapter 2), this becomes impossible because of the mathematical complexity. In all cases, the standard deviation of the iterated results will be mentioned, beside the global recognition rate. Sometimes, when interesting, the whole confusion matrix will be indicated. Thus, in the example of section 5.2.1, the global performance rate becomes 99.02% after 10 cross-validation iterations, with a standard deviation of $\pm 0.96\%$. The confusion matrix is now:

$$\begin{bmatrix} 340 & 0 & 0 \\ 2 & 283 & 5 \\ 1 & 0 & 189 \end{bmatrix}$$

One can observe that the second class, the explosions, seems often confused (particularly with bottle breaks) compared to the other classes, whose recognition rates are nearly 100%. However, even if a tendency appears, one must be very careful before concluding on such few iterations. Thus, if the experiment is continued to reach 50 cross-validations, the global recognition rate decreases to 98.63% and the confusion matrix shows more uniformly distributed errors:

$$\begin{bmatrix} 1678 & 12 & 10 \\ 5 & 1429 & 16 \\ 9 & 4 & 937 \end{bmatrix}$$

The standard deviation increases ($\pm 1.57\%$) due to few bad models providing low recognition rates (93.90% for one models giving 5 bad results!). 20 models gave 100% recognition rates and 16 other models

provides only 1 error. This experiment lets clearly appear the importance of the training set, and the careful choice of the signals required when setting up the training sound samples.

5.2.3 Experimentation with the Little Database (3 classes)

In this sub-section, several experiments with the Bayes classifier will be presented, recognizing impulsive sounds distributed among 3 classes, as specified in Chapter 2. This little database is constituted of rather similar sounds in each class and represents the practical situation where the classifier is adapted to the type of sounds occurring in a specific place (on-site learning). The behavior of the different features extraction schemes described in Chapter 4 (Uniform spectrogram, Bark spectrogram, LPC, CC, MFCC, ear model) will be evaluated and compared, modifying some of their parameters and including features reduction techniques (PCA, LDA). At the end, only the best features schemes will be retained for more difficult classification tasks (6 classes), and for use with advanced architectures (GMM, HMM, NN).

Practically, to prevent numerical problems resulting from zero amplitude samples, the class models are built adding a very small amount of white noise to all the input audio signals, corresponding to 90 dB SNR. The same operation is performed before recognizing any input pulse.

5.2.3.1 Spectrogram features

The first type of features to be considered is the spectrogram. Both schemes A and B (see Section 4.2) are evaluated. In the first case, the spectrogram of the impulsive sound is determined during exactly one second starting from the pulse attack, without eliminating possible silent portions near the end. For the B type, the whole signal is taken into account. Each analyzed time frame is made of N samples (1/8 Hanning overlap), and a uniform frequency decomposition of the energy in N_b spectral bands is performed, from 0 to F_{sup} Hz. The energies are expressed in decibels, and normalized by the maximum value over the whole signal. In method A, the energy values are further concatenated into one large vector, and reduced to a dimension of 10 using a principal component analysis. This PCA reduction is imperative because the original features dimension is much too large, widely exceeding the number of training signals. The value of 10 is obtained with the criterion 4.11. In some experiments, an LDA transformation is further applied to keep the only 2 dimensions with non-zero eigenvalues. Those transformations result in an important increase of mathematical operations during training stage, but lower

dimensions sometimes results in a benefic complexity reduction at testing time. Furthermore, if the transformations succeed in sufficiently removing correlation between features, most terms out of the covariance matrix diagonal becomes very small, and it is possible to work with zero approximations of those terms, leading to faster processing.

Table 5.1 shows the evolution of the recognition performance when varying the number of samples per time frame *N,* i.e. the time resolution of the spectrogram. The measures are considered for F_{sup} = 15 kHz, using N_b uniform bands. To some extent, it can be observed that adding redundancy (smaller *N*) to the original features increases the recognition performance. The final selection of the optimal value for *N* should depend on the targeted complexity of the system. Low resolutions still provide surprisingly good results in method A, for a very low computational complexity. The best value stands around *N* = 512 or 1024 audio samples.

N	256	512	1024	2048	4096	8192	16384
Spectrogram A, N_b = 5, PCA reduction (to 10)	98.27 ± 1.70	97.50 ± 2.47	97.84 ± 2.00	97.20 ± 2.45	97.16 ± 2.97	96.94 ± 2.99	92.45 ± 13.00
Spectrogram A, N_b = 5, PCA (to 10), LDA (to 2)	99.33 ± 1.29	99.18 ± 1.27	**99.39** ± 0.99	98.66 ± 1.40	98.23 ± 1.49	98.41 ± 1.26	97.65 ± 1.35
Spectrogram B, N_b = 5, no reductions	85.17 ± 4.82	98.70 ± 1.46	98.27 ± 1.81	98.32 ± 1.70	98.39 ± 1.36	97.49 ± 1.63	95.29 ± 2.66
Spectrogram A, N_b = 50, PCA (to 10), LDA (to 2)	too complex	99.51 ± 0.93	99.27 ± 1.00	98.23 ± 1.56	97.93 ± 1.59	97.80 ± 1.29	97.87 ± 1.72
Spectrogram B, N_b = 50, no reductions	86.10 ± 5.78	98.90 ± 1.21	99.15 ± 0.82	99.27 ± 0.63	99.27 ± 0.63	98.90 ± 1.07	47.56 ± 19.66
Spectrogram B, N_b = 50, PCA (to 10)	97.68 ± 1.89	99.51 ± 0.73	**99.57** ± 0.72	99.51 ± 0.73	98.72 ± 1.08	98.35 ± 1.14	96.16 ± 2.94
Spectrogram B, N_b = 50, PCA (to 10), LDA (to 2)	97.05 ± 2.10	99.02 ± 0.88	99.29 ± 0.82	98.98 ± 0.91	98.62 ± 1.22	98.02 ± 1.40	95.72 ± 2.68

Table 5.1: *Recognition performance and scattering values (%), when varying the time resolution in the spectrogram. The upper part of the table deals with 5 frequency bands, while the lower part uses 50 bands, to be able to test the reductions on method B. 100 iterations are done for 5-bands values and 20 iterations for 50-bands values.* F_{sup}*=15kHz.*

Let's observe the effect of the features reductions. For method A, the first two lines of the table show that the LDA reduction improves the results

significantly. The increase in performance is more evident when poor results are obtained without applying the reduction. It seems that the LDA transformation performs a stabilization of the results. This can also be observed, looking at the scattering values, which are smaller than without LDA: the influence of bad models is decreased with an LDA reduction. Considering the scattering, the larger the value, the lower the recognition rate.

For method B, the lower part of the table shows that the transformations have smaller effects on the recognition performance, except for very long or very short frames. Globally, the PCA reduction provides a slight enhancement. This result indicates that near optimal features are used for N around 1024, using N_b=50. In fact, the amount of redundancy is smaller when considering individual frames, than with features concatenation. The bad effect of too high redundancy can be observed in method B, for N=256 samples. In this case, the reductions provide interesting improvements. Another effect is illustrated for N=16384. There, the number of frames per signal is very low, and bad results are obtained without reduction, because the number of training samples gets too small compared to the important number of features dimensions (N_b=50).

In the upper part of the table, methods A and B can be compared for the same number of bands (N_b=5 or 50). One can observe that the difference is not very important, provided that the LDA reduction is applied in method A. In such case, the concatenation technique A is slightly better, and seems more appropriate, particularly when the time resolution is lower, that means, when low-computational charge is required. This can be explained by the time evolution, which is taken into account with features scheme A. Surprisingly, the improvement is not important. However, one should note that the signals in the database were not scaled over the time axis, and particular dimensions of the concatenated feature vector do not always correspond to the same part of the impulsive signal. A scaling operation, like the Discrete Time Warping (DTW) used in speech recognition [Rabi93] could improve the results for method A.

In the following experiments, the time resolution will be set to 512 samples per frame, and method B will systematically be used, because of reduced memory requirements and appropriate design for real-time processing. In most experiments, no reduction is applied, even if a slight improvement could be expected from the PCA reduction, for some particular number of bands.

In Table 5.2, the number N_b of frequency bands is varied, keeping a uniform decomposition between 0 and 15 kHz. This table shows an increased

performance when raising the frequency resolution, but the results slightly decrease when too much redundancy appears (50 bands). The optimal performance is obtained using around 15-30 bands. The effect of the PCA and LDA reductions is positive only for non-optimal values of N_b. The reductions decrease the performance around the optimal number of bands (15-30).

N_b	3	5	10	15	20	30	50
Spectrogram B, no reduction	98.12 ± 1.93	98.87 ± 1.31	98.94 ± 0.96	**99.40** ± 0.84	99.23 ± 1.18	99.39 ± 0.70	99.11 ± 0.83
Spectrogram B, PCA reduction to 10, LDA to 2	Not applicable	Not applicable	99.02 ± 1.10	98.30 ± 1.33	98.10 ± 1.46	98.76 ± 1.07	99.18 ± 0.80

Table 5.2: *Recognition performance (%) when varying the number of uniform frequency bands in the spectrogram, between 0 and 15 kHz. N = 512, 100 iterations for each value.*

In Table 5.3, the contribution of higher frequency components in the energy spectrum is evaluated. For that, the upper frequency limit of the spectrum is varied, adapting the number of bands accordingly. The width of each band is kept unchanged: 1000 Hz. In this way, the effect of adding high frequency information can be measured.

F_{sup} [kHz] or N_b	4	8	12	15	18	21
Spectrogram B, no reduction	93.37 ± 2.48	**99.39** ± 0.89	99.06 ± 0.99	**99.39** ± 0.86	99.07 ± 1.01	98.91 ± 0.80

Table 5.3: *Recognition performance (%) when varying the upper frequency limit F_{sup} of the spectrum, keeping fixed bandwidths (1000 Hz) for all bands. N = 512. 100 iterations for each value.*

This table doesn't show an important increase of performance when adding higher frequency components to the features. On the opposite, raising the bandwidth to 18 kHz or 21 kHz decreases the performance, maybe because of noisy component or because of insufficient frequency bandwidth in the recorded database. A surprising good value stands for F_{sup} = 8 kHz. This fact will have to be confirmed, using more classes, with more diversity. Maybe this particular result comes from well-separated spectral properties between classes, below 8 kHz.

5.2.3.2 Non-uniform Bark scale spectrogram features

In the next experiment, the Bark scale (see Appendix B) is used as the frequency bands arrangement. 25 frequency bands are considered, according to [John88]. The superior frequency limit is 21 kHz. Table 5.4 shows that very good recognition power can be attained with the Bark scale decomposition, even when the time resolution is not precise (4096 samples per block).

N	512	1024	2048	4096	8192	16384
Spectrogram B, Bark scale, no reduction	96.01 ± 2.27	99.70 ± 0.53	**99.83** ± 0.49	99.37 ± 0.78	98.73 ± 0.99	93.38 ± 2.72

Table 5.4: *Recognition performance (%) using frequency band features arranged according to the human Bark scale (Appendix B). 100 iterations for each value.*

The optimal window length of 2048 samples provides near 100% performance (99.83%). The confusion matrix in this case is:

$$\begin{bmatrix} 3400 & 0 & 0 \\ 0 & 2900 & 0 \\ 14 & 0 & 1886 \end{bmatrix} \tag{5.14}$$

In fact, the only 14 observations that are confused belong to the glass breaking class, and are interpreted as door slams. All other signals are correctly classified. The individual recognition rates for doors, explosions and glass breaking sounds are respectively 100%, 100%, and 99.25%.

When using other sizes of block length *N*, it can be observed that both larger and smaller blocks provide lower results. In the smaller case, this is explained by the reduced number of spectral points in the lower bands: with 512 samples, only one spectral point is available in the band 0-100 Hz, which is a very poor resolution.

5.2.3.3 LPC features

Table 5.6 shows the recognition performance, using LPC of order 15 for blocks of size *N*. The influence of *N* is not critical, but the best value stands

around 1024, what approximately corresponds to the same duration as used in speech, that is 30ms.

N	256	512	1024	2048
LPC order 15, no reduction	89.67 ± 3.79	88.05 ± 2.68	**90.59** ± 3.84	89.42 ± 5.07

Table 5.5: *Recognition performance (%) using LPC of order 15, for different block sizes N. 100 iterations for each value.*

On the opposite, the number of LPC coefficients must be increased compared to speech, to reach interesting recognition performance, as shown in Table 5.6:

LPC order	15	20	50
No reduction, *N*=1024	90.59 ± 3.84	92.25 ± 3.93	**97.96** ± 1.47
PCA (to 10) and LDA (to 2) reduction, *N*=1024	**97.92** ± 1.01	96.93 ± 0.91	95.13 ± 1.52

Table 5.6: *Recognition performance (%) using LPC of order 15, 20 and 50, for N=1024 samples per block. 100 iterations for each value.*

As it could be expected, LPC are not the best features to model non-stationary impulsive sounds, compared to Bark or uniform spectrogram features. Once again, a reduction process, with PCA and LDA transformations can improve the results for insufficient number of coefficients, but seems not to have positive effects for 50 coefficients.

5.2.3.4 Cepstral features

Table 5.7 shows the recognition performance, using cepstral coefficients of order 20 and 50, for blocks of size *N*. Good recognition results are obtained with 20 or 50 coefficients, particularly for very short frames (256 samples). In the case of 256 samples per block and 50 coefficients, exactly the same confusion matrix as (5.14) is obtained, and only the glass breaking sound class is still confused. However, no specific signal can be clearly identified to be responsible for this situation. It must be noted that this class has few training samples available.

N	256	512	1024
CC 20, no reduction	99.71 ± 0.55	99.63 ± 0.63	99.58 ± 0.64
CC 50, no reduction	**99.83** ± 0.52	99.30 ± 0.95	99.45 ± 0.84
CC 20, PCA (to 10) and LDA (to 2) reductions	99.76 ± 0.49	98.50 ± 1.29	97.91 ±1.09
CC 50, PCA (to 10) and LDA (to 2) reductions	**99.83** ± 0.49	98.99 ± 0.97	98.56 ± 1.13

Table 5.7: *Recognition performance (%) using CC of order 20 and 50. 100 iterations for each value.*

5.2.3.5 Mel-Frequency cepstral features

Table 5.8 shows the recognition performance, using mel-frequency cepstral coefficients of order 20 and 40, for blocks of size 512 and 1024.

N	512	1024
MFCC 20, no reduction	99.45 ± 0.70	99.70 ± 0.54
MFCC 20, PCA (to 10) and LDA (to 2) reductions	-	97.96 ± 1.78
MFCC 40, no reduction	**99.94** ± 0.27	99.91 ± 0.31

Table 5.8: *Recognition performance (%) using MFCC of order 20 and 40. 100 iterations for each value.*

Those results are the best that could be obtained with the little database, and the Bayes classifier. Choosing between 512 or 1024 block sizes is difficult. It seems that cepstral or mel-frequency cepstral coefficients are not too sensitive to block length settings. The reduction scheme is not good in this near-optimal case, what can be explained by noting that MFCC coefficients are known to inherently present very good statistical independence properties.

5.2.3.6 Human-ear model features

The last experiment with 3 classes involves the Meddis Hair Cell model (MHC) of the human ear, as presented in Sub-section 4.4.2. The gammatone

filterbank is designed with 64 or 128 channels between 200 and 22050 Hz, and the firing rates are decimated on the time axis, using a factor of 100 or 200. The 100 times decimation factor gives a time resolution that could be interpreted as time windows of 441 samples duration. A simple FIR all-pole low-pass filter of order 1 is used before decimation, with coefficient –0.99.

The results shown on Table 5.9 are not as good as expected. Probably, better results can be obtained using more advanced ear models (see Section 4.4). A new Matlab toolbox has recently been published [Harm00a], implementing such improved models. The Meddis hair cell model has the benefit of being extensively described in the literature (for both concept and implementation), but it is one of the oldest and simplest models. Nevertheless, its computational and memory requirements are very intensive, and its practical use may be difficult.

N_b	32 channels decim. 200	32 channels decim. 100	64 channels decim. 100
MHC	95.98 ± 1.91	96.34 ± 1.72	**96.71** ± 1.29

Table 5.9: *Recognition performance (%) using a gammatone filterbank (32 and 64 channels) and the Meddis Hair Cell model. No reductions, 10 iterations for each value.*

5.2.3.7 Discussion

As a conclusion for this 3-class study, the following points must be underlined:

- In a clean environment, very good results (near 100%) can be obtained using simple classifier architecture, provided that the intra-class properties are rather similar (on-site learning).
- Spectrogram and cepstral coefficient features give very good performance, especially when considered on a human-like frequency scale (Bark spectrogram and MFCC). The MFCC scheme with 40 coefficients is slightly better, but more computationally intensive.
- LPC features are not appropriate. Maybe some derived method could be better, like Warped LPC [Harm00b] or Line Spectrum Pairs (LSP)

[Kaba86] which performance reveals good for speaker recognition problems [Gras00b].

- PCA and LDA reductions globally improve results when concatenated A-type features are used. In a more general way, the reduction is benefic in case of non-optimal features design (too many or too small number of features) or inappropriate features (for example LPC). However, for good features design (Bark spectrum and CC or MFCC coefficients), the reduction process can decrease the performance.

- The spectrogram A solution (uniform bands) with PCA and LDA reductions slightly outperforms the frame-by-frame procedure B, and its performance is astonishingly good with very few features. However, the memory requirement is much larger, and an important delay is present. As a consequence, the B scheme will be preferred in most of the subsequent experiments.

- In Sub-section 5.2.3.6, the human auditory Meddis Hair Cell model seems not very efficient, considering the performance to complexity ratio. Its implementation is very heavy, taking a lot of time to get classification results. For that practical reason, its behavior won't be studied any further. However, other more advanced models could be considered in future studies, searching for optimized implementation techniques.

5.2.4 Experimentation with the Medium Database (6 classes)

Using the medium database (see Chapter 2), 6 classes are considered and an important diversity is present in each class, turning the classification task in a much more difficult problem. In this section, only the best parameters observed with the little database are considered. Uniform and Bark frequency spectrograms will be further investigated with their best design, as well as cepstral and MFCC coefficients. The behavior of reduced and LPC features will be mentioned for curiosity purpose.

Table 5.10 shows the recognition performance obtained when applying those features extraction techniques to the medium database. The conclusions are slightly different, compared with the well separated 3-class problem:

- Satisfying recognition results (97%) can still be obtained in clean environment with the simple Bayes classifier, despite of the important diversity in each class.

Features Type	Uniform Spec. B N_b=15 [0⟶15kHz]	Uniform Spec. B N_b=50 [0⟶15kHz]	Uniform Spec. B N_b=50 [0⟶20kHz]	Bark Spec. B	CC 50	MFCC 40
Medium Database	92.31 ± 0.72	**97.21** ± 0.72	96.78 ± 0.78	96.70 ± 0.73	95.90 ± 1.93	96.09 ± 0.93

Table 5.10: *Recognition performance (%) for the medium database (6 classes), according to different features schemes. N=1024, 100 iterations.*

- The Uniform spectrogram features outperforms the other schemes when the frequency resolution is high (50 bands). Cepstral and MFCC coefficients seem rather sensitive to the increase of intra-class diversity.

- The Bark spectrogram shows the best recognition performance to complexity ratio, considering the relatively small features dimension (25).

- The total confusion matrix, that leads to the Bark spectrogram performance (96.70%) is:

$$\begin{bmatrix} 15293 & 15 & 104 & 2 & 286 & 0 \\ 7 & 2880 & 2 & 0 & 211 & 0 \\ 95 & 0 & 4282 & 0 & 23 & 0 \\ 9 & 0 & 0 & 3583 & 8 & 0 \\ 123 & 335 & 131 & 0 & 10611 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3000 \end{bmatrix} \begin{matrix} (\textit{Door Slams}) \\ (\textit{Explosions}) \\ (\textit{Glass Breaks}) \\ (\textit{Human Screams}) \\ (\textit{Gunshots}) \\ (\textit{Machines}) \end{matrix}$$

  One can observe that the last class (stable machine sounds) is perfectly recognized. The overall performance without this class becomes 96.44%. Human screams are well discriminated too (99.52%). On the opposite, the second (explosions) and fifth (gunshots) classes are rather confused (92.90% and 94.74% respectively), what is not too surprising. Glass breaking sounds (97.31%) and door slams (97.40%) classifications are generally correct. One can observe that more stationary sounds are easier to classify.

- The overall performance decreases to 64.93% when using 50 LPC coefficients (N=1024). The classification of machine sounds is particularly bad.

- The PCA and LDA reductions are not constructive, as used here. Considering the uniform spectrogram features with 50 bands (97.21%), the application of the PCA reduction (to 10 dimensions) decreases the results to 92.31%, and a further LDA reduction to 5 features (corresponding to the remaining non-zero eigenvalues) gives only 70% of recognition! It seems that the high level of intra-class diversity is the source of these particularly bad results. This confirms that some amount of redundancy in the features do reinforce the recognition capabilities, in case of difficult situations like important diversities inside each class or large number of classes. When reducing the features dimension, the criterion 4.11 still recommends keeping around 10 dimensions after the PCA reduction. On the opposite, the number of LDA non-zero eigenvalues must increase, because the number of axes necessary to discriminate between 6 classes is larger than for 3 classes (let's remember that the LDA transformation tries to optimize an inter-class separability criterion). Therefore, the number of LDA non-zero eigenvalues is more important when more classes are considered (5 against 2 for the 3-class problem). Maybe, another design of the new dimensions could improve the results when considering more dimensions in the PCA reduction (20 or 30), to keep some correlation between features.

- Taking higher frequency information into account improves the results in the following way:

F_{sup} *[kHz] or* N_b	8	15	20	22
Spectrogram B, no reduction	85.37 ± 1.54	92.31 ± 0.72	**94.33** ± 0.95	93.34 ± 1.58

Table 5.11*: Recognition performance (%) when varying the upper frequency limit* F_{sup} *of the spectrum, keeping fixed bandwidths (1000 Hz) for all bands. N=512, 100 iterations for each value.*

The optimum seems to stand around 20 khz. More bandwidth decreases the results, as already observed with 3 classes. On the opposite, this time, choosing 8 kHz as the superior frequency limit is not favorable.

5.2.5 Conclusion

This first study has shown that a basic classifier (with relatively low architectural complexity) can provide interesting recognition performance under clean environment, even in case of important intra-class diversity (97-98%). Of course, properly designed and appropriate features have to be associated with it. Globally, the best features scheme was found to be the simple energy spectrogram, with 50 bands, uniformly distributed between 0 and 20 kHz. However, the Bark spectrogram provides a better compromise, its complexity being much lower (1/2), for a nearly equivalent performance. In case of low intra-class diversity, the MFCC scheme with 40 coefficients provides near 100% performance, but its complexity is larger.

In the next sections, more advanced architectures will be considered, trying to raise the performance in situations with high intra-class diversity or considering more classes. For that purpose, the medium (6 classes) and larger (10 classes) databases will be experimented with Gaussian Mixtures Models (GMM) and Hidden Markov Models (HMM).

5.3 Gaussian Mixtures Model

5.3.1 Introduction

In the previous experiments that involved the traditional Bayes classifier and a simple gaussian model of the features PDF, an interesting observation was done when looking at the *a-posteriori* probabilities. Let's consider one particular explosion signal (represented in Appendix C), and the *a-posteriori* probabilities obtained for each successive frame (spectrogram B). The *a-posteriori* probabilities listed in Appendix C indicate that the signal belongs to the explosion class, because they are globally higher for this class (second line). However, one can remark that at the beginning of the signal, the probabilities are not so clear, and would rather indicate the first class, that is doors.

In fact, the beginning of the explosions, that is the impulsive part of the signal, is often not correctly classified. The reason is that is doesn't look like the longer subsequent stationary part of the signal. As a consequence, using one single gaussian for the explosion PDF seems not sufficient, and leads to a situation where only the preponderant stationary part of the signal is

principally modeled, attributing less importance to the relatively short pulse attack. In other words, the classification is performed on the stationary part of the input signal. This fact not always turns into a problem, but a better solution with an improved PDF model of the signal could perform better in difficult situations, such as confused signals or noisy environment where the pulse release is rapidly disrupted by noise. In this goal, the gaussian mixtures model (GMM) can be considered.

5.3.2 GMM Principles

In the previous sections, the distribution of each feature was simply modeled using one normal curve, and the multivariate gaussian PDF (5.7) was used to integrate all dimensions. However, the Bayes decision rule (5.10) can theoretically be applied to any form of parametric multivariate distributions $p(\vec{x}\,|\,\omega_i)$.

Let's consider a generic parametric distribution $p(\vec{x}\,|\,\omega_i,\theta_i)$, where θ_i represents the parameters of the distribution, like $\theta_{i,g}=\{\vec{\mu}_i,C_i\}$ represents the parameters of a single gaussian PDF. In such a generic case, the derivation of the optimal parameters that model each class is not as straightforward as for gaussian distributions, where (5.8) and (5.9) are simply used. However, one understand intuitively that the training of the class ω_i, that involves K_i training features vectors $\vec{x}_k$, should determine the optimal θ_i that globally maximizes the probabilities for each sample of the training set to come out. This procedure can be mathematically stated and is called the *Maximum Likelihood* procedure [Theo98]. The likelihood to be maximized, function of θ_i, is defined as the product of all probabilities for the training set:

$$L(\theta_i)=\prod_{k=1}^{K_i} p(\vec{x}_k\,|\,\omega_i,\theta_i) \tag{5.15}$$

Then, the optimized θ_i are:

$$\hat{\theta}_i=\arg\left\{\max_{\theta_i}\prod_{k=1}^{K_i} p(\vec{x}_k\,|\,\omega_i,\theta_i)\right\} \tag{5.16}$$

Identically, the logarithm of the likelihood can be maximized, providing some mathematical simplification in case of gaussian PDFs:

$$\hat{\theta}_i = \arg\left\{ \max_{\theta_i} \sum_{k=1}^{K_i} \ln p\left(\vec{x}_k \mid \omega_i, \theta_i\right) \right\} \tag{5.17}$$

All this development is very nice, but the space of possible parameters may be large, and finding the optimal θ_i is sometimes a difficult task. In the following lines, the particular situation of gaussian mixtures distribution (GMM) [Reyn94] will be considered, and a recursive algorithm will be used to find the optimal parameters.

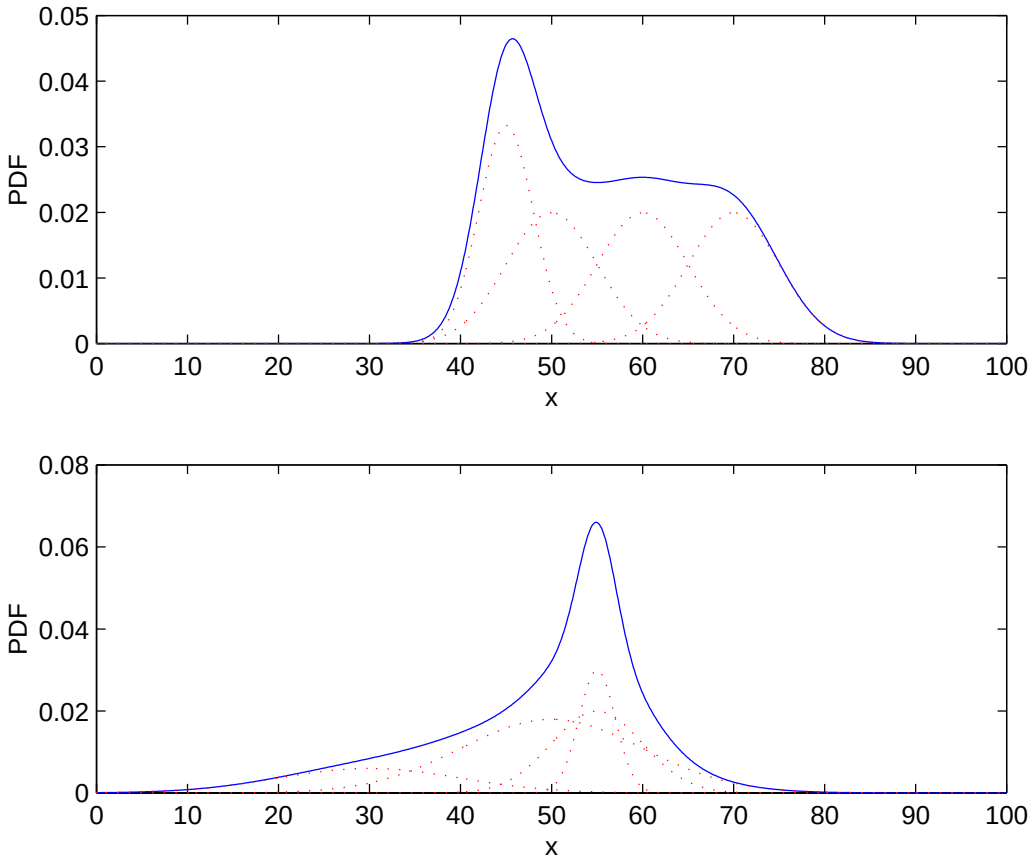


Figure 5.2: *Two examples of GMM density functions (1 dimension), with 4 gaussian components.*

The GMM density function is a weighted sum of *M* multivariate gaussian functions $g_m\left(\vec{x}, \phi_m\right)$, each one with its set of parameters $\phi_m = \left\{\vec{\mu}_m, C_m\right\}$. The associated weighting factor is p_m:

$$p(\vec{x} \mid \omega_i, \theta_i) = p_{gmm}(\vec{x} \mid \omega_i, \theta_i) = \sum_{m=1}^{M} p_m g_m(\vec{x}, \phi_m) \tag{5.18}$$

The M multivariate component functions have the form:

$$g_m(\vec{x}, \phi_m) = \frac{\exp\left[-\frac{1}{2}(\vec{x} - \vec{\mu}_m)^t C_m^{-1}(\vec{x} - \vec{\mu}_m)\right]}{(2\pi)^{D/2}\sqrt{\det(C_m)}} \tag{5.19}$$

In the member on the right of expression (5.18), as well as in the next lines, the class index i is not explicitly mentioned for clarity purpose. Of course, one such mixture PDF is associated with each class ω_i. Therefore, each class can be represented by its set of parameters θ_i or θ:

$$\theta = \{p_1, ..., p_M, \vec{\mu}_1, ..., \vec{\mu}_M, C_1, ..., C_M\} \tag{5.20}$$

The GMM distributions are most interesting because they can theoretically model every kind of statistical behavior, provided that the number of constitutive gaussian components is sufficient (see Figure 5.2). At training time, the parameters θ can be determined using an iterative procedure: the *Expectation Maximization* algorithm (EM).

5.3.3 Expectation Maximization Algorithm

The recursive EM algorithm starts from initial values of θ_i, and provides new values at each step, that increase the likelihood. This procedure can be performed until a gradient threshold is met. The convergence is ensured, but may be local. Sometimes, if the likelihood is too small, other initial parameters may be tried, leading to an improved maximum likelihood.

Let's consider the learning of the parameters θ for one particular sound class ω_i. The density function associated with this class is made of M gaussian components, each one with its own mean, covariance, and weight parameters, μ_m, C_m, and p_m respectively. The goal of the training is to find the optimal parameters for each gaussian, from the available training set of this class. In fact, the gaussians can be considered as M sub-classes ω_m in ω_i, and each features vector $\vec{x}_k$ among the training set of ω_i must come from one of these M sub-classes. In this view, each gaussian can be interpreted as the PDF of one sub-class:

$$p\left(\vec{x} \mid \omega_m, \phi_m\right) = g_m\left(\vec{x}, \phi_m\right) \tag{5.21}$$

and each weight p_m is the *a-priori* probability of the corresponding sub-class. Then, according to the Bayes rule, the *a-posteriori* probability for the training sample $\vec{x}_k$ to come from the m^{th} sub-class is

$$P\left(\omega_m \mid \vec{x}_k, \theta\right) = \frac{p\left(\vec{x}_k \mid \omega_m, \phi_m\right) p_m}{\sum_{m=1}^{M} p\left(\vec{x}_k \mid \omega_m, \phi_m\right) p_m} \tag{5.22}$$

The problem is that no label is known in the training set for the sub-class attribution of each $\vec{x}_k$ (*incomplete data set*). As a consequence, a direct likelihood maximization is not possible in each sub-class, and getting the parameters of each gaussian individually can't be done. Instead, the idea of the recursive EM algorithm is to perform a non-supervised learning, maximizing at each step the *expectation* of the (log-)likelihood function over the M sub-classes, using the whole training set $\vec{x}_1, ..., \vec{x}_{K_i}$, and the parameters $\theta(t)$ available at step t. The expectation of the log-likelihood can be expressed as [Theo98]:

$$\begin{aligned} E\left[\sum_{k=1}^{K_i} \ln\left(p\left(\vec{x}_k \mid \omega_{m,k}, \phi_{m,k}\right) p_{m,k}\right)\right] = \\ = \sum_{k=1}^{K_i} \sum_{m=1}^{M} P\left(\omega_m \mid \vec{x}_k, \theta\right) \ln\left(p\left(\vec{x}_k \mid \omega_m, \phi_m\right) p_m\right) \end{aligned} \tag{5.23}$$

In the left member, $\omega_{m,k}$ should be interpreted as the sub-class from which the current $\vec{x}_k$ is supposed to originate. $\phi_{m,k}$ and $p_{m,k}$ are the corresponding parameters. On the right, this labelling information is not needed anymore, because of the expectation that introduces the probabilities (5.22).

Thus, starting with the current parameters $\theta_{current}$, the maximization of (5.23) with respect to μ_m, C_m, and p_m provides new parameters θ_{new} that increase the likelihood (5.15). For each gaussian component m of the mixture, the recursions of the EM algorithm take the form [Couv97]:

$$p_{m,new} = \frac{1}{K_i} \sum_{k=1}^{K_i} P\left(\omega_m \mid \vec{x}_k, \theta_{current}\right) \tag{5.24}$$

$$\vec{\mu}_{m,new} = \frac{\sum_{k=1}^{K_i} P(\omega_m \mid \vec{x}_k, \theta_{current})\, \vec{x}_k}{\sum_{k=1}^{K_i} P(\omega_m \mid \vec{x}_k, \theta_{current})} \tag{5.25}$$

$$C_{m,new} = \frac{\sum_{k=1}^{K_i} P(\omega_m \mid \vec{x}_k, \theta_{current})\, (\vec{x}_k - \vec{\mu}_{m,new})(\vec{x}_k - \vec{\mu}_{m,new})^t}{\sum_{k=1}^{K_i} P(\omega_m \mid \vec{x}_k, \theta_{current})} \tag{5.26}$$

More detailed information about the EM algorithm can be found in [Demp77] and [Redn84]. A good tutorial is available in [Moon96]. Of course, this training method to be done for each class ω_i, is much more computationally expensive than in the case of a simple gaussian PDF. Therefore, reasonable values for *M* rarely exceed a few tens, what is generally sufficient. Figure 5.2 shows examples of gaussian mixtures with 4 components. In this work, a maximum of eight gaussians will be used.

One important issue concerning the GMM is the choice of good initial parameters θ_{init}, to start the recursion. Even if they can be randomly selected, it seems that better convergence is obtained when performing a prior classification of the training samples, using a clustering technique such as the *Split Vector Quantization* (SVQ) [Rabi93, Capp]. In this study, this well-known iterative procedure will be used as stated below, as starting point of the EM algorithm:

- Given the whole training set $\vec{x}_k$ (k = 1,…, K_i) of the class ω_i, the centroid (mean) vector $\vec{y}_0$ is first determined.

- This centroid vector is then binary *split* in two separate vectors $\vec{y}_1$ and $\vec{y}_2$ such as $\vec{y}_1 = \vec{y}_0(1+\varepsilon)$ and $\vec{y}_2 = \vec{y}_0(1-\varepsilon)$, where ε is the *splitting parameter,* generally chosen between 0.01 and 0.05. Those two vectors constitute the first splitting of the dataset in two categories or *codebooks*.

- All vectors in the training set can then be attributed to one or the other category, in the way to minimize the squared distance $\left|\vec{x}_k - \vec{y}_j\right|^2$, j = 1, 2.

- The balance of all vector attributions is considered, and new centroid vectors $\vec{y}_{1,new}$ and $\vec{y}_{2,new}$ are calculated as the mean of each category members.

- The repetition of the previous two stages is known as the *k-means* algorithm. It has to be performed till the summed squared distance gets lower than a given threshold, or for a specified number of iterations. In this work, 100 iterations are performed. Then, optimized centroids are obtained in the two categories: $\vec{y}_{1,opt}$ and $\vec{y}_{2,opt}$.

- A new binary split can now be applied to each centroid, to get 4 categories. Each split procedure is the same as above, followed by the dataset classification and k-means stages that provides 4 optimal centroids.

- This scheme is reiterated until the number of categories reaches the specified number of GMM gaussians that must be a power of 2.

The above procedure provides the initial means for each sub-class gaussian, as well as the initial covariances, derived when considering the attributions of all vectors in the training set. On the opposite, the initial weights are generally selected either randomly or equal.

In the following sub-sections, the best features extraction schemes, as experienced with the Bayes classifier in Sub-section 5.2.4, are applied to the GMM classification architecture. The EM algorithm involves 20 iterations, which is a good compromise between quality and learning duration. The initial weights are chosen equal. All implementations are based on the Matlab functions provided in [Capp].

5.3.4 Experimentation with the little database (3 classes)

There are not a lot of results to present, when considering the problem of classifying among the 3 classes of the little database. Most recognition performances with the Uniform (50 bands) or Bark spectrograms (method B) and the 40 MFCC coefficients reach 100%, with 2 gaussians. The number of cross-validations is reduced to 10.

However, let's take the occasion of such a simple case to illustrate the form of encountered probability density functions. In the example of Figure 5.3, the Bark spectrogram was used for recognition, with PCA (to 10) and LDA (to 2

dimensions) inside each frame. The covariance matrices are supposed diagonal (no correlations between features). The Figure shows that the space regions attributed to each class can have more advanced forms than just ellipses. In this way, the PDF model is more precise, and the recognition performance increases.

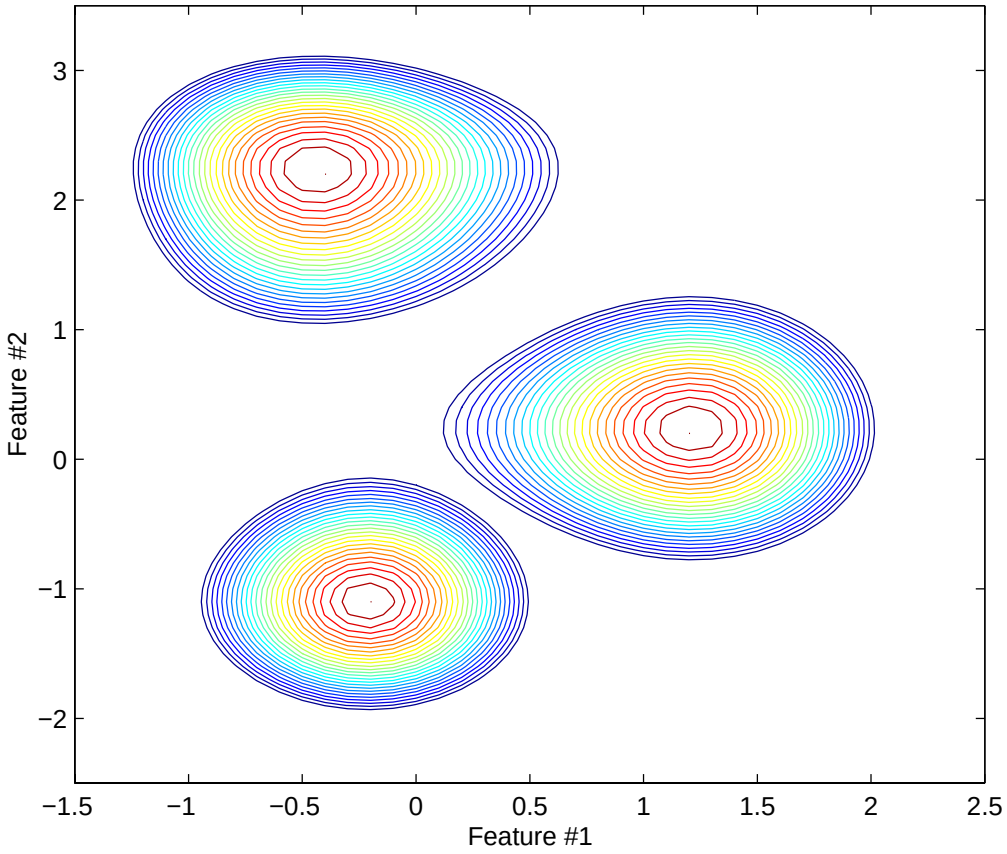


Figure 5.3: *Level-Curves corresponding to 3-class GMM PDF. Each PDF is built with 2 gaussians.*

However, choosing too many gaussian components is not a good idea, because the model becomes sensitive to particular elements of the training set. Indeed, the experiments show that using 4 gaussians for the little database is not adapted, and results in a slight performance decrease. In such case, some particular training samples can be very near to another class border, and will give rise to a rather important PDF component in this region, causing the classification to be more confused than with global models (see Figure 5.4). Furthermore, when using more gaussians, one must be careful about the size of the training database. If this size is too small, not enough training features

are attributed to one or several gaussian component, resulting in non-significant models. In such cases, in a practical point of view, numerical problems rapidly occur in the recognition process, because of unstable models, leading to singular matrices. More interesting illustrations of models with increasing number of gaussians can be found in Appendix D, comparing 1, 2, and 4 gaussians PDF for the same training set.

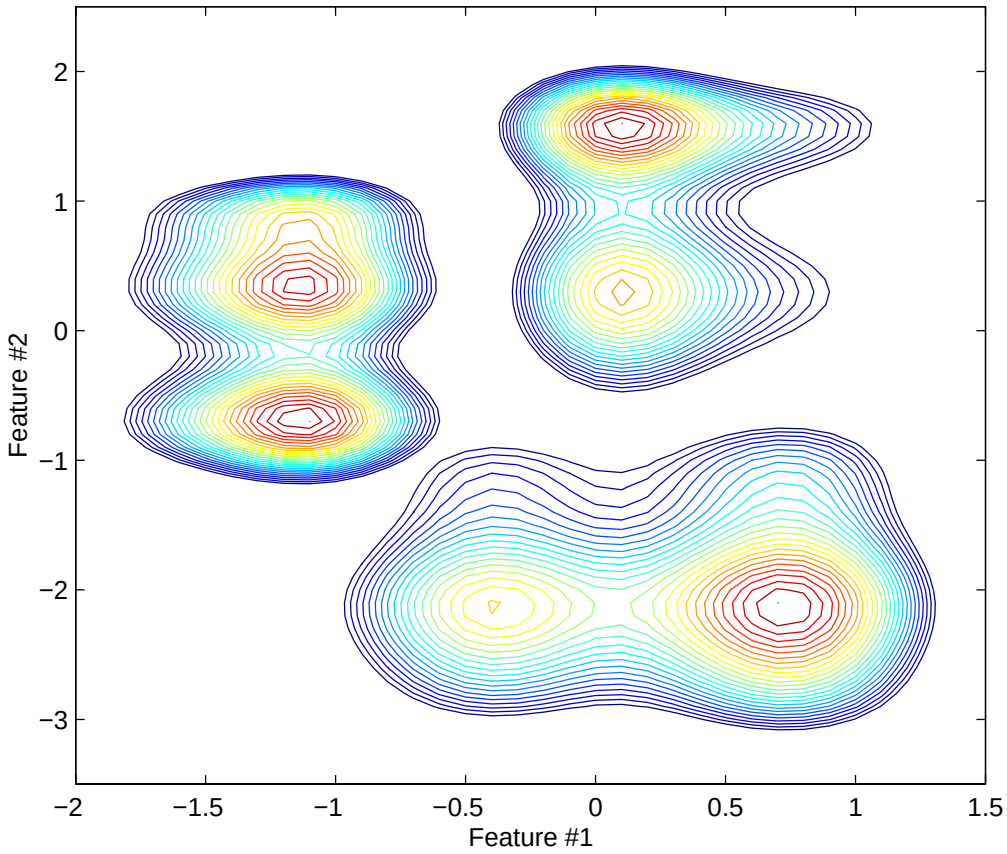


Figure 5.4: *Level-Curves corresponding to 3-class GMM PDF. 4 gaussians.*

5.3.5 Experimentation with the medium database (6 classes)

The situation is more difficult, when working with the medium database and its intra-class diversity. Both 4 and 8 gaussians mixtures are used. The 16 component case had also been considered in previous studies [Besa99], but good models were difficult to find. Table 5.12 shows the obtained results for the uniform and Bark spectrograms, as well as for CC and MFCC coefficients features. No reductions are used.

	4 gaussians	8 gaussians
Uniform Spec. B, N_b=20 (0 -> 20kHz)	94.85 ± 1.69	97.10 ± 1.07
Uniform Spec. B, N_b=50 (0 -> 20kHz)	**97.56** ± 1.16	**97.56** ± 0.68
Bark Spectrogram	96.66 ± 0.80	97.19 ± 0.86
CC 50 coef.	96.48 ± 0.61	95.00 ± 1.21
MFCC 40 coef.	**97.56** ± 1.04	95.85 ± 1.38

Table 5.12: *Recognition performance (%) using GMM classifier. 10 iterations for each value. N=1024 except for MFCC: N=512 samples.*

Those results are slightly better than for the Bayes classifier (see Table 5.10), but the difference is not too impressive. Looking at the results obtained with particular models, one can see that some models lead to particularly good results and some other ones do not. For example, in the case of the 50 band spectrogram with 4 gaussians, rather different recognition rates have been obtained among the 10 iterations: 95.37 or 96.34 but also 97.80, 98.05, 98.54, and 98.78 (two occurrences). Once again, this remark insists on the importance of the training set, even if its size is rather large. Let's remember that the six class models are built from using a total of 412 sounds and that 410 other unknown sounds are tested. Even with such a large database, some models are much better than others. As a practical consequence, several models should be built at training stage, to be tested before use. Then the best model could be selected. This procedure may particularly be advised in case of important intra-class diversity.

5.3.6 Conclusion

For clean environments, the GMM model slightly increases the performance, compared with Bayes models. In the 3-class problem, the 100% recognition rate is reached, using 2 gaussians per class. However, the GMM doesn't succeed in perfectly solving the high diversity of the medium database, even with 8 components. Some particular models show a performance near 99%. In the next section, the HMM model is considered, trying to improve this situation.

5.4 Hidden Markov Models

5.4.1 Principles

In most impulsive types of sounds, the statistics of the acoustic signal changes during the progress of the sound, and it is difficult to model the whole sound with the same density function. In the Gaussian Mixtures, some diversity inside of the same sound class can be integrated in the model, using different gaussian components. Of course, this diversity can be the image of fundamental differences from one signal of the class to other ones. However, it can also be understood as diversity in the same signal, between attack and release moments, for example. To some extent, this makes the GMM a rather good model for time-varying signals.

In this section, another important classifier architecture is considered, to better and more concretely model the time evolution of signal statistics: this is the Hidden Markov Model (HMM) [Rabi89]. In this model, several *states* of the signal are considered, each one with its own statistics (PDF). Furthermore, the transitions between those states are hold in the model. Thus, in HMM, each observation results from a double stochastic process: a *hidden* or underlying process regulates the evolution between (hidden) states, and a second stochastic process generates the observation at each successively encountered state.

The hidden process can be characterized by a *Markov chain* made of S states A_1, …, A_S, with transition probabilities a_{ij} from state A_i to A_j. The order of the Markov chain is 1, meaning that the current transition probability only depends on the originating state, and not on the previous ones.

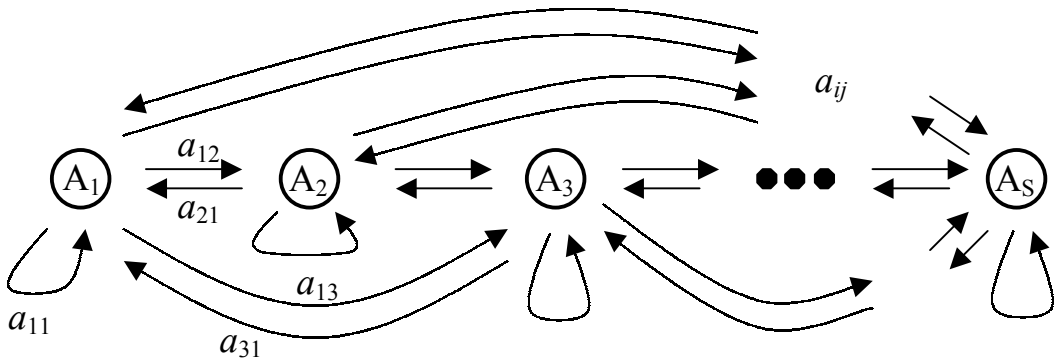


Figure 5.5: *Markov chain*

Two types of Markov chains can be distinguished: in *ergodic* chains, all transition probabilities may be different from 0, allowing any transition between states. On the opposite, in *left-right* Markov chains, the way back to previously encountered states is impossible, and the evolution is only possible in one direction: a_{ij} is 0 if $i > j$. For the problem of impulsive sound recognition, left-right models are well-fitted, due to the important evolution of the signals. On the opposite, when considering more stationary sounds, ergodic models may be more appropriate.

Considering the number of states, 3 states is a good starting point, distinguishing the attack, steady-state and release areas of the impulse. 5 states can encompass possible silent sequence at beginning and end of the signal, especially when the boundaries of the signals are not exactly known. More than 5 states seems not very useful.

In the second stochastic process, a discrete or continuous *state-conditional probability density function* $g_i(\vec{x})$ is associated with each state A_i, to generate an observation vector $\vec{x}$. Discrete density functions are sometimes used, because of their lower complexity and because they do not need as large training sets than continuous PDF. However, their performance is reduced by the vector quantization stage that is required for features discretization. On the opposite, continuous PDF are better, but more computationally intensive. They can theoretically be of any form. However, on complexity reduction purpose, a simple form is generally chosen, as a gaussian curve, or sometimes as a mixture of such gaussians (GMM). If a GMM is used, the number of HMM parameters is important, and problems may be encountered with the size of the training set that should increase accordingly. For that reason, in this work, only continuous gaussian distributions are associated with each state. Thus, each one of the considered sound classes can be modeled with the following parameters:

$$\theta = \left\{ \vec{\mu}_1, ..., \vec{\mu}_S, C_1, ..., C_S, a_{ij}, \pi_1, ..., \pi_S \right\} \tag{5.27}$$

The parameters π_i are the initial probabilities to be in state A_i at the beginning of the sequence. A typical HMM model can be represented as shown on Figure 5.6, for a sequence of features made of T+1 observation vectors $\vec{x}_t \ (t = 0, ..., T)$, generated through a hidden sequence of states $\{z_0, z_1, z_2, ..., z_T\}$. The point is that the same observation sequence, that will be noted $\vec{x}$, can be produced by several different hidden sequences.

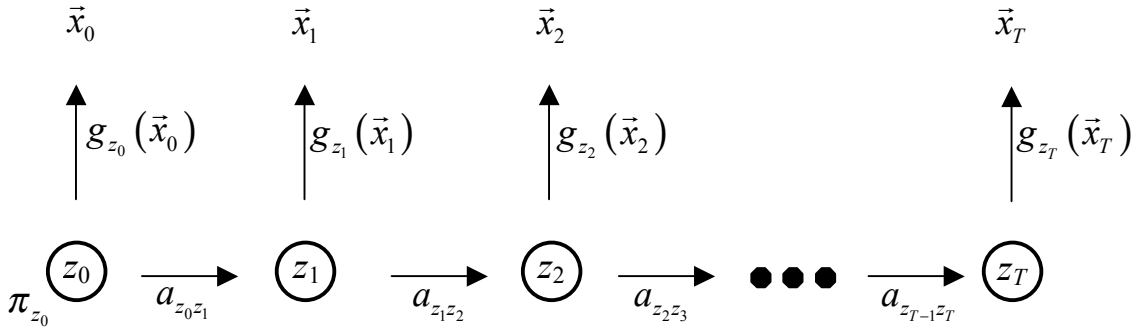


Figure 5.6: *Illustration of the HMM paradigm*

Let's consider the observation of T+1 consecutive features vectors $\vec{x}_t$ that can be collected in one sequence, noted $\vec{\vec{x}} = \{\vec{x}_0, \vec{x}_1, \vec{x}_2, ..., \vec{x}_T\}$. Actually, in the HMM, the probability variable is not the single features vector $\vec{x}$ like in gaussian or GMM models, but a sequence $\vec{\vec{x}}$ of such features. Then, as for other statistical models, the HMM probability density function associated with each class can be expressed in terms of the sequence $\vec{\vec{x}}$, as a function of the parameters θ (the traditional class index ω_i is removed for clarity purpose):

$$p(\vec{\vec{x}},\theta) = \sum_{\substack{All\ possible\ sequences \\ \vec{\vec{z}}=\{z_0,...,z_T\}}} P(\vec{\vec{z}},\theta)\, p(\vec{\vec{x}} \mid \vec{\vec{z}},\theta) \tag{5.28}$$

In this expression, the probability of $\vec{\vec{x}}$ is viewed as the probability to first have one particular hidden sequence $\vec{\vec{z}}$:

$$P(\vec{\vec{z}},\theta) = \pi_{z_0} a_{z_0z_1} a_{z_1z_2} a_{z_2z_3} \cdots a_{z_{T-1}z_T} \tag{5.29}$$

and if so, to produce the observed sequence $\vec{\vec{x}}_k$:

$$p(\vec{\vec{x}} \mid z,\theta) = g_{z_0}(\vec{x}_0)\ \ g_{z_1}(\vec{x}_1)\ \ g_{z_2}(\vec{x}_2) \cdots g_{z_T}(\vec{x}_T) \tag{5.30}$$

Then, summing for all possible state sequences, (5.28) becomes:

$$p(\vec{x},\theta)=\sum_{\substack{\text{All possible}\\ \text{sequences}\\ \vec{z}=\{z_0,\ldots,z_T\}}} \pi_{z_0} g_{z_0}(\vec{x}_0)\ a_{z_0 z_1} g_{z_1}(\vec{x}_1)\ a_{z_1 z_2}\ g_{z_2}(\vec{x}_2)\cdots a_{z_{T-1} z_T}\ g_{z_T}(\vec{x}_T)$$

(5.31)

With this expression, the HMM model can be considered as any other PDF model of the features, and the Bayes recognition theory can be applied all the same: (5.31) can be maximized over all classes to find the most probable class for a given observed sequence $\vec{x}_k$ (as in (5.5)). Furthermore, the maximization of the likelihood (5.17) can be done at training stage, to find the optimal parameters of each class. The only problem is that the exhaustive search through all S^{T+1} possible states sequences in the calculation of (5.31) represents a mathematically highly intensive process. For that reason, efficient or simplified algorithms have been introduced, as the *Forward-Backward* algorithm [Kil96] that is used to determine (5.31) with a number of operations proportional to only $2S^2T$. Often, an approximation of (5.31) is performed, using the *Viterbi* algorithm that requires the same number of operations [Vite67]. For its part, the *Baum-Welsh* recursion [Kil96] (derived from the EM algorithm) is dedicated to the efficient maximization of the likelihood during training. A functional introduction to those algorithms is presented in the following sub-sections, but the reader is invited to consult the appropriate papers for more details. A very good and didactic introduction to HMM is provided in [Pori88], while a complete and detailed presentation of the HMM paradigm is provided in [Kil96], with implementation aspects, training data considerations, and recent improvements on class separability (*discriminant HMM*).

In this work, the implementation of the HMM-related algorithms was performed using the Cappe Matlab toolbox, as a starting point [Capp]. Practically, a particular attention must be attributed to the numerical behavior of the recursions, particularly when the number of parameters increases, and in respect to the size of the training sets. As an illustration, even if the Viterbi algorithm only gives an approximation of the likelihood, it is more easily used practically, because of better numerical behavior than the Forward-Backward recursion.

5.4.2 Forward -Backward Algorithm

This algorithm efficiently determines the probability density (5.31), for an observed sequence $\vec{x}_k$. It is actually made of two different recursions: the *forward* recursion and the *backward* recursion. Both of them deal with partial sequences of features vectors. Under parameters θ, the forward algorithm is interested in the joint probabilities to observe the partial sequence $\{\vec{x}_0, \vec{x}_1, ..., \vec{x}_t\}$, and to reach the hidden state A_i at time t:

$$\alpha_t(i) = p(\vec{x}_0, \vec{x}_1, ..., \vec{x}_t; z_t = i \mid \theta) \tag{5.32}$$

Then, starting at the beginning of the sequence with $\alpha_0(i) = \pi_i g_i(\vec{x}_0)$, and supposing independence between successive vectors (!), the probability to go to the next state A_j at time t+1, while observing the subsequent features vector $\vec{x}_{t+1}$ is:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{S} \alpha_t(i) a_{ij} \right] g_j(\vec{x}_{t+1}) \tag{5.33}$$

At each step t, this probability can be recursively determined for each one of the S potential states A_i, until the last observation is met (t=T). In the end, the total probability (5.31) for the complete sequence $\vec{x} = \{\vec{x}_0, \vec{x}_1, ..., \vec{x}_T\}$ to arise is obtained, summing the last values of α for each possible final state:

$$p(\vec{x}, \theta) = \sum_{i=1}^{S} \alpha_T(i) \tag{5.34}$$

The computational efficiency of this method comes from the accumulation of the past probabilities into the only S previous $\alpha_t(i)$ that enters the determination of the new values $\alpha_{t+1}(j)$. There is no need to consider the whole history of the sequence. Thus, the number of required products is $S + 2T S^2$, compared to the exhaustive search that involves $S + 2T S^{T+1}$ products when pre-calculating the S initial state values. Thus, the dependence to the observation length T is not exponential, but linear.

The backward recursion, in its turn, is interested in the probability to observe the future sequence $\{\vec{x}_{t+1}, \vec{x}_{t+2}, ..., \vec{x}_T\}$, starting from state A_i at time t:

$$\beta_t(i) = p(\vec{x}_{t+1}, \vec{x}_{t+2}, ..., \vec{x}_T \mid z_t = i, \theta) \tag{5.35}$$

The idea is similar, but the iterations start from the end of the sequence and progress toward its beginning. Then, starting at $t = T$ with $\beta_T(i) = 1$, this probability can be recursively derived for each decreasing $t = T\text{-}1, T\text{-}2, \ldots, 0$, using:

$$\beta_t(i) = \sum_{j=1}^{S} a_{ij} g_j(\vec{x}_{t+1}) \beta_{t+1}(j) \tag{5.36}$$

The backward probability is not necessarily needed to obtain (5.31), but it is complementary to the forward probability, when considering the probability for the complete sequence $\{\vec{x}_0, \vec{x}_1, ..., \vec{x}_t, ..., \vec{x}_{T-1}, \vec{x}_T\}$ to be observed and to pass through state A_i at time t:

$$\begin{aligned} &p(\vec{x}_0, \vec{x}_1, ..., \vec{x}_T; z_t = i \mid \theta) = \\ &p(\vec{x}_0, \vec{x}_1, ..., \vec{x}_t; z_t = i \mid \theta)\; p(\vec{x}_{t+1}, \vec{x}_{t+2}, ..., \vec{x}_T \mid z_t = i, \theta) = \\ &\alpha_t(i) \beta_t(i) \end{aligned} \tag{5.37}$$

Consequently, the sequence probability can be expressed from any partial point $t = 0, \ldots, T$, what is necessary during training (see section 5.4.3):

$$p(\vec{x}, \theta) = \sum_{i=1}^{S} \alpha_t(i) \beta_t(i) \tag{5.38}$$

From an implementation point of view, the forward and backward variables $\alpha_t(i)$ and $\beta_t(i)$ are products of probabilities that can get very small when t increases. To prevent numerical problems, each new value $\alpha_t(i)$ can be scaled by the current sum over all S states:

$$\hat{\alpha}_t(i) = \frac{\alpha_t(i)}{\sum_{j=1}^{S} \alpha_t(j)} \tag{5.39}$$

The scaled value $\hat{\alpha}_t(i)$ is used in the recursion (5.33), for determining the next $\alpha_{t+1}(i)$. The same scaling is applied to $\beta_t(i)$, which is supposed to be in the same dynamic range:

$$\hat{\beta}_t(i) = \frac{\beta_t(i)}{\sum_{j=1}^{S} \alpha_t(j)} \tag{5.40}$$

With this scaling, the probability $p(\vec{x},\theta)$ of (5.34) must be modified. If the scaling (5.39) is expressed as $\hat{\alpha}_t(i) = d_t \alpha_t(i)$, it follows:

$$d_t = \frac{1}{\sum_{j=1}^{S} \alpha_t(j)} \qquad or \qquad d_t \sum_{j=1}^{S} \alpha_t(j) = 1 \tag{5.41}$$

then, reaching $t=T$ and considering the T successive scaling steps:

$$d_T \sum_{j=1}^{S} \alpha_T(j) = 1 \qquad or \qquad \prod_{t=0}^{T} d_t \sum_{j=1}^{S} \alpha_T(j) = 1 \tag{5.42}$$

Finally, comparing (5.42) (on the right) with (5.34):

$$p(\vec{x},\theta) = \frac{1}{\prod_{t=0}^{T} d_t} \tag{5.43}$$

The maximization can be advantageously performed over the logarithm of $p(\vec{x},\theta)$:

$$\log p(\vec{x},\theta) = -\sum_{t=0}^{T} \log d_t \tag{5.44}$$

5.4.3 Viterbi Algorithm

Using the Viterbi algorithm, the summation in (5.31) is replaced by its most important component. This is an approximation that can be expressed as:

$$p(\vec{x},\theta) \cong \max_{\substack{All\ possible\ sequences \\ \vec{z}=\{z_0,\ldots,z_T\}}} P(\vec{z},\theta)\, p(\vec{x}\,|\,\vec{z},\theta) \tag{5.45}$$

In other words, only the most probable hidden states sequence is considered, when calculating the probability for the observed sequence $\vec{x}$ to belong to the considered class. Consequently, the problem is to find the optimal hidden states sequence $\vec{z}$ that best corresponds to the observed sequence $\vec{x}$, for each class model θ.

In this goal, the Viterbi recursion is performed: at each new observation $\vec{x}_{t+1}$, and for each candidate state $A_{j,t+1}$ (j = 1,…, S), the principle is to retain the previous state $A_{i,t}$ that is most likely to have lead to the new observation. At the last observation, the state that maximizes the final probability is identified, and a backtracking is performed to select the previous states and form the complete optimal path $\hat{z}$. The algorithm can be stated as follows. It involves $\delta_{t+1}(j)$, the highest probability along a single path, to end at state A_j at time t, for model θ. $\psi_{t+1}(j)$ keeps the indexes of the past retained states $A_{i,t}$ at time t, for every current state $A_{j,\,t+1}$.

- *Initialization*:

$$\delta_0(i) = \pi_i g_i(\vec{x}_0) \qquad i = 1,\ldots,S \tag{5.46}$$

- *Recursions for* $t = 0,\ldots, T-1$:

$$\delta_{t+1}(j) = g_j(\vec{x}_{t+1}) \max_i \left(\delta_t(i) a_{ij}\right) \qquad j = 1,\ldots,S \tag{5.47}$$

$$\psi_{t+1}(j) = \arg\max_i \left(\delta_t(i) a_{ij}\right) \qquad j = 1,\ldots,S \tag{5.48}$$

- *Termination*:

$$\hat{P} = \max_i \left(\delta_T(i)\right) \quad and \quad \hat{z}_T = \arg\max_i \left(\delta_T(i)\right) \tag{5.49}$$

- *Backtracking:*

$$\hat{z}_t = \psi_{t+1}(\hat{z}_{t+1}) \qquad t = T-1,\ldots,0 \tag{5.50}$$

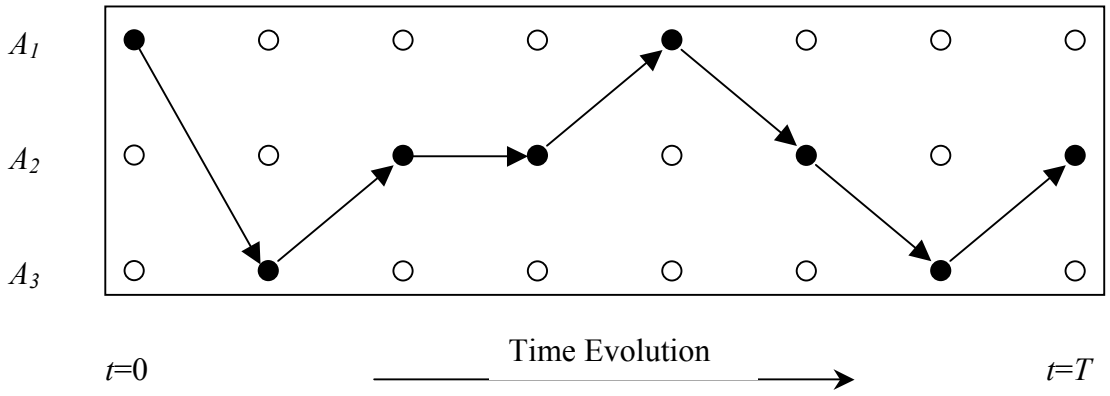


Figure 5.7: *Illustration of an optimal state path in the case of 3-states HMM. The grid represents the three states vertically and the successive observations horizontally. In this example, the optimal sequence* $\hat{z}$ *is* $\{A_1, A_3, A_2, A_2, A_1, A_2, A_3, A_2\}$.

In this work's implementations, the Viterbi estimation of (5.31) is generally used instead of the forward-backward recursion, when classifying a sequence of sound features. However, the determination of the forward and backward variables $\alpha_t(i)$ and $\beta_t(i)$ is needed during the training stage, as specified in the following section.

From an implementation point of view, numerical underflows can be reduced in the Viterbi recursion, using logarithms of the parameters π_i, a_{ij}, *and* $g_i(\vec{x}_t)$ at all stages [Kil96]. This method was used in this study for all HMM classifications.

5.4.4 Baum-Welsh Re-estimation Algorithm

Like for GMM, the training of the HMM is performed in two steps. At first, an initial set of parameters θ is coarsely estimated. Then iterations are performed to determine new parameters that progressively increase the likelihood $L(\theta)$ for the *K* training sequences of the considered class. The log-likelihood to maximize is:

$$\log L(\theta) = \sum_{k=1}^{K_i} \ln p(\vec{x}_k, \theta) \tag{5.51}$$

For continuous left-right HMM with one gaussian PDF associated to each state, the initial parameters can be derived while dividing the training sequences in S time sections. Each section is attributed to one state, starting from A_1 and ending in A_S. Then, for each section, the gaussian statistics of the regrouped features vectors are estimated: $\vec{\mu}_i$ and C_i. The initial probabilities π_i are set to zero, except for the first state, whose probability is 1. The transition probabilities between states i and j can be set to particular values, as shown in the 3-states example below, where the a_{ij} are the component of the matrix A. As for any left-right model, the matrix is triangular superior, with each line summing to 1.

$$A = \begin{bmatrix} 0.4 & 0.5 & 0.1 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.52}$$

For ergodic models, as no temporal structure is supposed, a vector quantization procedure (like for GMM) should be performed to partition the training set in S states, independently of the sequence progression. Then, the PDF statistics in each state can be estimated like for left-right models. The initial and transition probabilities can be taken in a uniform manner for each state, or estimated from the sequence repartition.

Then, as mentioned above, successive re-estimations of the parameters are undertaken in a second step, to increase the likelihood. At first, let's consider only one training sequence $\vec{x}_k$. Then, the searched model $\hat{\theta}$ is aimed to generate the observed sequence with the maximum probability $p(\vec{x}_k, \theta)$. Instead of maximizing this likelihood directly, Baum proposed an approach, similar to the EM algorithm, that maximizes the expectation of a more tractable auxiliary function of both current and new models [Baum70]. This is the Baum-Welsh algorithm that ensures $p(\vec{x}_k, \theta_{new}) > p(\vec{x}_k, \theta_{current})$. The algorithm can be iterated a few times or until some gradient limit is attained. Reaching a local maximum is yet possible, and it is advised to start the recursion from several different initial points. The Baum-Welsh algorithm involves the following probabilities at each step:

- The probability $\gamma_t(i)$ to be at state A_i on time t: (defined using the forward and backward variables, and normalized to sum at 1)

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{p(\vec{x}_k,\theta)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{S}\alpha_t(i)\beta_t(i)} \tag{5.53}$$

- The joint probability $\xi_t(i,j)$ to be at state A_i on time *t,* and at state A_j on time *t*+1 (with similar normalization):

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}\beta_{t+1}(j)g_j(\vec{x}_{t+1})}{p(\vec{x}_k,\theta)} = \frac{\alpha_t(i)a_{ij}\beta_{t+1}(j)g_j(\vec{x}_{t+1})}{\sum_{i=1}^{S}\sum_{j=1}^{S}\alpha_t(i)a_{ij}\beta_{t+1}(j)g_j(\vec{x}_{t+1})} \tag{5.54}$$

After running the forward-backward algorithm to obtain the values of the current $\alpha_t(i)$ and $\beta_t(i)$, that depend on the current parameter set $\theta_{current}$, the re-estimation formula are expressed in function of $\gamma_t(i)$ and $\xi_t(i,j)$:

$$\pi_{i,new} = \gamma_1(i) \tag{5.55}$$

$$a_{ij,new} = \frac{\sum_{t=0}^{T-1}\xi_t(i,j)}{\sum_{t=0}^{T-1}\gamma_t(i)} \tag{5.56}$$

In the case of gaussian density functions, the new statistical parameters are:

$$\vec{\mu}_{i,new} = \frac{\sum_{t=0}^{T}\gamma_t(i)\vec{x}_t}{\sum_{t=0}^{T}\gamma_t(i)} \tag{5.57}$$

$$C_{i,new} = \frac{\sum_{t=0}^{T}\gamma_t(i)\left(\vec{x}_t - \vec{\mu}_{i,new}\right)\left(\vec{x}_t - \vec{\mu}_{i,new}\right)'}{\sum_{t=0}^{T}\gamma_t(i)} \tag{5.58}$$

Similar expressions can be obtained [Kil96] in the more complex case of gaussian mixtures. From a numerical point of view, the scaled forward and backward variables ((5.39) and (5.40)) can be used all the same in the re-estimation formula.

When integrating all the K training sequences of one class, the re-estimation expressions are slightly modified. All sequences are individually considered and $\gamma_t(i)$ and $\xi_t(i,j)$ are calculated for each of them. Then, it can be shown [Rabi89] that the re-estimation can be performed, summing for all sequences in both numerator and denominator of (5.56), (5.57), and (5.58). In (5.55), the mean value over all sequences is simply selected:

$$\pi_{i,new} = \frac{1}{K}\sum_{k=1}^{K}\gamma_{1,k}(i) \tag{5.59}$$

The other expressions become:

$$a_{ij,new} = \frac{\sum_{k=1}^{K}\sum_{t=0}^{T-1}\xi_{t,k}(i,j)}{\sum_{k=1}^{K}\sum_{t=0}^{T-1}\gamma_{t,k}(i)} \tag{5.60}$$

$$\vec{\mu}_{i,new} = \frac{\sum_{k=1}^{K}\sum_{t=0}^{T}\gamma_{t,k}(i)\,\vec{x}_{t,k}}{\sum_{k=1}^{K}\sum_{t=0}^{T}\gamma_{t,k}(i)} \tag{5.61}$$

$$C_{i,new} = \frac{\sum_{k=1}^{K}\sum_{t=0}^{T}\gamma_{t,k}(i)\left(\vec{x}_{t,k}-\vec{\mu}_{i,new}\right)\left(\vec{x}_{t,k}-\vec{\mu}_{i,new}\right)'}{\sum_{k=1}^{K}\sum_{t=0}^{T}\gamma_{t,k}(i)} \tag{5.62}$$

In the following sections, the left-right HMM architecture will be applied to the sound recognition problem, using 3 and 5 states. The best features schemes obtained with Bayes and GMM classifiers will be used. At training stage, 20 iterations of the Baum-Welsh algorithm are performed. The initialization of the transition probabilities is done according to (5.52) for 3 states and according to the following matrix for 5 states:

$$\begin{bmatrix} 0.4 & 0.3 & 0.2 & 0.05 & 0.05 \\ 0 & 0.4 & 0.3 & 0.2 & 0.1 \\ 0 & 0 & 0.4 & 0.3 & 0.3 \\ 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

5.4.5 Experimentation with the little database (3 classes)

Like for the GMM architecture, the results obtained with 50 uniform bands and 25 bands Bark spectrograms reach 100% recognition rate, using 3 states. Using 5 states would not be appropriate in a clean environment.

5.4.6 Experimentation with the medium database (6 classes)

	3 states	5 states
Uniform Spec. B, N_b=20 (0 -> 20kHz)	95.51 ± 0.94	95.91 ± 0.97
Uniform Spec. B, N_b=50 (0 -> 20kHz)	97.10 ± 0.54	96.98 ± 0.70
Bark Spectrogram	96.66 ± 0.67	97.52 ± 0.47
CC 50 coef.	96.30 ± 1.44	95.73 ± 0.86
MFCC 40 coef.	96.66 ± 0.83	96.22 ± 1.21

Table 5.13: *Recognition performance (%) for the medium database, using left-right HMM classifier. 10 iterations for each value, N=1024.*

Those results are globally the same as the GMM's. A few models are satisfying (98.54%), but other ones are not. The above table can not clearly decide for 3 or 5 states. In fact, using only 10 cross-validation iterations seems not to be sufficient to provide precise results. Unfortunately, when using important databases, HMM and GMM recognition simulations are very intensive jobs, and performing more iterations is difficult.

5.4.7 Experimentation with the larger database (10 classes)

Let's now consider the larger database. Compared to the 6-class set, four more sound types are added: dog barks, phone rings, low bangs and children voices. Those added classes possess a lower diversity compared to the other ones. However, of course, different dogs or types of barking are encountered, as well as electronic or mechanical phone rings. The following table compares the results obtained with HMM (3 states), GMM (4 components) and Bayes classifiers, using Bark spectrogram features, with N=1024.

Bayes	GMM 4 comp.	HMM 3 states
90.34 ± 0.82	97.02 ± 0.80	97.19 ± 0.55

Table 5.14: *Recognition performance (%) for the larger database (10 classes), using Bayes, GMM (4 components) and 3 states left-right HMM classifiers. 10 iterations for each value, with the Bark spectrogram. N=1024.*

Those results are quite encouraging, because no important performance decrease is noted for GMM and HMM recognizers, compared to the medium database. An example of confusion matrix obtained after 10 iterations in the HMM case (3 states) is illustrated below, corresponding to 97.19% of good recognition. Some models were observed to reach 98.2%. This matrix shows which classes have important diversity, or which pairs could easily be confused. As an example, human screams are sometimes interpreted as children voices and explosions can be perceived as gun shots, what is not surprising.

$$\begin{bmatrix} 1537 & 4 & 9 & 0 & 15 & 0 & 0 & 0 & 0 & 5 \\ 2 & 279 & 0 & 0 & 29 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 428 & 0 & 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 348 & 0 & 0 & 0 & 0 & 0 & 12 \\ 25 & 10 & 5 & 0 & 1080 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 300 & 0 & 0 & 0 & 0 \\ 2 & 0 & 2 & 9 & 0 & 0 & 252 & 0 & 0 & 5 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 243 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 430 \end{bmatrix}$$

Unlike GMM or HMM architectures, the limitation of the single gaussian PDF classifier appears in Table 5.14. The values show that the Bayes classifier is sensitive to the increase of class number, and its performance drops down to 90%, instead of 96% for 6 classes. In fact, this is a property that can be observed more generally in other applications. In [Gras00], the same GMM algorithms have been applied to the speaker recognition problem, involving 430 speaker classes. As a result, the performance of the GMM (16

components) were rather satisfying, despite of the huge number of classes. The single gaussian model wouldn't have supported such a difficult situation.

5.4.8 Conclusions

The HMM classifier doesn't significantly improve the recognition rate (97-98%), compared to the GMM solution. Furthermore, its mathematical charge is higher at recognition time. However, only one gaussian was used in each state. Maybe better results could be expected at the price of increased complexity, using mixtures of gaussians in each state, with 2 or 4 components. In such case, care should be taken about the size of the training database that may have to increase because of the higher number of parameters per class.

In another way, ergodic HMM could be interesting too. Preliminary studies published in [Besa99] had shown that a slight increase could be expected.

5.5 Neural Network Classifier

Inspired from the brain structure of neurons, the (artificial) neural network (NN) theories are widely applied in diverse domains of application. They mainly provide efficient ways of performing pattern recognition, but they can be used in many other tasks, like implementation of mathematical functions (*2-layer Perceptron*) or signal processing operations (*ADALINE network* for adaptive filtering). Since years, a lot of network architectures have been developed, each of them being better adapted in some specific situation. For example, some architectures include recursive possibilities (like the *Hopfield network*) and can model dynamic systems. Thus, pattern recognition with unsupervised learning is possible (*Competitive Networks*). In other problems involving static pattern recognition, non-recursive networks like the well-known *Multi-layer Perceptron (MLP)* provides non-linear boundaries between classes. During past years, important effort have been dedicated to find efficient algorithms for the implementation and practical use of neural networks, especially concerning the highly complex training algorithms. Because of the parallel structure of the networks, their implementation on parallel architectures is adequate, motivating specific computers to be constructed.

In this text, experiments with one of the most popular network, the Multi-Layer Perceptron, will be applied to the sound recognition problem. The next section is concerned with this particular network, which is well adapted to the pattern recognition problem, when a supervised learning is possible. More information about Neural Networks can be found in [Haga95], presenting a very didactical and complete introduction to the domain, with extended examples and *Matlab* illustrations.

5.5.1 Perceptron Neural Network

The mathematical model of one artificial neuron can be illustrated as on Figure 5.8:

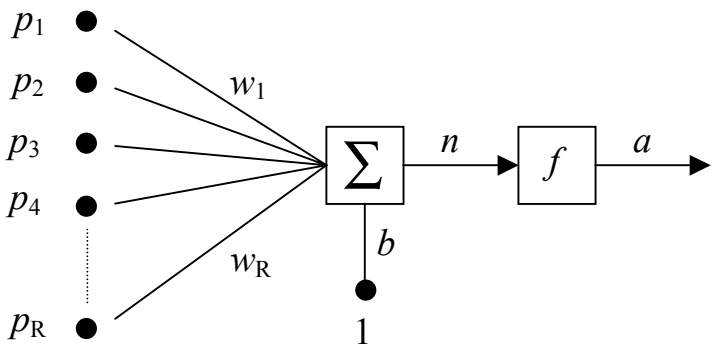


Figure 5.8: *Single neuron with multiple inputs.*

This figure represents a neuron with R inputs (p_i), the associated *weights* (w_i), and the *bias* internal parameter (b), excited by a unit input. The output n of the summation is the result of a linear combination of all the inputs. Then, the function f represents the non-linear part of the model, providing a as the neuron output. Most of the used functions $f(n)$ saturate to 1 when n increases and get to 0 (or –1) when n decreases below 0. The transition occurs at $n = 0$. Examples are the so-called log-sigmoid $a = 1/(1+\exp(-n))$, the hard limit ($a = 0$ *for* $n \le 0$, *and* $a = 1$ *for* $n > 0$), or a saturating linear curve [Haga95]. The global transfer function is:

$$a = f\left(\sum_{i=1}^{R} w_i p_i + b\right) \tag{5.63}$$

In the biological model of the neuron, the inputs correspond to connections (*synapses*) from other neurons, and the output is the produced reaction of the neuron to all incoming excitations, according to an internal chemical process that adds and thresholds the input electrical stimuli.

When several neurons are considered, each one being connected to the same set of inputs, one *layer* of neurons is constituted. In the case of S neurons, and according to a widely adopted representation, the layer can be illustrated as:

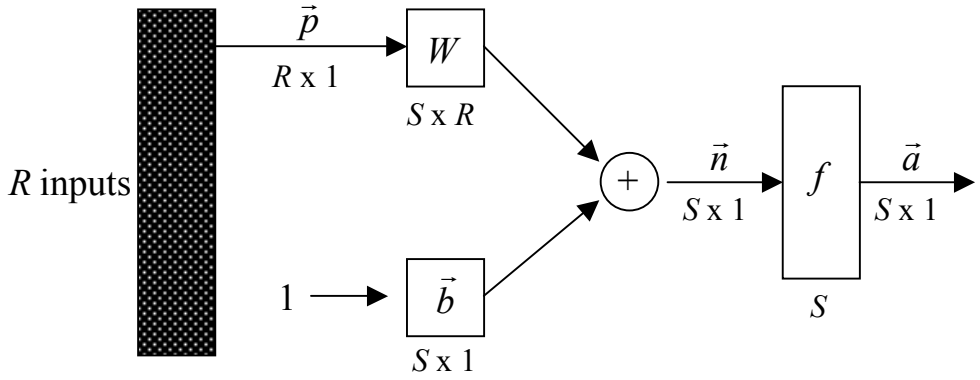


Figure 5.9: *One layer of S neurons. The dimension of the involved vectors and matrices is indicated.*

In this representation, all the weights are grouped in one matrix W of size $S \times R$, and all other parameters are placed in vectors. Generally, the function is identical for each neuron. In this way, the equation of the layer can be stated as:

$$\vec{a} = f(W\,\vec{p} + \vec{b}) \tag{5.64}$$

This is the perceptron, in which the hard limit function f is traditionally used, to provide only binary outputs. In pattern recognition problems, the perceptron can be used to classify R-dimensional input features among 2^S classes, each class being represented by one particular binary output vector. To do this, a supervised learning must be performed, to determinate the parameters of the neurons. Labeled inputs are successively presented to the network, and the output is calculated. The parameters are recursively modified, until correct outputs are obtained, indicating the correct class for each input vector. The *Perceptron Learning Rule* is used for this parameter

learning [Haga95], minimizing the error between the output $\vec{a}$ and the target output $\vec{t}$.

The basic single-layer perceptron, as illustrated in Figure 5.9 is limited because only linear boundaries can be produced between regions of the features space corresponding to classes. This fact can be observed for each neuron of the layer, from equation (5.63): applying $n = 0$, which corresponds to the hard limit function boundary, gives an expression with linear combination of the inputs:

$$\sum_{i=1}^{R} w_i p_i + b = 0 \tag{5.65}$$

This limitation can be avoided, using several connected layers of neurons, leading to the Multi-Layer Perceptron (MLP). The principle is to take the S_1-dimensional output $\vec{a}_1$ of a first layer constituted of S_1 neurons, and to provide it as the input ($\vec{p}_2$) to a second layer, constituted of S_2 neurons:

$$\vec{p}_2 = \vec{a}_1 \tag{5.66}$$

The S_2-dimensional output $\vec{a}_2$ of the second layer can be the final network output (2-layer), or can be once again transmitted to a third layer, as input. Practically, more than three layers are not useful. A standard notation for describing the number of involved neurons and layers in a MLP is $R - S_1 - S_2 - S_3$. The recursive equations of the MLP, for M layers are:

$$\begin{aligned} &\vec{a}_0 = \vec{p} \\ &\vec{a}^{m+1} = f^{m+1}(W^{m+1}\,\vec{a}^{m} + \vec{b}^{m+1}) \qquad m = 0, 1, ..., M-1 \\ &\vec{a} = \vec{a}^{M} \end{aligned} \tag{5.67}$$

In this case, the training is more complex, and a specialized algorithm, the *backpropagation* was developed, recursively minimizing a performance index (that can be the mean-squared error between the output $\vec{a}$ and the target $\vec{t}$), for several labeled inputs. Details about this algorithm can be found in [Theo98]. In the next section, a functional approach of the backpropagation, which is used in the experiments, is presented.

From a practical point of view, the design of the MLP is not straightforward. In a 2-layer network, the number of neurons in the second layer (*output layer*) generally corresponds to the number of involved classes, and each output represents one class. In this way, the classification provides a

"1" output for the selected class, and "0" values for the other ones. In the first layer (*hidden layer*), the number of neurons should be guessed, depending on the problem to solve. For complex pattern recognition problems, the final choice often depends on trial and errors experimentation.

Choosing a 3-layer network is not always useful and must be reported to the increase of complexity. Once more, the number of neurons to be used in the added hidden layer (second hidden layer) is up to the experiment, but the "curse of dimensionality" should be kept in mind: networks can get "overfitted" to their training set, when they are trained with too few examples, compared to the number of available internal parameters. Like in traditional pattern recognizers, the network becomes most sensitive, and a lack of generalization is observed, causing erroneous results for inputs not present at training stage. As a simple rule, the number of network parameters should be lower than the number of training features. However, on the extreme opposite, too many training patterns can lead to difficulties for convergence of the training algorithms.

Practically, the binary hard limit function is replaced by a log-sigmoid, providing some refined information in the output, near the boundary $n = 0$. This can be useful to evaluate the *certainties* of classification. However, the main reason to do this is a differentiation problem that occurs in the backpropagation algorithm.

5.5.2 Backpropagation

The goal of the backpropagation algorithm is to determine the network parameters that best fit the training set, constituted of pairs $(\vec{p},\vec{t})$. In most pattern recognition problems, the target vector $\vec{t}$ is chosen to be filled with zeros, except at the dimension that corresponds to the class label of the input: this element contains a unit value. Then, more concretely, after decided for one function f^m in each layer *m*, the weights $w_{i,j}^m$ and biases b_i^m corresponding to each neuron *i* and input *j* must be derived, minimizing the classification error $\vec{e}$ for each training pair $(\vec{p},\vec{t})$:

$$\vec{e} = \vec{t} - \vec{a} \tag{5.68}$$

Globally, considering all the whole training set, a so-called *performance index* or *cost function J* can be chosen, to be minimized. For example, the cost function can be the mean squared error (MSE) or the sum-squared error (SSE)

over all output neurons and summed for all L available training patterns. Using the SSE case:

$$J_{SSE} = \sum_{l=1}^{L} J_l = \sum_{l=1}^{L} \left(\sum_{i=1}^{S_M} \left(t_{i,l} - a_{i,l} \right)^2 \right) \tag{5.69}$$

The minimization is performed using iterative optimization techniques [Haga95]. One appropriate method in this situation is the *Gradient Descent* or *Steepest Descent* algorithm [Theo98]. In this technique, starting with small initial random values, new parameters (iteration k+1) are obtained by adding the current parameters (iteration k) and a value that is proportional to the opposite gradient of the cost function. Thus, the new parameters move closer to a (potentially local) minimum of the cost function J.

$$\begin{aligned} w_{i,j}^m(k+1) &= w_{i,j}^m(k) - \alpha \frac{\partial J_{SSE}}{\partial w_{i,j}^m} &&= w_{i,j}^m(k) - \alpha \sum_{l=1}^{L} \frac{\partial J_l}{\partial w_{i,j}^m} \\ b_i^m(k+1) &= b_i^m(k) - \alpha \frac{\partial J_{SSE}}{\partial b_i^m} &&= b_i^m(k) - \alpha \sum_{l=1}^{L} \frac{\partial J_l}{\partial b_i^m} \end{aligned} \tag{5.70}$$

The *learning rate* α determines the speed of convergence towards the minimum. Its selection is difficult, because the convergence may be slow in case of little value, but on the opposite, when α is too high, oscillations can appear, sometimes leading to divergence. Optimization methods do exist to adapt α during the recursion. Typical fixed values are between 0.01 and 0.1.

The key problem of the backpropagation algorithm is the determination of the gradients in (5.70). Those gradients can be computed for each training data l, using the differentiation *chain rule*, putting the arguments n_i^m of the neuron functions to the fore. For reading convenience, the index l is removed:

$$\frac{\partial J}{\partial w_{i,j}^m} = \frac{\partial J}{\partial n_i^m} \frac{\partial n_i^m}{\partial w_{i,j}^m} \quad \text{and} \quad \frac{\partial J}{\partial b_i^m} = \frac{\partial J}{\partial n_i^m} \frac{\partial n_i^m}{\partial b_i^m} \tag{5.71}$$

Then, taking account of

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = \frac{\partial}{\partial w_{i,j}^m} \left(\sum_{j=1}^{S_{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m \right) = a_j^{m-1} \quad \textit{and} \quad \frac{\partial n_i^m}{\partial b_i^m} = 1, \tag{5.72}$$

the only terms that remain to evaluate are the so-called *sensitivities:*

$$s_i^m = \frac{\partial J}{\partial n_i^m} \tag{5.73}$$

Here, the name *backpropagation* is explained, because the sensitivities are derived using a backward recursion, starting from the last layer M and going back to the first one. The starting point for the algorithm is expressed in the following equation, requiring derivability properties of the neuron function:

$$\begin{aligned} s_i^M &= \frac{\partial J}{\partial n_i^M} = \frac{\partial J}{\partial a_i^M}\frac{\partial a_i^M}{\partial n_i^M} = \frac{\partial}{\partial a_i^M}\left(\sum_{i=1}^{S_M}(t_i - a_i)^2\right)\frac{\partial f^M\left(n_i^M\right)}{\partial n_i^M} = \\ &= -2(t_i - a_i^M)\,\dot{f}^M\left(n_i^M\right) \end{aligned} \tag{5.74}$$

Then, when decreasing m, a recursive expression from the current layer (m) to the previous one (m-1) can be constructed, through the chain rule:

$$s_i^{m-1} = \frac{\partial J}{\partial n_i^{m-1}} = \sum_{ii=1}^{S^m}\frac{\partial J}{\partial n_{ii}^m}\frac{\partial n_{ii}^m}{\partial n_i^{m-1}} = \sum_{ii=1}^{S^m} s_{ii}^m \frac{\partial n_{ii}^m}{\partial n_i^{m-1}} \tag{5.75}$$

The summation is required because all neurons ii in the current layer depend on the i^{th} neuron of the previous layer. Replacing

$$\begin{aligned} \frac{\partial n_{ii}^m}{\partial n_i^{m-1}} &= \frac{\partial}{\partial n_i^{m-1}}\left(\sum_{j=1}^{S_{m-1}} w_{ii,j}^m a_j^{m-1} + b_{ii}^m\right) = \\ &= \frac{\partial}{\partial n_i^{m-1}}\left(\sum_{j=1}^{S_{m-1}} w_{ii,j}^m f^{m-1}\left(n_j^{m-1}\right) + b_{ii}^m\right) = w_{ii,i}^m\,\dot{f}^{m-1}\left(n_i^{m-1}\right) \end{aligned} \tag{5.76}$$

in (5.75), gives the final recursion:

$$s_i^{m-1} = \left[\sum_{ii=1}^{S^m} s_{ii}^m w_{ii,i}^m\right]\dot{f}^{m-1}\left(n_i^{m-1}\right) \tag{5.77}$$

In summary, after initialization of all weights $w_{i,j}^m$ and biases b_i^m, the backpropagation algorithm requires the four successive operations listed below, at each iteration k:

- Calculation of the network outputs (forward propagation using (5.67)) for all inputs available in the training set. By the way, storage of all n_i^m nodes for later determination of the sensitivities.

- Calculation of the cost function J_{SSE} , using (5.69).
- Back propagation of the sensitivities (using (5.74) and (5.77)) to determine the gradients of the cost function for each training sample.
- Update of the parameters, using (5.70).

This is the *batch mode* of the algorithm in which all training patterns are presented at each iteration. Another solution is to update the weights after each new training sample: *pattern mode*. In this case, the quality of gradient approximation is lower, but more randomness appear during training, and the algorithm has more chance to jump out of a local minimum, especially when the order of presentation of the training samples is not cyclic.

The termination of the algorithm can be done when the cost function gets below some fixed threshold, or when the gradient becomes small. Unfortunately, the convergence of the backpropagation algorithm is very slow, and different solutions to fasten the procedure were proposed in the literature. One simple solution is to use variable learning rates, adapting α to the variations of squared error. Sometimes, the *momentum* method is used: in this case, the correction factor of the update (5.70) is not only proportional to the current gradient, but also depends on the previous gradient. This technique allows reducing the potential oscillations of the gradient that slow down the convergence. Typical values for the momentum factor μ are between 0.1 and 1. (5.70) becomes:

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) + \Delta w_{i,j}^{m}(k+1) \tag{5.78}$$

with:
$$\Delta w_{i,j}^{m}(k+1) = -\alpha \frac{\partial J_{SSE}}{\partial w_{i,j}^{m}} + \mu \, \Delta w_{i,j}^{m}(k) \tag{5.79}$$

Otherwise, in more important modifications, the gradient descent algorithm can be replaced by the *Conjugate Gradient* method (faster convergence), or by the *Newton* method (leading to the very fast *Levenberg-Marquardt* algorithm). Those more efficient solutions are documented in [Haga95]. In this work, 2 and 3 layer perceptrons are experimented with the Neural Network Toolbox for Matlab [Demu98]. The experiments use the batch mode, with momentum and adaptive learning rate or Levenberg-Marquardt implementations of the backpropagation algorithm. No important differences were observed between those two solutions, which correspond to the best compromise between acceptable memory requirements, fast processing, and recognition performance. A lot of other implementations do exist, as listed

and described in [Demu98]. The optimal choice is not easy to do and depends on the configuration of the particular problem (number of classes, network parameters, size of the training set, …). The pattern mode gives very bad results.

5.5.3 Experimentation with the little and medium databases

Let's consider at first the 3-class problem. The Bark spectrogram features are principally considered, as they represent the highest quality-to-complexity ratio. The network is trained using the SSE cost function, a log-sigmoid function in each layer and the momentum method with adaptive learning rate. The learning rate increasing and decreasing ratios are 1.05 and 0.7 respectively, and the momentum factor is fixed to μ=0.95. The backpropagation is stopped either when the sum-squared error gets below 0.01, or if the gradient is lower than 10^{-16}. The number of neurons in the hidden layer is fixed to 30 for 2 layers and if 3 layers are used, 10 and 30 neurons are used in the first and second hidden layers respectively. All those values were designed using a trial-and-error procedure, directly on the impulsive sound database.

MLP	2 layers	3 layers
Bark Spec. Type A PCA (10), LDA (2)	98.95 ± 1.41	97.68 ± 1.86
Bark Spec. Type B No reductions	97.95 ± 1.37	97.84 ± 1.00

Table 5.15: *Recognition performance (%) using MLP classifier with the little database (3 classes). N=1024, 100 iterations for each value.*

The table shows that the performance is globally lower than with other classifiers. Using 3 layers is not worth. The concatenation method A seems better than the B technique. However, when having a look to the confusion matrices, one can observe that in the frame-by-frame procedure B, the recognition rates are 100% for all classes except for the bottle breaks (91.15%):

$$\begin{bmatrix} 3400 & 0 & 0 \\ 0 & 2900 & 0 \\ 141 & 27 & 1732 \end{bmatrix}$$

It turns out that the network has great difficulties to deal with this class. This phenomenon was not observed for other classification architectures, and it is also not observed for the concatenated features which confusion matrix is more balanced:

$$\begin{bmatrix} 3365 & 6 & 29 \\ 9 & 2886 & 5 \\ 36 & 1 & 1863 \end{bmatrix}$$

Using another features scheme, like 50 cepstral coefficients, the same observation can be done, with accentuated performance degradation:

$$\begin{bmatrix} 3399 & 1 & 0 \\ 0 & 2900 & 0 \\ 726 & 1 & 1173 \end{bmatrix}$$

Here, the bottle breaks are really confused and interpreted as doors (61.73%). It's interesting to see that the inverse is not true. It seems that the door model takes a very large domain in the features space, widely overlapping with the more restricted bottle breaks region. The scattering of the results is high (±6.14%), some models reaching 98% of global recognition rate, and other ones only 80%. This reinforces the above interpretation, some particular outlying doors being responsible for the confusion. As another confirmation, substituting doors with dog barks, the results become very satisfying, reaching 98.78%. On the opposite, when replacing the bottle breaks class with the dog barks, the results are bad again for all classes except doors. This shows that when some specific class is difficult to recognize, investigations should be focused on the other classes.

As a first impression, compared to other classifiers, it seems that the MLP neural network is much more sensitive to the classes involved, as well as to the particular signals of the training database. Great care must also be devoted to the features type, and particularly to the features scales: forgetting to normalize the features dynamic turns into a disaster. Practically, with method B, the learning procedure is very slow: the SSE performance goal had to be relaxed to 0.1, because otherwise, the backpropagation algorithm never converges.

In fact, exceeding some amount, further increasing the number of training samples seems to put the network into confusion. The convergence gets more and more difficult. As an illustration, let's now examine what happens with

the medium database, and its important diversity. Still working with the Bark features scheme B, the results get very bad:

MLP	2 layers
Bark Spec. Type A PCA (10), LDA (5)	85.56 ± 1.29
Bark Spec. Type B No reductions	73.73 ± 2.00
Bark Spec. Type B PCA (10), LDA (5)	74.56 ± 1.88

Table 5.16: *Recognition performance (%) using MLP classifier, with the medium database (6 classes). N=1024, 100 iterations for each value.*

Compared to the 97-98% of good results obtained with the GMM or HMM schemes, this performance is very deceiving, even with reductions. The important diversity inside each class is responsible for the considerable quality decrease. This time, the problem is global: unlike in the 3-class case, the confusion matrix is equally balanced, showing no particular deficient class. All classes suffer from a too large diversity between signals. The MLP neural network cannot afford such a difficult case. This problem is increased in method B, because frames are considered individually. In such a scheme, as illustrated on Appendix C, some frames (especially during the pulse attack) can be totally different than the core of the class, or even worse, similar to other class attacks. This makes the training set very confused, and raise the diversity inside each class, what is very difficult to afford for the MLP.

In conclusion, the multi-layer perceptron seems not to be appropriate for such a difficult application. It could be used when the sound classes are very well delimited, but important diversity inside the different classes is difficult to deal with. Different and more advanced networks could be further studied and considered, like Radial Basis Functions (RBF) or Masking Fields [Lebe93]. It is possible that such network architectures reveal as more adapted to the considered problem, in which the number of input observation features is very high.

5.6 Conclusion

In this chapter, a comparative study of techniques for the recognition of impulsive sounds has been performed. Different signal analysis schemes and features optimization methods were tested, involving four types of classification architectures: Bayes classifiers, GMM, HMM, and MLP. Three sound databases have been considered, involving 3 classes with reduced diversity, 6 classes with high intra-class diversity, and 10 classes with balanced diversities.

The study has shown that encouraging results could be obtained (around 97-98% depending on particular training samples), even with important intra-class diversity. When all signals in the considered classes are rather similar, very good results can be achieved (near 100%), only using a simple classification architecture as the gaussian Bayes model. However, more advanced systems (GMM, HMM) are necessary when the number of sound classes increases. The gaussian mixtures model (4 components) or hidden Markov model (3 states) can provide a perfect 100% recognition rate, when applied to a few well-definite sound types. On the opposite, using neural networks, the multi-layer perceptron turned to be very sensitive, especially to intra-class diversity. Its use may be inadvisable to such a difficult application, involving non-stationary signals.

The best features scheme was found to be the Bark spectrogram, in which the frequency bands are arranged according to the human ear way of perceiving sounds. This type of sound analysis represents a very good compromise between complexity, quality, and robustness against intra-class variations. The mel-frequency cepstral coefficients proved to perform better in case of low intra-class diversities, but the number of required dimensions must be higher, increasing the complexity.

Features optimization techniques have been applied to the different features schemes. The principal component analysis (PCA) and linear discriminant analysis (LDA) can really improve the performance when the features are highly redundant or not really appropriate. On the opposite, if the original features are selected in an optimal way, their use doesn't provide any improvement, and can even deteriorate the results dramatically in case of high intra-class diversity. In fact, using such transformations and reductions should be considered thinking of the class diversities and of the targeted complexity. Reductions raise the computational load at training stage, but can decrease it during recognition, depending on the amount of dimension reduction.

To sum up, the above experiments are globally satisfying. However they were carried out assuming two important hypotheses that turn the recognition task easier:

- Clean environment,
- Closed set of sound categories.

In the next chapter, tests will be conducted to determine the robustness of the recognition algorithms, for both features schemes and classification architectures. Then, in Chapter 7, the possibility of rejecting sounds that do not belong to the categories of considered classes will be treated.

6. Robustness

This chapter studies the behavior of the features and recognition techniques presented in the above sections, when the acoustic environment is disturbed by some noise.

A few experiments will first be tried with typical background environments, as traffic noise, restaurant ambiance, or musical sounds. Gaussian white noise will be considered too. In this case, a technique will be presented to improve recognition robustness by incorporating the white noise into the learned model of each class.

Finally, techniques to whiten the acoustical background will be tested, trying to reduce all types of disturbing noise to the white situation. Thus, the efficient technique developed for this case may be used all the same.

6.1 Real World Background Noise

The robustness of recognition techniques is a critical point. Usually, models of the involved sound classes are learned under clean environment, and the performance is optimal when the incoming sounds to classify are also clean. Of course, good features should stay insensitive to background noise, but this is only possible to a certain extent. When some background noise is present, the features extracted from the incoming sound are often biased by components belonging to the background. In fact, the position and the form of the class models in the features space is modified by the background noise properties. As a consequence, comparing noisy incoming features with clean models is not correct, and can lead to bad classifications. In this section, the effect of several typical background noises on the recognition performance is observed, considering different classification methods, features schemes and signal-to-noise ratios. The experiments concern the following noises:

- Traffic noise
- Restaurant ambiance

- Musical background
- Gaussian white noise

Applying such noises to the 3 class recognition problem, the performance is as shown in the following tables, involving the Bayes classifier:

	SNR [dB]	Traffic	Restaurant	Music	White
Uniform Spec. (20 bands)	70	99.02 ± 1.12	98.80 ± 1.07	98.78 ± 1.00	98.90 ± 0.39
	60	98.90 ± 0.86	98.54 ± 1.12	98.17 ± 1.19	98.78 ± 0.81
	50	98.17 ± 1.54	98.54 ± 1.61	99.51 ± 0.63	97.93 ± 1.63
	40	98.05 ± 1.04	98.78 ± 1.00	99.15 ± 1.16	70.12 ± 6.36
	30	98.17 ± 1.04	98.78 ± 1.15	74.02 ± 8.13	47.20 ± 6.88
	20	98.17 ± 1.04	91.10 ± 8.51	40.49 ± 9.53	23.54 ± 0.82
	10	97.20 ± 1.82	35.98 ± 7.52	23.66 ± 1.54	23.17 ± 0.00
	0	35.61 ± 8.00	23.29 ± 0.30	23.17 ± 0.00	23.17 ± 0.00
	-10	23.17 ± 0.00	23.17 ± 0.00	23.17 ± 0.00	23.17 ± 0.00
Bark Spectrogram	70	99.63 ± 0.59	99.63 ± 0.59	99.51 ± 0.63	99.76 ± 0.51
	60	99.88 ± 0.39	99.39 ± 0.64	99.76 ± 1.19	99.51 ± 0.63
	50	99.15 ± 0.59	99.76 ± 0.51	99.51 ± 0.63	99.39 ± 0.64
	40	99.63 ± 0.82	99.51 ± 0.63	100.00 ± 0.00	96.95 ± 2.32
	30	98.90 ± 1.21	99.76 ± 0.51	99.76 ± 0.77	91.83 ± 2.82
	20	97.80 ± 2.29	96.34 ± 4.60	66.10 ± 16.41	85.12 ± 4.10
	10	87.93 ± 5.70	52.68 ± 13.46	26.22 ± 4.89	38.90 ± 6.23
	0	45.85 ± 12.68	23.17 ± 0.00	23.17 ± 0.00	23.17 ± 0.00
	-10	23.17 ± 0.00	23.17 ± 0.00	23.17 ± 0.00	23.17 ± 0.00
MFCC coefficients (40)	70	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.88 ± 0.39
	60	100.00 ± 0.00	99.88 ± 0.39	99.76 ± 0.51	100.00 ± 0.00
	50	99.88 ± 0.39	99.88 ± 0.39	99.63 ± 0.59	99.76 ± 0.51
	40	99.27 ± 0.63	99.27 ± 0.63	99.39 ± 0.64	99.51 ± 0.63
	30	99.15 ± 0.82	99.27 ± 1.03	98.29 ± 1.74	98.05 ± 1.54
	20	98.17 ± 3.94	96.46 ± 1.57	93.90 ± 1.29	91.10 ± 2.00
	10	90.61 ± 3.54	92.07 ± 3.06	81.95 ± 4.33	76.34 ± 4.42
	0	87.80 ± 3.54	78.66 ± 3.91	69.39 ± 3.61	64.63 ± 3.59
	-10	76.10 ± 2.24	66.34 ± 2.89	65.37 ± 2.65	64.15 ± 2.09

Table 6.1: *Recognition performance (%) when varying the SNR, for the Bayes classifier. Comparison for different features: 20 uniform bands (between 0 and 20 kHz) spectrogram, Bark spectrograms, and 40 MFCC coefficients (N=1024).10 iterations for each value. Little database (3 classes).*

Those results are interesting. Before interpreting them, let's precise that 23.17% of recognition rate with 3 sound classes is the same as no recognition. In such situations, all sounds are attributed to the same sound class, that is bottle breaks. This is not astonished, when realizing that this class possesses the wider spectrum among the little set. The above table inspires the following remarks:

- The performance decreases when the SNR gets lower, but not with the same intensity, depending on the type of disturbing signal.

- Noises with wide spectrum are dramatically more disturbing than narrow-band noise. Table 6.1 shows that the white noise (which is very near to river or more generally water sounds) is the most difficult environment to deal with. The recognition performance with uniform spectrogram features already falls to 70% for a SNR at 40 decibels! The road traffic noise has a more restricted spectrum (located in low frequencies), and the uniform spectrogram recognizer still gives 97% recognition at 10 dB. Other noises are situated between those quite different performances.

- The Bark spectrogram is much more robust than the uniform distribution of the spectral bands, especially for difficult environments. At 20 dB, the Bark features provide 85% of good results, while the uniform method is not anymore trustable (23%). However, at very low SNR, the Bark technique cannot give any correct results.

- The 40 MFCC coefficients represent the most robust scheme. Without any de-noising method, they still give results between 64% and 75% for –10 decibels of SNR. This is a very good performance in such a difficult case. Let's remind, however, that –10 dB signal-to-noise ratios still correspond to situations where the pulse slightly appears over the noise level. See section 2.3 for details about the way of determining SNR estimations for impulsive signals.

Those last results are rather satisfying, but the MFCC scheme is a quite complex technique. Other solutions could be envisaged for improving the results using more simple features. The first method that comes in mind is to apply a de-noising pre-process to the input sound. This seems possible to some extent, even if danger does exist to alter important properties in the signal of interest. A few standard methods were experienced, like the spectral subtraction [Rama99] or some derivatives [Schn99]. However, no convincing results were obtained with basic methods, and studying advanced de-noising techniques would be out of scope in this report. Therefore, other solutions

which are more connected to the recognition world are considered in the next lines.

Let's consider at first the robustness of more advanced classification architectures:

	SNR [dB]	GMM (4 components)		HMM (3 states)	
		Traffic	White	Traffic	White
Bark Spectrogram	70	100.00 ± 0.00	100.00 ± 0.00	99.39 ± 0.86	100.00 ± 0.00
	60	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	50	100.00 ± 0.00	100.00 ± 0.00	98.78 ± 1.72	96.95 ± 0.86
	40	100.00 ± 0.00	98.17 ± 2.59	98.78 ± 0.00	89.63 ± 7.76
	30	100.00 ± 0.00	87.20 ± 14.66	98.78 ± 1.72	83.54 ± 7.76
	20	98.78 ± 0.00	74.39 ± 20.70	98.78 ± 1.72	72.56 ± 14.66
	10	96.34 ± 0.00	67.07 ± 5.17	99.39 ± 0.86	58.54 ± 6.90
	0	73.17 ± 10.35	56.71 ± 14.66	77.44 ± 16.38	25.61 ± 0.00
	-10	42.07 ± 11.21	40.85 ± 6.04	26.22 ± 4.31	24.39 ± 1.72
40 MFCC Coefficients	70	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.76 ± 0.51
	60	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.15 ± 1.16
	50	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.39 ± 1.19
	40	100.00 ± 0.00	98.29 ± 2.17	99.39 ± 0.86	98.78 ± 1.41
	30	99.88 ± 0.39	93.90 ± 5.04	98.78 ± 0.00	91.22 ± 4.37
	20	95.73 ± 3.31	88.05 ± 5.36	95.73 ± 0.86	69.88 ± 10.83
	10	92.07 ± 4.20	74.88 ± 6.48	85.37 ± 1.72	59.88 ± 12.39
	0	85.00 ± 3.64	63.05 ± 4.23	88.41 ± 0.86	57.56 ± 7.42
	-10	71.83 ± 4.32	61.83 ± 2.48	77.44 ± 6.04	62.93 ± 5.02

Table 6.2: *Recognition performance (%) when varying the SNR, for the GMM (4 components) and HMM (3 states) classifiers. Comparison for the Bark spectrogram and 40 MFCC features (N=1024). 2 iterations for each value. Little database (3 classes).*

For Bark spectrograms, this table shows that GMM and HMM models provide better results than the Bayes classifier in both clean environment (100%) and high background noise level (40% -> 70% for traffic noise and 20% -> 40% for white noise at 0 dB). However, using MFCC features, this progression is not so evident. In this case, their use is not as good as expected, and approximately equivalent results are obtained with the single gaussian models. This conclusion may change with more classes and more intra-class diversity.

In the next section, the most difficult case of gaussian background noise will be specially considered, building several models of each sound class, one for all different potential levels of white noise that may disturb the impulsive sound.

6.2 Models with Background White Noise

In fact, in a more general way, the comparison between the incoming features and the learned models should be done under *matching conditions*. Indeed, if the models of the classes are learned with a noisy background that corresponds to the background present at the moment of the classification, the recognition performance are observed to be satisfactory. As a consequence, one solution would be to adapt the clean models to the current background noise, modifying them with a parametric linear transformation *T*, that "adds" the noisy background:

$$p(\vec{x} \mid \omega_i) \mapsto p'(\vec{y} \mid \omega_i) \tag{6.1}$$

with:

$$\vec{y} = T(\vec{x}) = A\vec{x} + \vec{b} \tag{6.2}$$

The difficulty stands in finding the correct transformation and associated parameters. This solution is well adapted to simple situations, where a limited number of listed background noises can occur. This closed set of backgrounds can be analyzed and a fine estimation of the transformation parameters can be done. Otherwise, an on-line non-supervised re-learning of the models should be used [Couv97, *adaptive classifier*], determining the appropriate transformation based on both clean and unsupervised training set. This task is however difficult and computationally intensive to be used in practice. Another simpler solution could be considered, leading all situations back to the "generic" white noise case.

The band-limited gaussian white noise is a commonly used approximation for most encountered background types. However, white noise presents components in all frequency bands, what makes the spectral features very sensitive to its presence. As observed in the above tables, recognition results are already bad with very little amount of noise (40 dB). However, if the class models are built adding the corresponding level of white noise, it appears that the performance gets clearly better for low SNR. On the opposite, in clean

environments, adding a little amount of white noise (70 dB SNR) already deteriorates the results.

	SNR [dB]	Clean Models	Matching Conditions
Uniform Spec. (20 bands)	70	98.90 ± 0.39	97.56 ± 1.29
	60	98.78 ± 0.81	97.32 ± 1.50
	50	97.93 ± 1.63	97.80 ± 1.26
	40	70.12 ± 6.36	97.68 ± 1.21
	30	47.20 ± 6.88	96.10 ± 2.80
	20	23.54 ± 0.82	91.22 ± 5.04
	10	23.17 ± 0.00	86.10 ± 3.78
	0	23.17 ± 0.00	78.78 ± 4.42
	-10	23.17 ± 0.00	72.80 ± 4.56

Table 6.3: *Comparison of recognition performance (%) under different levels of background white noise, using clean models (90 dB) or matching condition models. Bayes classifier, Uniform spectrogram (20 bands between 0 and 20 kHz, N=1024).10 iterations per value. 3 classes.*

As a consequence, it can be advantageous to build and store several models for each class, each model corresponding to a specified level of white background noise. A sufficiently fine level resolution should be selected, like one model each 10 SNR dB. Then, after roughly estimating the current SNR level (see section 2.3), the corresponding class models can be used for classification of the incoming signal. In such a way, good recognition performance can be obtained, as observed when comparing Table 6.4 with the corresponding values in Table 6.1 and Table 6.2. One can see that values above 80% can be obtained at –10 dB, with HMM and MFCC coefficients. Using simpler techniques, the Bayes classifier provides results around 72% at the same noise level, with only 20 uniform spectral bands. Those results are most satisfying.

As an option, if the SNR determination is not sure, several models of each class corresponding to levels around the obtained SNR value can be used. Then, the performance is sometimes better, but computational complexity and time too. The best compromise consists in using the nearest model, as well as its two directly surrounding ones, especially when the obtained SNR is low. For example, if the SNR estimation gives about 20 dB, the models that correspond to 10, 20, and 30 dB could be tested. Then, the recognizer will have to deal with $3N_c$ classes, choosing the one that gives the higher *a-*

*posterior*i probability. In such a method, however, too many classes are confronted for the Bayes classifier to give good results. GMM or HMM architectures are more adapted.

	SNR [dB]	Bayes Classifier	GMM (4 comp.)	HMM (3 states)
Bark Spectrogram	70	99.51 ± 0.63	100.00 ± 0.00	99.39 ± 0.86
	60	99.51 ± 0.63	98.78 ± 1.72	100.00 ± 0.00
	50	99.15 ± 0.82	99.39 ± 0.86	100.00 ± 0.00
	40	99.63 ± 0.59	100.00 ± 0.00	100.00 ± 0.00
	30	98.54 ± 0.96	99.39 ± 0.86	100.00 ± 0.00
	20	97.80 ± 1.38	98.78 ± 0.00	99.39 ± 0.86
	10	94.02 ± 2.60	97.56 ± 0.00	100.00 ± 0.00
	0	86.34 ± 4.84	95.73 ± 0.86	93.29 ± 4.31
	-10	74.63 ± 5.94	77.44 ± 9.49	82.93 ± 3.45
40 MFCC coefficients	70	99.76 ± 0.51	100.00 ± 0.00	100.00 ± 0.00
	60	99.76 ± 0.51	100.00 ± 0.00	100.00 ± 0.00
	50	99.76 ± 0.51	100.00 ± 0.00	99.39 ± 0.86
	40	98.41 ± 1.00	100.00 ± 0.00	100.00 ± 0.00
	30	98.05 ± 1.18	100.00 ± 0.00	98.78 ± 0.00
	20	94.63 ± 1.04	96.95 ± 0.86	99.39 ± 0.86
	10	88.54 ± 1.84	95.73 ± 0.86	98.17 ± 0.86
	0	86.83 ± 3.14	93.90 ± 1.72	96.34 ± 1.72
	-10	80.00 ± 4.15	76.22 ± 2.59	82.32 ± 2.59

Table 6.4: *Recognition performance (%) with white noise in matching conditions, when varying the SNR. 2 iterations for each value. 3 classes.*

6.3 Noise Whitening

The previous section has shown that good recognition performance is possible even when an important level of noise is present in the background. For that, *matching conditions* are essential between learning and classification time. In some particular situations, pre-built models can be stored, including the main types of encountered backgrounds, with proper levels. This solution is certainly the one that provides the best recognition performance. However, it requires an on-place training, and eventually periodic re-designs of the models, what represents an important human support.

In a more simple way, one could say at first, that white noise is the best compromise to approximate all kinds of noise, and compare all possible input signals to models trained with white noise in the background. The results would be as illustrated in the following tables for background examples including car traffic, music or restaurant ambiance:

	SNR [dB]	Traffic	Restaurant	Music
Uniform Spec. (20 bands)	70	98.29 ± 1.18	98.54 ± 1.12	98.54 ± 0.96
	60	97.32 ± 2.14	97.80 ± 1.89	97.44 ± 1.57
	50	96.71 ± 1.63	95.37 ± 2.43	94.63 ± 2.17
	40	91.95 ± 3.91	92.32 ± 3.59	94.02 ± 2.47
	30	85.61 ± 4.80	86.83 ± 4.33	94.76 ± 1.73
	20	85.69 ± 3.61	84.02 ± 5.41	91.95 ± 3.41
	10	78.17 ± 6.46	62.32 ± 5.93	83.90 ± 3.93
	0	75.00 ± 5.15	54.27 ± 4.75	69.76 ± 4.73
	-10	60.98 ± 2.15	37.93 ± 8.84	26.71 ± 7.89
Bark Spectrogram	70	99.63 ± 0.59	99.63 ± 0.59	99.88 ± 0.39
	60	99.76 ± 0.51	99.63 ± 0.59	99.63 ± 0.59
	50	98.90 ± 1.07	99.27 ± 063	99.27 ± 0.85
	40	97.80 ± 1.12	97.56 ± 1.99	98.90 ± 0.69
	30	94.02 ± 1.46	94.88 ± 2.36	95.37 ± 2.29
	20	91.34 ± 2.11	87.44 ± 4.91	90.49 ± 2.92
	10	89.88 ± 4.07	71.71 ± 7.60	80.61 ± 5.85
	0	72.44 ± 5.55	51.10 ± 9.63	63.78 ± 2.44
	-10	48.90 ± 4.16	36.59 ± 15.05	28.05 ± 9.46
MFCC coefficients (40)	70	99.76 ± 0.51	99.76 ± 0.51	99.51 ± 0.63
	60	99.51 ± 0.89	99.88 ± 0.39	99.76 ± 0.51
	50	99.27 ± 0.89	99.15 ± 0.82	99.51 ± 0.63
	40	99.02 ± 0.96	99.02 ± 1.12	98.54 ± 1.38
	30	97.07 ± 1.31	97.56 ± 1.15	97.56 ± 1.15
	20	90.24 ± 4.74	94.76 ± 2.70	95.49 ± 1.73
	10	84.02 ± 2.66	91.10 ± 3.30	91.46 ± 1.91
	0	83.54 ± 5.46	82.20 ± 4.50	86.34 ± 3.34
	-10	76.95 ± 5.59	73.54 ± 6.33	74.02 ± 4.81

Table 6.5: *Recognition performance (%) when varying the SNR, for the Bayes classifier. Models with corresponding levels of background white noise. Comparison for different features: 20 uniform bands (between 0 and 20 kHz) spectrogram, Bark spectrograms, and 40 MFCC coefficients (N=1024).10 iterations for each value. 3 classes.*

Comparing this table with Table 6.1, the results are impressive. In the case of low SNR levels, the values are largely better than with clean models, especially when using simple features schemes. As an example involving Bark spectrograms at 10 dB, the performance changes from 26.22% to 80.61% for music background. For traffic noise, whose performance is already good with the clean model, the values stand from 87.93% to 89.88% in the same situation. Here, considering the 0 dB level in which the clean model suddenly falls to 45.85%, the white noise model raise the performance to 72.44%.

On the opposite, the method globally decreases the performance when the SNR stands between 30 and 70 dB. In such rather acceptable level of noise, the clean models are better than models using added white noise. In fact, the beneficial effects of the method largely depend on both noise level and spectrum span of the background noise. Thus, the low-frequency traffic noise takes less advantages of this method than the musical background, whose spectrum is larger and nearer to the modeled white noise spectrum. In fact two main observations can be done:

- When the background spectrum is narrow, clean models are sufficiently robust to give good results. Of course, when the SNR gets very low, measures must be taken, and the use of white noise models is justified.

- When the background spectrum is larger, advantage does exist to use white noise models, already with higher SNR values.

When using GMM or HMM architectures, the same observations globally hold. Slightly better values are obtained at very low SNR levels, and the HMM solution seems particularly robust, particularly with MFCC coefficients (80% at –10dB).

	SNR [dB]	GMM (4 comp.)	HMM (3 states)
Bark Spectrogram	70	99.39 ± 0.86	100.00 ± 0.00
	60	100.00 ± 0.00	98.78 ± 0.00
	50	97.56 ± 1.72	94.51 ± 2.59
	40	90.24 ± 5.17	88.41 ± 0.86
	30	82.32 ± 2.59	59.15 ± 4.31
	20	86.59 ± 3.45	57.32 ± 6.90
	10	69.51 ± 3.45	66.46 ± 7.76
	0	72.56 ± 11.21	62.20 ± 18.97
	-10	32.93 ± 8.62	50.61 ± 7.76

40 MFCC coefficients	70	100.00 ± 0.00	98.78 ± 1.72
	60	100.00 ± 0.00	100.00 ± 0.00
	50	99.88 ± 0.39	100.00 ± 0.00
	40	97.68 ± 2.60	95.12 ± 0.00
	30	94.51 ± 3.69	92.68 ± 1.72
	20	94.51 ± 2.71	94.51 ± 4.31
	10	91.34 ± 2.33	89.02 ± 0.00
	0	89.63 ± 2.52	85.98 ± 6.04
	-10	75.49 ± 5.53	80.49 ± 10.35

Table 6.6: *Recognition performance (%) when varying the SNR, for the GMM and HMM classifiers. Traffic background noise. Models with corresponding levels of background white noise. Comparison for different features: Bark spectrograms, and 40 MFCC coefficients (N=1024).2 iterations for each value. 3 classes.*

However, this solution doesn't complete the matching conditions perfectly. More improvements can be expected for narrow-band background spectra. Two solutions can be considered:

- Performing an estimation of the current background spectrum, and applying models that were trained not only with the corresponding level of background noise, but also with the corresponding spreading of the white spectrum (narrow-band white noise). For example, 25 types of narrow-band white spectrum could be included, corresponding to partial Bark spectrums, where the i^{th} stored spectrum would possess non-zero components in the i first critical bands. However, this would require to store a lot of models in memory, because in addition to the 10 level models, 25 different spectrum models should be included for each SNR level. This method won't be investigated in this report, but good results could be reasonably expected.

- Performing a whitening of the background noise, before starting the recognition process. Let's consider this solution in more details:

The solution presented in the following lines still consists in learning and storing several models per sound class, corresponding to different levels of gaussian white noise (like in the previous section). Then additionally, when a noisy input signal is presented to be classified, a simple time-domain whitening of the real background noise is performed, replacing that noise with some randomly generated white sequence, of equivalent variance. In this way,

the matching condition can be nearly reached. Practically, the whitening process should not affect the signal of interest (that is the main part of the pulse), which is supposed to be higher than the background level. In fact, a threshold is placed just above the previously estimated background level, and only the audio samples, which amplitudes stand below this threshold, are replaced by a gaussian noise. This process supposes that the background noise should be sufficiently stable, and that the noise is insignificant in the time zone corresponding to the main part of the pulse. As a consequence, the method will not work with very low SNR. The principle is illustrated on Figure 6.1.

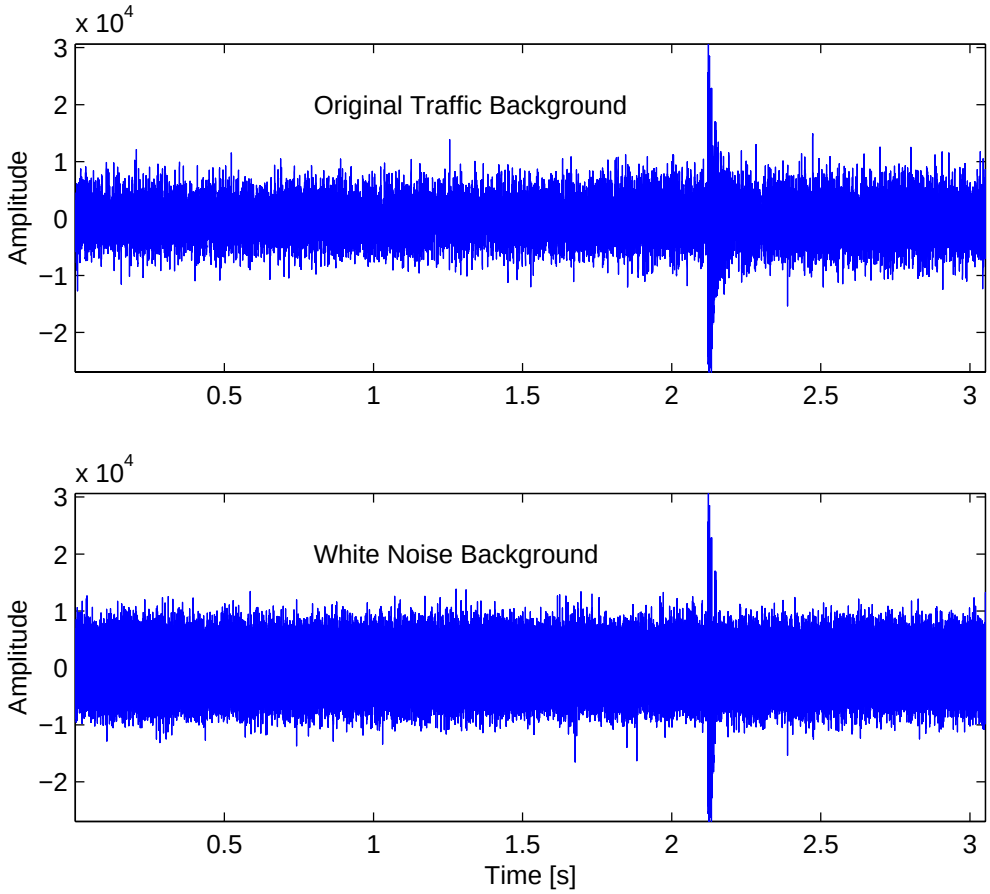


Figure 6.1: *Comparison of the signals before and after the whitening process, in the case of a gunshot pulse in cars traffic background. The SNR is 1.5 dB, the variance of the white noise is the same as the original noise variance, and the threshold for whitening is set at the amplitude of 15000.*

It can be observed that the new level of noise is slightly above the original level. In case of very difficult noise, both parameters of the whitening process

can be tuned, that is the variance of the replacing white noise, and the amplitude threshold. The amplitude threshold must be high enough to correctly replace the original background, but not too large to keep a sufficient amount of the pulse. Ideally, it should be adapted to the signal-to-noise ratio. The value can be selected in an adaptive manner, measuring both noise variance and maximas, and impulsive signal maximum.

The performance of this method is measured using the HMM classifier and the medium database, which is more difficult to deal with because of the high intra-class diversity. Table 6.7 shows the results when trying to recognize signals of the medium database placed in a musical background. In the SNR range between 0 and 50 decibels, the values illustrate a noticeable improvement compared to non-whitened signals. At 20 dB, the improvement nearly reaches 30%. The amplitude threshold is set to 3 times the variance.

	SNR [dB]	Matching Conditions without Whitening	Matching Conditions with Whitening
Uniform Spec. (40 bands)	70	97.80	96.10
	60	96.59	94.88
	50	93.41	93.41
	40	84.39	93.17
	30	68.05	82.44
	20	52.20	81.71
	10	42.44	66.34
	0	33.13	42.69
	-10	52.17	39.13

Table 6.7: *Recognition Performance (%) for the HMM classifier in a musical background, with and without the whitening process. 40 uniform bands are used as features. 6 classes (medium database). Only one iteration.*

6.4 Conclusion

In this chapter, the problem of recognition robustness was addressed. MFCC coefficients were found to be the most robust features, performing better than the Bark spectrogram, which still represents the best compromise, when taking the complexity (number of dimensions) into account. Uniform

spectral decompositions seem definitely not adapted, except if a lot of redundant dimensions are present (40 or 50 bands).

Encouraging results have been obtained when taking the background noise into account during the training of the sounds. The matching conditions still provide recognition rates above 80% around –10 dB SNR levels. Unfortunately, they are not easy to obtain in all real world situations. A whitening process of the background was proposed, increasing the recognition rates for reasonable levels of noise. This method uses models of the different sound classes, in which white noise is added at training stage, according to different possible levels of noise. This idea could be deepened, building several models of the classes, each one with a different kind of spectral envelope as background noise.

In another way, advanced de-noising techniques could be considered at algorithm level, but also at system level, including beamforming techniques, working with 2 or more microphones. Such systems provide increased possibilities for the de-noising task and could represent a solution to difficult applications as portable systems.

More generally, the matching conditions should be considered as often as possible in such a recognition system. Their importance is not only primordial for noisy backgrounds, but also concerns the acoustical environment (room reverberation) or the prior stages of the recognition system (AD converters or microphones characteristics).

7. Rejection

All the recognition techniques studied in the above chapters were designed to always classify the input signal in one of the defined sound classes. It means that any signal, as different it can be from all considered classes, would be attributed to the nearest class. Of course this is not suitable, when working with an open set of sound classes. In this chapter, the possibility of rejecting detected sounds that do not belong to the considered set of interesting classes is addressed.

This is a most delicate task that often decreases the recognition performance by rejecting correctly classified sounds. Several solutions are tested, using thresholds on the a-posteriori probabilities, examining the coherence of the classification results in time, or building complementary world classes. Those diverse rejection schemes can be used in parallel and the final decision may depend on the globally gathered information.

7.1 Thresholding the A-posteriori Probabilities

When considering the Bayes decision rule, as presented in Section 5.2, the goal is to minimize the global classification error. This means that the same importance is attributed to all class errors. However, it can sometimes be useful to attribute more importance on some specific type of miss-classifications. For example, in the case of hearing aids, interpreting a bird sound as a car horn may not be critical, but the opposite could put a life in danger. This problem will not be experienced in this study, but the *Generalized Bayes Decision* rule, that allows such differentiation, can be used all the same to introduce the rejection problem.

Let's consider the generalized Bayes decision rule, as described in [Devi82]. Instead of searching the maximum among all *a-posteriori* probabilities, like in (5.3), the principle is to determine the winner class $\omega(\vec{x})$ that minimizes the weighted sum of *a-posteriori* probabilities over all classes:

$$\omega(\vec{x}) = \omega_i \;\Big|\; i = \arg\left\{ \min_{i=1,..,N_c} \sum_{j=1}^{N_c} r_{ij} P(\omega_j \mid \vec{x}) \right\} \tag{7.1}$$

In this expression, the weight parameters r_{ij} represent a *risk* or *loss* associated with the wrong attribution of the input signal to class *i,* when it really belongs to class *j*. This would be a *false classification* (or *false acceptance*). The false classification risk is 0 when $i = j$, and the value is positive otherwise. Considering the possible attribution of the input signal to class *i,* the risk values r_{ij} associated to this class weight the *a-posteriori* probabilities obtained for all classes j, the higher the risks, the lower the chance for class *i,* to be selected. If all $r_{ij,\, i \neq j}$ are chosen equal, (7.1) becomes equivalent to (5.3), that is:

$$\omega(\vec{x}) = \omega_i \;\Big|\; i = \arg\left\{ \min_{i=1,..,N_c} \sum_{\substack{j=1 \\ j \neq i}}^{N_c} P(\omega_j \mid \vec{x}) \right\} \tag{7.2}$$

For the rejection, one more class can be imagined, corresponding to the rest of the "world". This is the rejection class, noted ω_0. Then, the Generalized Bayes decision rule can be used, attributing the same risks to all N_c considered classes ($r_{ii} = 0$ and $r_{ij,\, i \neq j} = 1$ when $i, j = 1,..., N_c$), and a *rejection risk* $r_{0j} = \lambda_r$ $(j = 1,..., N_c)$ to the rejection class. The rejection risk represents the situation in which the correct sound class was found by the system, but the result was rejected all the same, because of insufficient certainties. This would be a *false rejection.* The rejection risks are usually equal for each class *j*. Their values λ_r are typically between 0 and 1, the risk of rejection being generally lower than the false classification risk. Then, the classification can be done including the rejection class into (7.1) that becomes:

$$\omega(\vec{x}) = \omega_i \;\Big|\; i = \arg\left\{ \min_{i=0,..,N_c} \sum_{j=1}^{N_c} r_{ij} P(\omega_j \mid \vec{x}) \right\} \tag{7.3}$$

Otherwise expressed, the classification consists in choosing:

$$\begin{cases} \omega_i \text{ if } \sum_{\substack{j=1 \\ j\neq i}}^{N_c} P(\omega_j \mid \vec{x}) \text{ is the minimum among indexes } i = 1,...,N_c \\ \omega_0 \text{ if } \lambda_r \sum_{j=1}^{N_c} P(\omega_j \mid \vec{x}) = \lambda_r \text{ is much lower.} \end{cases} \quad (7.4)$$

Taking a three class example, and supposing that the class that corresponds to the minimum sum in the first line of (7.4) is ω_3, the rejection occurs if λ_r is much lower than this minimum sum:

$$\begin{aligned} \lambda_r &< P(\omega_1 \mid \vec{x}) + P(\omega_2 \mid \vec{x}) \\ \lambda_r &< 1 - P(\omega_3 \mid \vec{x}) \\ P(\omega_3 \mid \vec{x}) &< 1 - \lambda_r \end{aligned} \quad (7.5)$$

In other words, the rejection option is selected when the most probable class among $i = 1,...,N_c$ obtains an *a-posteriori* probability that does not exceed the threshold 1-λ_r. Hence, a rejection can be simply decided, using the traditional Bayes decision rule (5.3), when the maximum *a-posteriori* probability is lower than the probability 1-λ_r. The choice of 1-λ_r must be performed according to experiments, being a compromise between *false classification* and *false rejection* rates. Typical values for 1-λ_r start from 0.7 and can go near to 1 (0.999).

This method works for features of *type A* (see Section 4.2), in which all the signal is globally classified. When successive frames are considered (*type B*), the resulting average *a-posteriori* probability (or global likelihood) must be calculated over all frames, and used in the rejection criterion. Another possibility consists in first rejecting all frames that do not obtain individual *a-posteriori* probabilities higher than 1-λ_r. Then, if the number of remaining frames is not important enough, compared with the total number of frames in the significant signal (proportion lower than 70% for example), the signal can be declared as rejected. However, this solution involves two connected thresholds, and their optimal design is very difficult. Furthermore, a signal with mixed properties (its beginning being clearly attributed to one class, and the end to another class) would not be rejected.

7.1.1 Examples

Let's consider an example involving the three-class database and a Bayes recognizer working with type A spectrogram features. The spectrogram analysis is performed during one second for 1/8 overlapping blocks of 1024 time samples, and the number of frequency bands is 5 between 0 and 15 kHz. The concatenated features are reduced to dimension 10 with a PCA transformation, and then to dimension 2 with an LDA reduction (see Subsection 5.2.3).

The first experiment doesn't concern the rejection of out-of class signals, but its goal is to see how the *a-posteriori* probability threshold can be used to increase the recognition performance in a closed system. In such case, the function of the rejection is simply to discard signals that have great probability to be wrongly classified. Choosing a threshold of 0.99, and performing 10 cross-validation iterations, the following confusion matrices are obtained (rejected signals are eliminated of the confusion matrix representation):

Without rejection:
$$\begin{bmatrix} 339 & 1 & 0 \\ 3 & 284 & 3 \\ 2 & 0 & 188 \end{bmatrix} \tag{7.6}$$

With rejection:
$$\begin{bmatrix} 332 & 0 & 0 \\ 1 & 275 & 0 \\ 1 & 0 & 182 \end{bmatrix} \tag{7.7}$$

In other words, including the rejection option, the performance increases from 98.90% to 99.75%. The rejection was performed with success for 77.8% of the classification errors in (7.6), but 2.68% of correctly attributed signals were rejected. This may be an acceptable situation. Of course, decreasing the threshold to 0.90 would reduce both proportions (66.7% and 1.34% respectively), according to the well-known false classification/rejection compromise. Conversely, if 6.59% of false rejection is tolerated, a zero false classification rate can be reached, with a threshold of 0.999. In this case the recognition performance increases from 98.78% to 100%! This is a quite good compromise. However, this experiment lets appear the difficulty of the rejection job, because the falsely classified signals that remains in (7.7) have an *a-posteriori* probability that is above 99%, what means that they really possess properties near the wrong class they were classified in. As a

consequence, the false rejection rate has to be raised to 6-10% to be sure to eliminate false classifications. This limits the possibilities of the threshold method, and questions about the classifier and the features scheme. In fact, if the number of spectral bands is increased, or if the classifier architecture is improved, better results should be expected. In the example above, using the Bark spectral decomposition instead of 5 uniform bands, the 100% good rejection rate can be obtained with only 5.42% of false rejection (threshold 0.999).

Let's now consider the open-set case in which out-of-class signals may be presented to the recognizer. In addition to the 3 defined classes, the recognition system is confronted to a fourth type of signals. In the next examples, 30 dog barks, and then 30 children voices, are presented. The features scheme is the Bark spectrogram. After 10 cross-validation iterations, using a threshold of 0.99, the respective confusion matrices are as follows:

$$\begin{bmatrix} 318 & 0 & 3 \\ 0 & 277 & 0 \\ 1 & 0 & 156 \\ 81 & 25 & 15 \end{bmatrix} \text{ and } \begin{bmatrix} 323 & 0 & 0 \\ 0 & 271 & 0 \\ 0 & 0 & 168 \\ 9 & 24 & 18 \end{bmatrix}$$

The added last line represents the class to be rejected, for which no model was previously built. As a consequence, no classification is possible in the fourth column, that is not shown. Then, it turns out (on the left) that 40.33% of the dog barks are not rejected and essentially classified as door slams (false acceptance). Only 59.66% of them are correctly rejected. Therefore, dog barks are very near to door slams! Let's remind however (Section 5.5) that door slams seem to be a highly spread class.

Considering the children voices, the false acceptance rate is better, reaching 17%. In this last case, the false rejection rate is only 5.24%, and 15 previously bad-classified sounds were rejected by the thresholding method. This is a good deal, increasing the closed set performance from 98.17% to 100%.

It seems therefore that thresholding the *a-posteriori* probabilities is not a very trustable rejection method to be used all by itself. Depending on the involved classes and external noise types, the rejection performance can importantly vary. As an illustration, if doors and dog barks are exchanged in the previous experiment, including dog barks as the first indexed class and doors as the outsider class, the false acceptance rate drops to 23%, instead of 40.33%.

Let's now consider the B-type spectrogram with 20 bands between 0 and 20 kHz (no reduction) for the little database, as in (7.6) and (7.7). The rejection performance is decreased, compared with type A. When simply considering rejection inside of the closed set, the 0% false classification rate is reached at the price of 14.36% of false rejections, using an average threshold of only 0.8. With 0.75 as threshold, those values become 15.25% and 8,07% respectively. It turns out that the classification of each frame is not as clear as the classification of the whole signal. Thresholds must be lowered. In fact, more confusion is present in this case, because of individual frames that can not be surely attributed (for example, at beginning of the signal or during transition parts) or that clearly belong to another class. The importance of those frames was attenuated in A-type spectrograms, because of the reduction process. As a consequence a refined rejection method has to be designed for B-type features, as described in the next section.

7.2 Coherence of Successive Frames Classification

In the frame-by-frame recognition schemes (*type B* in Section 4.2), the recognition results are considered for all successive frames of a signal. If those results are sufficiently coherent during the significant part of the signal, the classification can be accepted. On the opposite, when the class with the average maximum *a-posteriori* probability is not the winner class for a sufficient majority of the frames (say 80%), the signal can be declared as rejected, independently of the values of *a-posteriori* probabilities.

In the following experiments, a first threshold th_1 is applied on the *a-posteriori* probabilities of each frame, according to section 7.1. The frames that do not obtain an *a-posteriori* probability above the threshold $th_1 = 1 - \lambda_r$ are rejected, and a sufficient relative percentage th_2 of non-rejected frames must be obtained. Then, involving all the remaining frames, the balance of classification results is performed: each one of the N_c classes may be the winner for some frames. The class $\hat{\omega}$ with the maximum number of winning frames should be chosen as the final classification result. However, if the *winning frames ratio* Wf_r (7.8) is not higher than a second threshold $th_3 = 1 - \lambda_r'$, the signal is rejected:

$$Wf_r = \frac{Nf_{\hat{\omega}}}{\sum_{i=1}^{N_c} Nf_i} \tag{7.8}$$

In this expression, Nf_i and $Nf_{\hat{\omega}}$ are the number of winning frames for the i^{th} class, and for the most probable class $\hat{\omega}$, respectively.

With such a scheme, tuning the thresholds th_1, th_2, and th_3 to 0.99, 0.50, and 0.85 respectively, the Bark spectrogram applied to the recognition of doors, explosions and bottles, provides a false acceptance rate for dog barks around 10.5%. All previously falsely classified signals were rejected, but the false rejection rate is 14.07%, essentially doors. Those rejection rates are globally not better than for the simple *a-posteriori* thresholding of the previous section. The values are still not satisfying enough, if GMM or HMM architectures are used. Furthermore, with the high diversity of the 6 class database, the false rejection rate reaches 28.5% for 26.4% of false acceptance and 88% of rejection among previously false classified signals. As a conclusion, another technique than just thresholds should be found, as described in the following section.

7.3 Creation of a "Rest of the World" Class

In this technique, one additional world class is really built, including all possible sounds not being part of the considered categories. The creation of this class should theoretically involve an exhaustive and very large library of possible sound examples. However, such a general and wide model is often difficult to build practically, considering the huge diversity of potential sounds. In fact, this technique may be efficient and provide interesting performance when the situation is well defined, that is, when a restricted list of out-of-class sounds can be drawn up. For example, in a specified room, the most often encountered out-of-class sounds can be recorded before use of the system, and used to train a well-fitted world class.

In some cases, and more simply, the world class could be built artificially, generating random points in the features space (or reduced features space). This process can be illustrated in the case of three classes with reduced features of dimension 2. In Figure 7.1, the three sound classes plus the world class are represented by their boundaries and features population. The world class is populated by artificial points in the features space, which are

randomly generated in a uniform manner between extreme fixed points of the space. However, this method gets rapidly difficult to control when a lot of features dimensions are present.

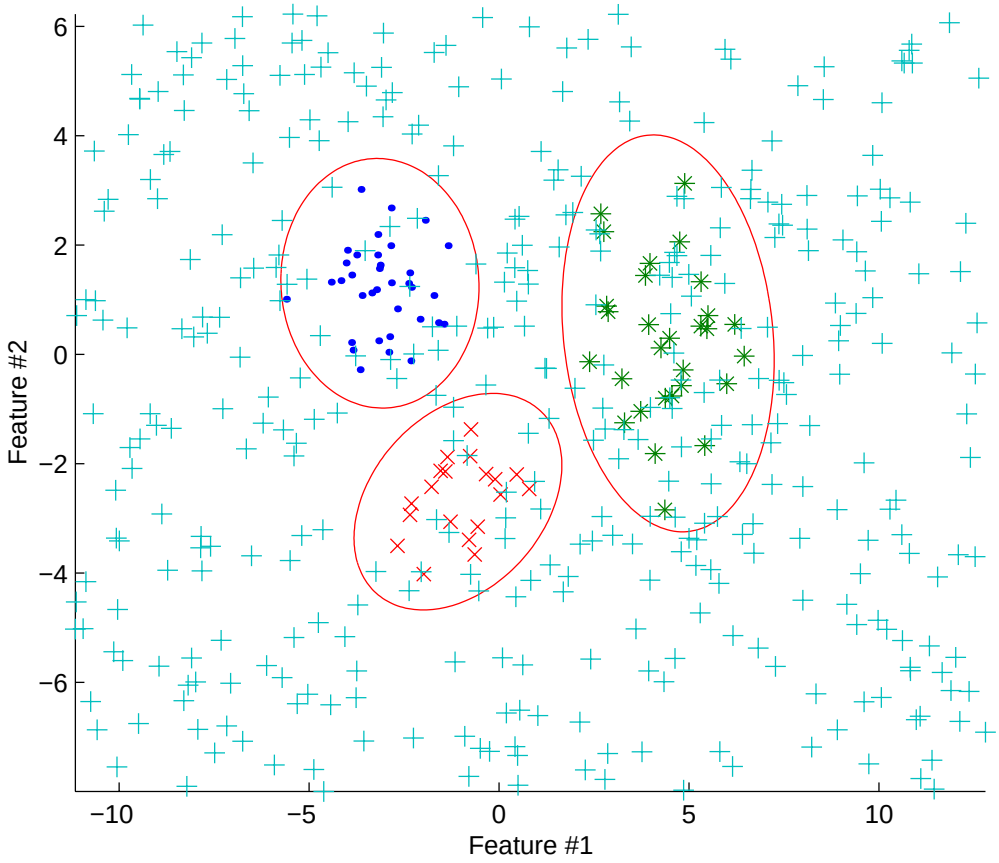


Figure 7.1: *Features population in the artificial world class (+) and boundaries with the three sound classes of interest.*

In the following experiments, a real world class is built from many diverse and typical sounds that can be encountered every day. Those sounds are typical of car traffic, site building, people talking, people walking, restaurant environment, television, music, birds in the forest, trains, stations, and so on. Many such sounds are used to train a world class, which is considered in the experiments, as a supplementary class. Half the sounds are used at training and the other part at testing, to be rejected. Additionally, an outsider class is considered to be rejected too, either being ruled out with the thresholding method, or being classified in the world class.

As an example, let's consider the modified little set constituted of 3 indexed classes (dog barks, explosions, bottles) plus the large world class. Furthermore, in a second step, children voices are introduced as an outsider class to be rejected. Using the Bark features scheme (type B) and a GMM architecture with 4 gaussians, the two matrices below are successively obtained. The first one represents the classification among the 3 classes plus the world. In this case, the world class is considered as one normal supplementary class in the closed set. No threshold rejection is applied and the children voices are not considered. Then, the matrix on the right shows the obtained results when the subsequent threshold rejection is performed, taking children voices into account.

$$\begin{bmatrix} 243 & 0 & 0 & 7 \\ 0 & 290 & 0 & 0 \\ 0 & 0 & 190 & 0 \\ 10 & 7 & 11 & 172 \end{bmatrix} \text{ and } \begin{bmatrix} 231 & 0 & 0 & 2 & 0 \\ 0 & 285 & 0 & 0 & 0 \\ 0 & 0 & 176 & 0 & 0 \\ 3 & 2 & 2 & 112 & 0 \\ 6 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Representing the situation before thresholding the *a-posteriori* probabilities, the first matrix can be interpreted in the following way:

- The closed set performance among the 3 considered classes is 100%.
- The false acceptance rate among the world class is 14%.
- The false rejection rate among barks, explosions and bottles is respectively 2.80%, 0% and 0%.

Then, looking at the second matrix, after having applied a rejection threshold (0.8) on the global *a-posteriori* probabilities of the 4 classes:

- The closed set performance among the 3 considered classes is 100%.
- The false acceptance rate among the world class is increased to 3.5%.
- The false rejection rate in the 3-class set is 7.6%, 1.72% and 7.37% respectively.
- The false acceptance rate among the children voices class is 2%. All rejected signals have either been classified in the world class, or rejected by the thresholding process, this last way being dominant.

Those results are very satisfying compared to the previous ones. In fact, the presence of the large world class in the closed set is really beneficial. The global false acceptance, with both world signals and children voices is 2.60%, for an averaged false rejection rate of 5.21%. Performing the same experiment with the Bayes classifier gives slightly lower results, respectively 4.40% and 8.77%.

Considering now the 6 class medium database, to which the world class is added, the confusion matrix is as below, representing a false acceptance rate of 19.5%, and a global performance of 96.61% (with the world class).

$$\begin{bmatrix} 1514 & 1 & 16 & 0 & 37 & 0 & 2 \\ 2 & 280 & 0 & 0 & 28 & 0 & 0 \\ 9 & 0 & 428 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 360 & 0 & 0 & 0 \\ 7 & 17 & 17 & 0 & 1079 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 300 & 0 \\ 11 & 0 & 9 & 5 & 14 & 0 & 161 \end{bmatrix}$$

10 iterations with the Bayes classifier and the Bark spectrogram were involved in this example. The results are satisfying, but when submitting the outsider children voices signals, nearly all of them are classified as human screams!

7.4 Conclusion

The rejection task turns out to be a most sensitive job, whose performance depends a lot on the involved classes and on the characteristics of the signals to be rejected. Signals that possess properties that are near some of the indexed classes are fundamentally difficult to reject. The problem is stronger in case of great intra-class diversity, and becomes harder when noisy environments are encountered.

In this chapter, three rejection techniques have been experienced. The first one simply sets thresholds on the values of *a-posteriori* probabilities obtained by the most likely class. A second method was applied to features schemes in which all individual frames get a classification result. In such cases, the temporal evolution of the classification tendencies of successive frames is

observed. When too many frames are not clearly affected to the dominant class, the whole signal can be rejected. Finally, a third solution was tested, building an extra world class, supposed to gather all signals that could be out of the considered class space. Several of those solutions were tried for parallel use. Even if good results are sometimes reached, the performance of such methods is globally deceiving.

Actually, the rejection problem seems to be the most difficult task for such a recognition system. This remark also stands for speaker recognition applications. More advanced techniques were recently published [Brou00] in this domain. One of the promising method does not consist in creating just one world class, but many artificial little ones, in order to fill out the world space. This solution may increase the complexity of the classification process, but it is able to model more precisely the boundaries between the world class and the interesting sounds. In the same direction, simpler solutions could go toward a refinement of the *a-posteriori* probability thresholds, applying specific threshold values in each class. This scheme would decompose the problem in too steps, involving a recognition process at first, to provide the most probable class, and then performing a *verification* process, in order to decide if this decision is good, considering class-specific properties.

As a general conclusion, in a given situation where some particular types of outsider sounds can possibly arise, it is always better to build a special class for each one of them, rather than letting the world class or *a-posteriori* thresholds apply a rejection process. Of course, the detection problem can be designed, thinking about the particular concerned application. For surveillance systems, lower rejection performance could sometimes be accepted if the system involves other components (based on video or temperature signals). However, giving false classification results should be avoided as much as possible, at the price of a few undefined answers.

8. Detection and Recognition System

This short chapter represents the synthesis of the study, putting together the detection and recognition parts. This association has to be properly checked, because potential problems can sometimes arise, particularly in case of important background noise.

The critical point is the choice of a good signal delimitation (beginning and end positions in time) when submitting the detected pulse to the recognizer. If a bad window of signal is presented to the classifier, missing for example the attack zone, then the recognition results can drop. As a solution, a refinement of the pulse start moment can be performed, and from there, the optimal duration for signal analysis and recognition, determined.

According to the robustness solutions of Chapter 6, the whole detection and recognition system of Figure 1.1 can be refined as follows:

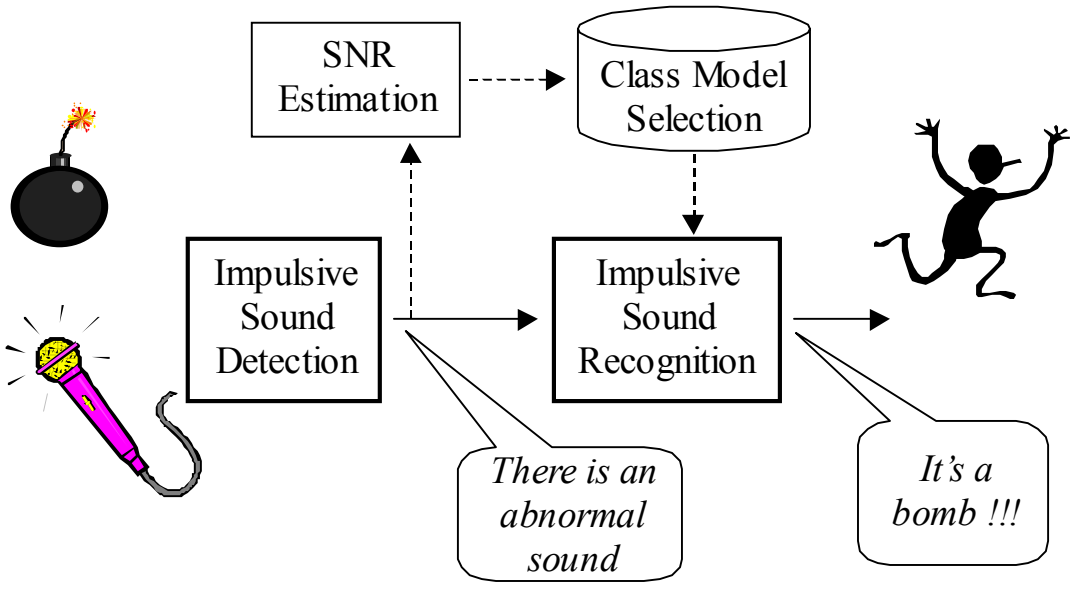


Figure 8.1: *Detection and recognition system with robustness solutions.*

In the previous chapters, the recognition results were obtained, directly performing tests on all the files in the database, one after the other, adding noise if necessary. The choice of a window for the analysis and recognition of the sounds was not involved, because the test was done on the whole file duration or on a specified duration (1 second, spectrogram type A). In this chapter, a more advanced experiment has to be performed, to test both detection and recognition performance. This experiment consists in first creating a long signal made of background noise, which duration is typically set to 10 seconds. Then, the pulse is randomly added somewhere on this long-term background, with a pre-defined SNR. The job of the system is to detect its position and then to recognize what kind of pulse it is.

The first results of this experiment proved particularly bad, and lower than the previously obtained recognition results:

Signal-to-Noise Ratio [dB]	Performance of recognition only (HMM) [%]	Performance of detection and recognition (HMM) [%]
70	98.54	98.29
60	97.07	96.83
50	95.85	93.90
40	95.37	92.93
30	94.63	90.00
20	90.73	85.12
10	88.54	72.93
0	81.95	55.26
-10	61.95	-

Table 8.1: *Comparison of recognition results with and without previous detection. Medium database (6 classes), Background white noise and 40 uniform bands features.*

It must be noted that those experiments were performed on the medium database, using the median-filter detection method presented in Section 3.3. The recognition was done with the HMM (3 states) recognizer using the simple uniform spectrogram features (40 bands, no reduction). One of the best sets of background white noise models built in Chapter 5 was used. After examining the potential causes, it was found that:

- The SNR estimation is not critical. Using three models around the estimated SNR value, with a resolution of 10 dB, as indicated in section 6.2, is sufficient.

- The detection method, used as presented in Chapter 3, is not precise enough, considering the starting time-position of the pulse. Its resolution is only 4000 samples, that is, about 1/10 seconds. A further refinement is necessary.

- Starting from the pulse attack, the length of the considered signal may be important too, depending on the noise level.

For the last two points, solutions can be proposed, improving the performance. A simple refinement of the pulse start position can be done, examining the evolution of the amplitudes around the detection point provided by the detection block of Chapter 3:

- Starting a quarter of a second before this point, the absolute amplitudes of the signal are averaged for each block of 200 samples (4.5 ms resolution), building a sequence $\vec{a}$ of 1.5 second duration.

- Then, the precise pulse start is searched over this sequence, using a *threshold*, which is fixed according to the following expression:

$$threshold = \frac{\max_i \left(a(i)\right) - basis}{sensitivity} + basis \tag{8.1}$$

  Good values for the sensitivity are between 5 and 10. The value *basis* represents the average level of absolute amplitudes in background noise only. It can be determined taking a few values at the beginning of the sequence $\vec{a}$, for example $L_b = 5$ values (1000 samples):

$$basis = \frac{1}{L_b} \sum_{i=1}^{L_b} a(i) \tag{8.2}$$

- The final starting point is selected when the first element of $\vec{a}$, being over the threshold is encountered. In case of no such elements, the starting point is positioned at the original point given by the detection block.

Concerning the choice of the signal duration to be considered, a fixed optimized value has been experimentally determined for the medium

database, corresponding to 0.75 seconds, starting from the pulse attack. Of course, better adaptive solutions could be more appropriate, for example exploiting the properties of detection algorithms as the one presented in Section 3.1, which is able to give information about the pulse duration. In another manner, a second threshold can be fixed to detect when the amplitude level returns to its original values, before the pulse.

Using those refinement techniques, the obtained results are:

Signal-to-Noise Ratio [dB]	Missed Detection rate [%]	Performance with detection and recognition (HMM) after refinement [%]
70	0	98.54
60	0	96.10
50	0	95.37
40	0	96.34
30	0	88.78
20	0	87.07
10	0	84.88
0	0	68.54
-10	12.04	48.83

Table 8.2: *Recognition results with detection, after refinement. For –10 dB SNR, 12% of the signals are not detected, and, as a consequence, not considered in the recognition performance value.*

One can observe that they are still not as good as the original values of Table 8.1, especially for low SNR. In fact, when an important level of noise is present, the precise moment of the pulse attack gets difficult to detect, because it can be lower than the noise level. In such cases, (*bad detection* situations like considered in Section 3), the recognition methods have difficulties to provide good results, especially classifiers like the HMM that take account of the time evolution of the signal. This explains the degradation of the recognition performance for SNR values below 10 decibels. In such highly noisy cases, an adaptive determination of the duration could be considered, although difficult. More improvements should be expected using traditional de-noising techniques, such as spectral subtraction for example. Those techniques could be applied at pre-processing stage, at the price of both delay and increased complexity.

To conclude this chapter, the illustration of a demonstrator developed for the Matlab environment is shown, reproducing the detection and recognition system described above. This demonstrator allows generating a 10 seconds audio signal, that contains a selected background (with some given level of noise) and one or a sequence of impulsive sounds which are randomly extracted from the database, and successively placed over the background noise. The class of each pulse can be chosen. Then, the detection algorithm searches for the pulses positions, the SNR of each pulse is determined, and finally, the HMM or GMM classifiers can be used for recognition, with the appropriate levels of background noise. The whole signal with all pulses, their respective detected positions, estimated SNR values, and recognized types are shown on the Matlab GUI interface:

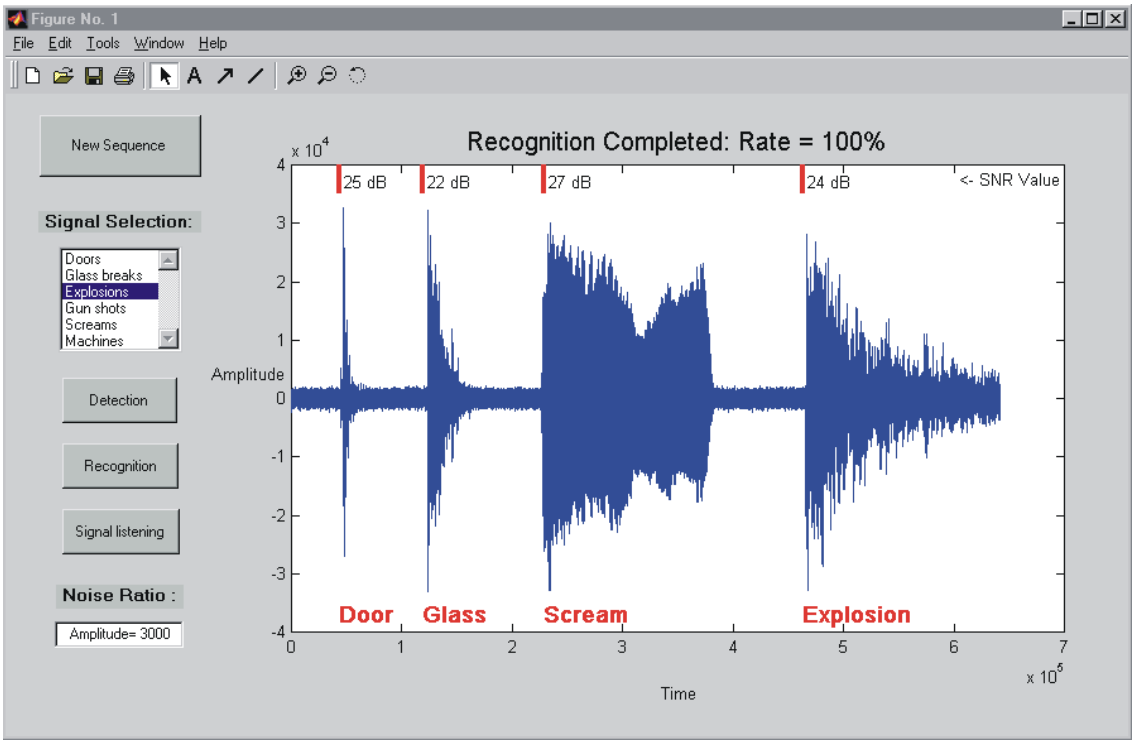


Figure 8.2: *Demonstrator with friendly Matlab interface, for the detection and recognition system.*

The global recognition performance for the considered sequence of pulses is finally indicated on the top of the figure, as well as the SNR estimations for each pulse.

9. Conclusions

As stated in introductory lines, the goal of this work was to study the feasibility of a detection and recognition system for impulsive sounds.

The main contributions can be found in the application itself, and in the system philosophy, but also in the detection and robustness methods presented in Chapters 3 and 6. The text was written in the perspective of representing a practical guide to readers starting in the field of the sound recognition problem.

This chapter will conclude the study, trying to draw up a balance sheet, emphasizing both promising results and problems that appeared along the chapters. Some guidelines toward a possible practical use of such a system as a component of a surveillance system will be given, and directions for further necessary work and improvements will be mentioned.

The goal of this study was to tackle the problem of recognizing alarm sounds using a rational two-step process. The principle consists, on one hand, in letting a simple detection system run and report all encountered special events, using sound energy variations. Then, secondly, if something is detected, and only if so, a recognition process is started during a short moment, to try and classify the incoming sound.

This innovative strategy is well fitted to surveillance systems, in which the recognition capabilities are intensively required during short alerts that do not often occur. One reason for such a strategy is to limit the continuous mathematical load, thinking about portable devices, highly sensitive to power consumption. Another reason is that the recognition algorithms are very intensive jobs, especially if they are intended to provide the best performance when confronted to a lot of classes, with high robustness capabilities and intelligent rejection characteristics. In such a case, their continuous activation could sometimes be unsuitable, if the whole classification process does not perfectly fit in the real-time limits. Let's think about the example of a robust HMM architecture having to deal with a dozen sound types and as many

artificial world classes, each of them having to be tested for several background noise levels and spectral envelopes! In fact, at system specification level, this two-step process represents the most rational compromise considering the quality to complexity ratio.

Using such a scheme, the performance of the whole system is conditioned by the detection block that has to be sensitive and precise enough. An error in the detection can either miss some important alarm, or on the opposite, provide the recognition function with too frequent signals, generally not clear and difficult to classify. In such case, the rejection function of the recognizer would have to be highly robust.

In fact, the performance of the whole recognition system depends on each one of its component part or constitutive sub-functions, like in a chained process. As expressed by the chapters' organization all along this text, the key points or functions in the system can be identified as:

- The detection function (sensitivity and precision),
- The construction of an appropriate database for class learning,
- The features selection (efficiency),
- The recognition architectures,
- The global robustness against corrupted environments,
- The rejection capabilities, for outsider alarms.

All points are most important and a defection in one of them turns the system unusable. Thus, as an important conclusion of this study, when looking at the whole application, the component that appears as the weaker link in the chain clearly proves to be the rejection functionality.

This fact represents another motivation for using the pre-processing detection stage, to eliminate all sounds that appear as not important, before they are presented to the recognizer. In the following paragraphs, conclusions, key points, and future amelioration connected to each of the previous listed functions will be developed. For more detailed conclusions on specific techniques, the reader can refer to the concerned chapters.

Pulse Detection

In Chapter 3, three simple pulse detection techniques, were presented, all based on the frame-by-frame evolution of the signal energy. One simple method using the standard deviation of the windowed energy sequence, and two other methods involving the median filter concept were described and tested. Their performance is globally satisfying, still reaching a good detection rate at low SNR values (100% at –10dB), with an insignificant false detection proportion. At system level, it is better to first perform a coarse detection, and to refine the pulse start position afterwards, considering the short-term amplitude variations of the signal. An important point about the detection is to select the correct detection threshold, according to the targeted application. The goal is to keep a good detection rate even in non-stationary background environments, without being too sensitive. The threshold should be made adaptive to the variations of energy level in the background environments. For the subsequent recognition task, in real-world background noises, no signals should be detected under 0, 10 or even 20dB SNR, to make the job tractable at rejection stage.

Training Database

In recognition systems with supervised learning, the database used at training stage completely determines the behavior of the system. The training samples should be selected with high care. The single signal that is not appropriate can considerably deteriorate the performance, and make the recognition rates of other classes getting really low. In this study, the performance appears clearly different between well-defined classes with restricted intra-variance (100%) and classes with important diversity, as the medium set (97%). As a consequence, it is always better to build many little classes, than few large classes. Even the use of advanced recognition architectures cannot solve the diversity problem. Furthermore, the rejection job becomes more difficult, as well as the behavior in noisy environments. To illustrate further the sensitivity of the models to particular signals, the cross-validation iterations showed that depending on the particular signals used for training, the performance of recognition tests changed a lot, even with the clearly defined classes of the little database.

Features Selection

In the selection of good features, it turns out that features including a logarithmic progression on both energy and frequency axis (as human ear systems) give much better performance than uniform scales, especially when

considering robustness capabilities. The Bark frequency scale spectrogram is the most efficient features scheme, considering the little number of required dimensions. The mel-frequency cepstral coefficients show the same property when compared to a uniformly spaced cepstrum. Unfortunately, the number of dimensions must be increased compared to Bark, but the robustness is really impressive (80% recognition rates at –10dB). The computational complexity is 3 times higher.

More advanced human ear models were also tested, but their huge complexity prevents them from practical use. Presently, the audio recognition community dedicates many efforts to the elaboration of such models and to their reasonable implementation. The models tested in this study didn't bring interesting results, but the number of experiments was reduced because of the complexity.

It was observed that features transformations and reductions are not really appropriate when important diversities are included in the sound classes. Furthermore, when optimal features are originally selected (like 25 Bark or 40 MFCC coefficients), the recognition scores are not improved a lot by the reductions, whose impact on the complexity is generally beneficial at recognition stage, but not during training. The use of such techniques should be decided regarding the targeted complexity of the system. Sometimes, the reduced performance of simpler (low-complexity) and non-optimal features schemes can be compensated with a features reduction.

Classification Architectures

In Chapter 5, four different architectures for recognition have been studied. The simple Bayes classifier proved to behave very well in case of little number of classes, if working in clean environments. In such cases, the improvements expected from more advanced systems can only provide a perfect and regular 100% recognition rate instead of a global 99.7% (some particular training sets giving only 98%). On the opposite, architectures like the gaussian mixture models (GMM) or hidden Markov models (HMM) are better for good robustness and large number of classes. Good scores were obtained reaching 98.5% with the high diversity six class set or 97.5% with 10 classes. No important differences in performance were observed between the GMM (4 or 8 components) and HMM (3 or 5 states, one gaussian per state) schemes. The GMM are well adapted to intra-class diversity, but the more complex HMM models should theoretically behave better when used with several gaussians per state. On the opposite, the multi-layer perceptron neural network proved very bad (73% scores), particularly when confronted to high

diversity situations. In such cases, the convergence toward parameters that should be optimal for all classes at the same time seems not tractable.

Future works should experiment advanced hybrid systems that mix both neural networks and HMM architectures in many different ways. They are the so-called Hidden Neural Networks (HNN). In such schemes, the HMM is used for its temporal alignment capabilities, and MLP neural networks are present to reinforce the classification capabilities or to perform some prior non-linear transformations on the features. Sometimes the HMM densities are replaced by a NN model. A good recapitulation of previously published HNN solutions is presented in [Riis97].

Complexity

To reduce complexity, the choice of the classifier architecture should be adapted to the environmental situation. Some trials indicate that a single Bayes classifier can be used efficiently in clean backgrounds, whereas a switch toward GMM or HMM models is necessary only during noisy periods. The complexity of HMM at recognition time is about 1.3 times larger than the GMM's, which computational load was estimated to be 2 times higher than a Bayes classification. Thus, hybrid systems that integrate several models built by a committee of architectures, and that support switching from one architecture to the other can be more efficient.

In the features choice, schemes that ensure sufficient independence between dimensions are interesting, because they notably reduce the computation, enabling zero approximations of the features inter-variance values. In this goal, MFCC are optimal.

Robustness

Concerning the robustness, this study has shown that using several sound models per class could improve the results, each one corresponding to a specific background noise level. In this way, matching conditions can be satisfied between training and testing time, when a preliminary estimation of the noise level is performed. Further investigations considering different possible forms of spectral envelope should be interesting too, provided that enough memory to store a few dozen models per class is available. Otherwise, de-noising techniques may be advised as pre-processing, under the condition not to filter out components in the bandwidth of the signals (in clean environments, it was observed that a 20 kHz bandwidth is globally optimal). Multi-layer Perceptrons (MLP) are sometimes used to learn and model a non-linear features transformation, mapping the clean features to their noisy

versions [Tamu90]. At system level, beamforming techniques involving two or several microphones could be included.

Rejection

Finally, as explained at the beginning of this chapter, the rejection function appears as the most difficult part in the recognition system. Performing the recognition among a closed set of 10 sound classes can lead to recognition rates near 98%, even in case of high diversity. However, rejecting sounds that do not belong to those 10 sets of classes is most difficult, and generally leads to an important false rejection rate among correctly classified signals. In Chapter 7, different methods are used to try and solve this problem. One method proves to provide acceptable results, which are much better than with probability threshold techniques. This method consists in building a supplementary large "world" class, in which all kind of signals are learned. However, the performance is good (4% false acceptance against 6% false rejection) only in clean environment and without an important diversity inside the different classes. In this last case, the rejection rate decreases to 80%, what is not practically acceptable.

Guidelines

To conclude, future works should be emphasized on robust hybrid recognition architectures, and on the rejection task, searching for improved techniques. One promising rejection method, involving many little garbage classes (artificially segmented world class) was recently published by Broun and Campbell for speaker recognition applications [Brou00].

More generally, an important subject that should be investigated is the problem of source separation [Couv97]. In fact, experiencing real alarm situations, one can often observe that a lot of different sounds (bangs, screams, gunshots, …) appear together at the same time. Thus, algorithms that could be able to deal with mixed sources and give information about the superposed components should be useful.

At the moment, the algorithms presented in this report could be used as one part of a surveillance system that would also include other alarm detectors, based on image or temperature variations for example. In most cases, the sound detection functionality can be relied on, and already represents an important information. However, for the moment, recognizing what is the source of the noise should be considered as an additional indication, that comes maybe with a confidence factor (*a-posteriori* probabilities, background level). This conclusion means that the envisaged portable system for deaf

people walking in the street is still not ready to work and provide perfectly relying information. However, under specific conditions, this study has shown that very good performance with nearly 100% recognition scores and 95% rejection rates can be reached for impulsive sounds. As a consequence, automatic audio surveillance systems for which time is at disposal to learn most specific sounds of the rooms and to carefully choose the best model for regular use, would correctly work under clean or slightly disturbed environments.

Acknowledgments

This dissertation is the final stage of seven research years passed in the team of Professor Fausto Pellandini, Head of the Electronics and Signal Processing Laboratory, IMT-Uni-Neuchâtel. I really would like to thank Professor Pellandini for the wonderful working environment he has offered to me (and to his whole research group) during all these years, at scientific level but also and particularly at human level. The numerous technical fields I could approach, with a valuable and formative freedom of action, did represent an extremely enriching experience for me. Of course, I would like to thank him for putting his trust in me and for supervising the realization of this thesis.

Also, I'm very grateful to the members of the jury committee for having accepted to read and evaluate this work. I would like first to thank Professor Peter Ryser, Head of the Institute of Production in Microtechnology (IPM) at the Swiss Federal Institute of Technology (EPFL). Professor Ryser is the person at the origin of several audio projects in which I was involved at IMT, and he particularly motivated this study in the sound recognition domain. Then, I would like to thank Professor Heinz Hügli, Head of the Pattern Recognition Group at IMT-Neuchâtel, for investing time to read attentively this text, and for his interesting and precious remarks, observations and discussion.

As the last member of the jury committee, I would like to specially thank our Multimedia Group leader at IMT, Dr. Michael Ansorge. Working with him for all those years was a real pleasure, and all I have learned from him is priceless. I'm most grateful towards him for his intensive participation to the evaluation of this thesis, at a time he was particularly concerned with many other tasks. Thank you Michael, for your constant availability, your continuous support, all the precious comments or suggestions, that finally made this work so much better!

Remembering the beginning of this study, I would like to express all my grateful feelings to Dr. Laurent Besacier, who worked with me, helped me starting in the recognition domain, and gave me the initial impulse I needed, to get rapidly involved into the sound recognition problem. His important contribution to the implementation of the first solutions, and more recently his numerous comments on this text, were most important.

During this period at IMT, I had the opportunity to coach many students who were performing their semester or diploma work at EPFL or Uni-Neuchâtel. This was a most exciting experience from which I could learn a lot. Hence, I would like to thank them all for what they brought to me. I am specially grateful to Mr. Marc-André Schneider, who is now in the IMT team, and who first experienced the problem of real-world background environments, building a significant and most useful database.

All the computer simulations performed during this work were very intensive jobs. Many workstations were simultaneously running, during day and night. Hence, I wouldn't forget to thank our system manager, Mr. Heinz Burri and his occasional prestigious substitute, Dr. Jean-Pierre Amann (Group leader), whose contributions were of great value, saving months and months of work thanks to the perfect system organization. Here I have the opportunity of apologizing to all my colleagues who may have suffer from seeing my name on most of the powerful computers during years!

Then, of course, I would like to mention all my past and present colleagues. Their active support during all these years was most important, on the occasion of interesting technical discussions. I would particularly thank my every day colleagues in the speech and audio research team: Mrs. Giuseppina Biundo, Mrs. Sara Grassi, Mr. Davide Manetti, Mr. Eric Meurville, and Mr. Jean-Luc Nagel.

However, the most important point may be the friendly and encouraging discussions that occurred during the free time we spent together, talking about technical or non-technical subjects. The excellent atmosphere that prevails in the team of Professor Pellandini has contributed with no doubt to strike up friendships with colleagues, and indirectly to the good progress of this thesis. Particularly, I will never forget the early period spent in a real family atmosphere, with Dr. Louisa Grisoni, Mr. Christophe Calame, and Dr. Steve Tanner, in the famous (!) office No 308, at Tivoli 28, our old building. Later, with the arrival of numerous colleagues, my IMT family got wider, with Dr. Dequn Sun, our great singer, Mr. Eric Meurville, my DSP teacher, and Mrs. Catherine Lehnherr, our cheerful secretary. More recently, after that most of those people had turn to new horizons, Mr. Davide Manetti was nearly alone to endure my redaction period, often trying to discuss some problems and catch the attention of an inattentive colleague, who was searching for the correct word in the background! Thanks to all of you for your patience and for your kindness! See you soon!

Finally, I would like to dedicate this thesis to the most important persons in my life: my parents and my close family: my godmother, my grandmothers and my cousins Florence and Christine, Christophe and Juan and their little sons. I would like to tell them that without all the love they give to me, without all their care, support and presence of every day, without all we live together, without those strong binds that unite us since years, without all this, nothing would have been possible!

So thank *you*!

Alain Dufaux

Janvier 2001

References

[Ahle96] Ahlea Systems Corporation, "*Pattern Recognition Toolbox*", [http://www.ahlea.com], Nepean, Canada, 1996.

[Ashi93] T. Ashiya, M. Nakagawa "A proposal of a recognition system for the species of birds receiving birdcalls – an application of recognition system for environmental sound", *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E76-A, No. 10, October 1993, pp. 1858-1860.

[Baum70] L.E. Baum, T. Petrie, G. Soules, N. Weiss, "A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains", *Ann. Math. Stat*., Vol. 41, 1970, pp. 164-171.

[Bent97] P.M. Bentley, J.T.E McDonnell, P.M. Grant, "Classification of Native Heart Valve Sounds Using the Choi-Williams Time-Frequency Distribution", 1995 IEEE-EMBC (*European Molecular Biology Council*) and CMBEC (*Canadian Medical and Biological Engineering Conference*), pp. 1083-1084.

[Bass86] M. Basseville, A. Benveniste, Q. Zhang, "Surveillance d'installations industrielles: démarche générale et conception de l'algorithmique", Publication Interne No. 1010, IRISA, Rennes, France, Avril 1996.

[Besa99] L. Besacier, A. Dufaux, M. Ansorge, F. Pellandini, "Automatic sound recognition relying on statistical methods, with application to telesurveillance", in Proceedings of *COST 254, International Workshop on Intelligent Communication Technologies and Applications, with Emphasis on Mobile Communication*, Neuchâtel, CH, May 5-7, 1999, pp. 116-120.

[Brou00] C.C. Broun, W.M. Campbell, "Robust Out-of-Vocabulary Rejection for Low-Complexity Speaker Independent Speech Recognition", *Proc. of EUSIPCO 2000, European Signal Processing Conference 2000*, Tampere, FI, September 5-8, 2000, pp. 1289-1292.

[Cabe92] R.H. Cabell, C.R. Fuller, W.F. O'Brien, "Identification of Helicopter noise Using a Neural Network" *AIAA Journal*, Vol. 30, No. 3, March 1992, pp. 624-630.

[Camp97] J.P. JR. Campbell, "Speaker Recognition: A Tutorial", *Proceedings of the IEEE*, Vol. 85, No. 9, September 1997, pp. 1437-1462.

[Capp] O. Cappe, "h2m: A set of MATLAB functions for the EM estimation of hidden Markov models with Gaussian state-conditional distributions", ENST/Paris, [http://tsi.enst.fr/~cappe/h2m/index.html].

[Cogg98] K.M. Coggins, J. Principe, "Detection and Classification of Insect Sounds in a Grain Silo, using a Neural Network", *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, Anchorage, AK, Vol. III, 1998, pp. 1760-1764.

[Colo96] J. M. Colombi, T. R. Anserson, S. K. Rogers, D. W. Ruck, and G. T. Warhola, "Auditory Model Representation and Comparison for Speaker Recognition", in [Simp96], pp. 714-719.

[Couv97] C. Couvreur, "*Environmental Sound Recognition : a Statistical Approach"*, PhD thesis, Faculté Polytechnique de Mons, Belgium, June 1997.

[Davi80] S.B. Davis, P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 28, No. 4, August 1980, pp. 357-366.

[Deco84] F. De Coulon, *Théorie et Traitement des Signaux*, *Traité d'Electricité*, Vol. VI, Presses Polytechniques Romandes, Lausanne, CH, 1984.

[Delf98] C. Delfs, F. Jondral, "Classification of Transient Time-Varying Signals Using DFT and Wavelet Packet Based Methods", *Proc. of ICASSP'98, International Conference on Acoustics, Speech and Signal Processing*, Seattle, WA, May 12 - 15, 1998, Vol. 3, pp. 1569 - 1572

[Dell93] J.R. Deller, J.G. Proakis, and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals*, Prentice Hall, NJ, USA, 1993.

[Demp77] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society B*, vol. 39, pp.1-38, 1977.

[Demu98] H. Demuth, M. Beal, *Neural Network Toolbox for Use with Matlab,* Fifth printing, Version 3, January 1998.

[Devi82] P. R. Devijver, J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall, Englewood-Cliffs, NJ, 1982.

[Dufa99] A. Dufaux, L. Besacier, M. Ansorge, F. Pellandini, "Automatic Detection and Classification of Wide-band Acoustic Signals", *Proc. of ASA 99, 137th Meeting of the Acoustical Society of America and Forum Acusticum 99*, Berlin, D, March 14-19, 1999.

[Dufa00] A. Dufaux, L. Besacier, M. Ansorge, F. Pellandini, "Automatic sound detection and recognition for noisy environment", in Proceedings of *EUSIPCO 2000, European Signal Processing Conference 2000*, pp. 1033-1036, Tampere, FI, September 5-8, 2000.

[Embr91] P.M. Embree, B.Kimble, *C Language Algorithms for Signal Processing*, Prentice Hall, New Jersey, 1991.

[Eron00] A. Eronen, A. Klapuri, "Musical Instrument Recognition Using Cepstral Coefficients and Temporal Features" *Proc. of ICASSP'2000*, *International Conference on Acoustics, Speech and Signal Processing,* Istanbul, Turkey, 2000, pp. 753-756.

[Flet40] H. Fletcher, "Auditory Patterns", *Rev. Mod. Phys*., Vol. 12, 1940, pp. 47-65.

[Fuku90] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, USA, Second edition, 1990.

[Ghit94] O. Ghitza, "Auditory Models and Human Performance in Tasks Related to Speech Coding and Speech Recognition*", IEEE Transactions on Speech and Audio Processing*, Vol. 2, No. 1, Part II, January 1994, pp. 115-132.

[Gold76] U. Goldstein, "Speaker-Identifying Features Based on Formant Tracks", *Journal of the Acoustical Society of America*, Vol. 59, No. 1, January 1976, pp. 176-182.

[Gold93] R.S. Goldhor, "Recognition of Environmental Sounds", *Proc. ICASSP'93 International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, USA, 1993, Vol. I, pp. 149-152.

[Gras00b] S. Grassi, L. Besacier, A. Dufaux, M. Ansorge, F. Pellandini, "Influence of GSM Speech Coding on the Performance of Text-Independent Speaker Recognition", *Proc. EUSIPCO 2000, European Signal Processing Conference 2000,* Vol. 1, pp. 437-440, Tampere, FI, September 4-8, 2000.

[Haga95] M.T. Hagan, H. B. Demuth, M. Beale, *Neural Network Design*, PWS Publishing Company, Boston, USA, 1995.

[Harm00a] A. Härmä, K. Palomäki, "HUTear - Matlab toolbox version 2.0", [http://www.acoustics.hut.fi/~aqi/software/HUTear/], 2000.

[Harm00b] A. Härmä, M. Karjalainen, "WarpTB - Matlab Toolbox for Warped DSP", [http://www.acoustics.hut.fi/software/warp/], September 2000.

[Jial99] Jialong He, "Jialong He's Speech Recognition Research Tool", [http://www.speech.cs.cmu.edu/comp.speech/Section6/Recognition/jialong.html], 1999.

[John88] J.D. Johnston, "Transform Coding of Audio Signals Using Perceptual Noise Criteria", *IEEE Journal on Selected Areas in Communications*, vol. 6, No. 2, February 1988, pp. 314-323.

[Joll86] I.T. Jolliffe, *Principal Component Analysis*, New-York, Springer-Verlag, 1986.

[Kaba86] P. Kabal, P. Ramachandran, "The computation of Line-Spectrum Frequencies Using Chebyshev Polynomials", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 34, No 6, pp. 1419-1426, 1986.

[Kil96] D.H. Kil, F.B. Shin, *Pattern Recognition and Prediction with Applications to Signal Characterization*, AIP Press, Woodbury, New York, 1996.

[Kim99] D.-S. Kim, S.-Y. Lee, R.M. Kil, "Auditory Processing of Speech Signals for Robust Speech Recognition in Real-World Noisy Environments", *IEEE Transactions on Speech and Audio Processing*, Vol. 7, No. 1, January 1999, pp. 55-69.

[Kran94] A.A. Kranidiotis, "Human Audio Perception Frequently Asked Questions, [ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/info/HumanAudioPerception], 1994.

[Lebe93] J.-F. Leber, *The Recognition of Acoustical Signals Using Neural Networks and Open Simulator*, ETH Thesis Nr. 10016, Hartung-Gorre Verlag, Konstanz, 1993.

[Leon] Leonardo Software, [http://www.leonardosoft.com], Santa Monica, CA 90401.

[Medd86] R. Meddis, "Simulation of Mechanical to Neural Transduction in the Auditory Receptor", *Journal of the Acoustical Society of America*, Vol. 79, March 1986, pp. 702-711.

[Medd90] R. Meddis, "Implementation Details of a Computation Model of the Inner Hair-cell Auditory Nerve Synapse", *Journal of the Acoustical Society of America*, Vol. 87, April 1990, pp. 1813-1816.

[Moon96] T.K. Moon, "The Expectation-Maximization Algorithm", *IEEE Signal Processing Magazine*, Vol. 13, No. 6, November 1996, pp. 47-60.

[Ober95] S. Oberle, A. Kaelin, "Recognition of Acoustical Alarm Signals for the Profoundly Deaf Using Hidden Markov Models", *Proc. of ISCAS'95*, *International Symposium on Circuits and Systems*, Seattle, USA. 1995, pp. 2285-2288.

[Osha00] D. O'Shaugnessy, *Speech Communications, Human and Machine*, Second Edition, IEEE Press, Piscataway, New Jersey, 2000.

[Osun93] J.A. Osuna, G. S. Moschytz, and T. Roska, "A Framework for the Classification of Auditory Signals with Cellular Neural Networks", *Proc. ECCTD '93*, *European Conference on Circuit Theory and Design*, Davos, CH, August 1993, pp. 51-56.

[Osun95] J.A. Osuna and G. S. Moschytz, "Recognition of Acoustical Alarm Signals with Cellular Neural Networks", *Proc. ECCTD '95*, *European Conference on Circuit Theory and Design*, Istanbul, Turkey, Aug.-Sept. 1995, pp. 797-800.

[Para90] M.J. Paradie, S.H. Nawab, "The Classification of Ringing Sounds", *Proc. ICASSP'90*, *International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, NM, USA, April 1990, pp. 2435-2438.

[Perc00] Perceptual Coding of Digital Audio, *Proceedings of the IEEE*, Vol. 88, No. 4, April 2000.

[Pesu96] L. Pesu, E. Ademovic, J.-C. Pesquet, P. Helisto, "Wavelet Packet Based Respiratory Sound Classification", *Proc. of IEEE Symposium on Time-Frequency and Time-Scale Analysis*, 18 June 1996, Paris, France, pp. 377-380.

[Pori88] A. B. Poritz, "Hidden Markov models: A guided tour", in *Proc. of ICASSP'88, International Conference on Acoustics, Speech and Signal Processing*, New-York, USA, May 1988, pp. 7-13.

[Rabi78] L.R. Rabiner, R.W. Shafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[Rabi89] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", in *Proc. of the IEEE*, vol. 77, No. 2, February 1989, pp. 257-286.

[Rabi93] L.R. Rabiner, B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[Rama99] V. Ramanujam, R. Balchandran, R.J. Mammone, "Robust Speaker Verification In Noisy Conditions By Modification of spectral Time Trajectories", *Proc. ICSPAT'99, International Conference on Signal Processing Application and Technology,* Orlando, USA, November 1999.

[Redn84] R.A. Redner, H.F. Walker, "Mixture Densities, Maximum Likelihood and the EM Algorithm", *Society for Industrial and applied Mathematics (SIAM) Review*, vol. 26, No. 2, April 1984, pp. 195-239.

[Reyn94] D. Reynolds, "Speaker Identification and Verification using Gaussian Mixture Speaker Models", in *Workshop on Automatic Speaker Recognition, Identification and Verification*, Martigny, Switzerland, April 5-7, 1994, pp. 27-30.

[Rice98] B.F. Rice, "Tutorial on Automatic Signal Classification", at the *European Signal Processing Conference (EUSIPCO)*, September 8 - 11, 1998, Island of Rhodes, Greece.

[Riis97] S.K. Riis, A. Krogh, "Hidden Neural Networks: A framework for HMM/NN Hybrids", *Proc. ICASSP'97, International Conference on Acoustics, Speech, and Signal Processing,* April 21-24 1997, Munich, D, pp. 3233-3236.

[Samb75] M. Sambur, "Selection of Acoustic Features for Speaker Identification", *IEEE Transactions on Acoustic, Speech and Signal Processing*, Vol. 23, No. 2, April 1975, pp. 176-182.

[Schn00] M.A. Schneider, "Etude et implantation d'algorithmes de reconnaissance de signaux audio", Diploma work, Institute of Microtechnology, University of Neuchâtel, Switzerland, Semester 1999-2000.

[Sene84] S. Seneff, "Pitch and Spectral Estimation of Speech Based on Auditory Synchrony Model", *Proc. ICASSP'84, International Conference on Acoustics, Speech, and Signal Processing,* San Diego, USA, 1984, pp. 36.2.1-36.2.4.

[Shar70] B. Sharf, "Critical Bands", in *Foundations of Modern Auditory Theory*, Ch. 5, Ed. J.V. Tobias, Academic Press, New-York, 1970, pp. 159-202.

[Shie90] M.K. Shields, C.W. Therrien "A Hidden Markov Model Approach to the Classification of Acoustic Transients", *Proc. ICASSP'90*, *International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, NM, USA, April 1990, pp. 2731-2734.

[Simp96] P. K. Simpson, Editor, *Neural Networks Applications*, IEEE Technology Update Series, IEEE, New York, USA, 1996.

[Slan93] M. Slaney, An Efficient Implementation of the Patterson-Holdsworth Auditory Filter Bank, *Apple Computer Technical Report #35*, Apple computer Inc., 1993.

[Slan94] M. Slaney, "Auditory Toolbox", Apple Technical Report #45, Apple Computer Inc., 1994. [http://fife.speech.cs.cmu.edu/comp.speech/Section1/HumanAudio/auditory.tlbx.html].

[Solt98] H. Soltau, T. Schultz, M. Westphal, "Recognition of music types", *Proc. ICASSP'98, International Conference on Acoustics, Speech and Signal Processing,* Seattle, WA, 1998.

[Stev40] Stevens, S.S. and J. Volkman, "The Relation of Pitch to Frequency", *American Journal of Psychology*, Vol. 53, p. 329, 1940.

[Swet96] D.L. Swets, J.J. Weng, "Using Discriminant Eigen Features for Image Retrieval", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 8, August 1996, pp. 831-836.

[Tamu90] S. Tamura, M. Nakamura, "Improvements to noise reduction neural networks", *Proc. ICASSP'90*, *International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, NM, USA, April 1990, pp. 825-828.

[Theo98] S. Theodoridis, K. Koutroumbas, *Pattern Recognition*, Academic Press, San Diego, London, 1998.

[Uvac88a] B. Uvacek, H. Ye, G.S. Moschytz, "A New Strategy For Tactile Hearing Aids: Tactile Identification of Preclassified Signals (TIPS)", *Proc. ICASSP'88*, *International Conference on Acoustics, Speech, and Signal Processing,* New-York, USA, May 1988, pp. 2500-2503.

[Uvac88b] B. Uvacek, and G.S. Moschytz, "Sound Alerting Aids for the Profoundly Deaf", *Proc. ISCAS'88*, *International Symposium on Circuits and Systems*, Helsinki, FI, 1988, pp. 2615-2618.

[Varg92] A. P. Varga, H. J. M. Steeneken, M. Tomlinson, and D. Jones, "The NOISEX-92 Study on the Effect of Additive Noise on Automatic Speech Recognition", Tech. Rep., DRA Speech Research Unit, 1992.

[Vite67] A.J. Viterbi, "Error Bounds for Convolutional Codes and Asymptotically Optimum Decoding algorithm", *IEEE Transactions on Information Theory*, Vol. IT-13, April 67, pp. 260-269.

[Wold96] E. Wold, T. Blum, D. Keislar, J. Wheaton "Content-based classification, search and retrieval of audio", *IEEE MultiMedia Magazine,* Vol. 3, No. 3, 1996, pp. 27-36.

[Wood92] J.P. Woodard, "Modeling and Classification of Natural Sounds by Product code Hidden Markov Models", *IEEE Transactions on Signal Processing*, Vol. 40, No. 7, July 1992, pp. 1833-1835.

[Yang92] X. Yang, K. Wang, S.A. Shamma, "Auditory Representations of Acoustic Signals", *IEEE Transactions on Information Theory*, Vol. 38, No. 2, March 1992, pp. 824-839.

Appendix A

This appendix is concerned with the determination of transformed features, using the Linear Discriminant Analysis technique (LDA).

The problem is to find the eigenvalues and eigenvectors of the non-symmetric *Fisher Covariance Matrix* J_{wb} in a stable way:

$$J_{wb} = S_w^{-1} S_b \tag{11.1}$$

with:

$$S_b = \sum_{i=1}^{N_c} (\vec{\mu}_i - \vec{\mu}_0)(\vec{\mu}_i - \vec{\mu}_0)^t \tag{11.2}$$

$$S_w = \sum_{i=1}^{N_c} C_i = \sum_{i=1}^{N_c} \left[\frac{1}{K_i - 1} \sum_{k=1}^{K_i} \left(\vec{f}_k^{\,(i)} - \vec{\mu}_i \right) \left(\vec{f}_k^{\,(i)} - \vec{\mu}_i \right)^t \right] \tag{11.3}$$

In those expressions, N_c is the number of considered classes, $\vec{\mu}_0$ is the mean vector of the features over all classes, while $\vec{\mu}_i$ and C_i are the mean vector and covariance matrix inside of each class *i*. The K_i features that represent class *i* are noted $\vec{f}_k^{\,(i)}$.

The solution proposed in [Swet96] is to decompose the problem:

- The diagonalization of S_w is possible because of the symmetrical properties of the covariance matrix. Introducing a diagonal matrix *A* made of the eigenvalues of S_w and an orthonormal matrix *H* filled with its eigenvectors, it comes:

$$S_w = HAH^t \tag{11.4}$$

Then, as the transposed of any orthonormal matrix (*H*) is identical to its inverse, and remembering that the transposed of a diagonal matrix (*A*) is the diagonal matrix itself, calculating

$$\left(HA^{-1/2}\right)^t S_w \left(HA^{-1/2}\right) \tag{11.5}$$

using (11.4), gives:

$$\begin{aligned}\left(HA^{-1/2}\right)^t HAH^t \left(HA^{-1/2}\right) &= \left(HA^{-1/2}\right)^t HAH^t HA^{-1/2} = \\ &= \left(A^{-1/2}\right)^t H^t HAH^t HA^{-1/2} = \left(A^{-1/2}\right)^t AA^{-1/2} = \\ &= \left(A^{-1/2}\right)^t A^{1/2} = A^{-1/2}A^{1/2} = I\end{aligned} \tag{11.6}$$

- Next, the diagonalization

$$\left(HA^{-1/2}\right)^t S_b \left(HA^{-1/2}\right) = U\sum U^t \tag{11.7}$$

is stable because the left member is symmetric (like S_b). Hence, U is orthonormal and $\sum$ is diagonal. From (11.7), S_b can be isolated, multiplying by the appropriate inverse expressions on the left and on the right. It comes:

$$\begin{aligned}S_b &= \left[\left(HA^{-1/2}\right)^t\right]^{-1} U\sum U^t \left(HA^{-1/2}\right)^{-1} = \\ &= \left[\left(A^{-1/2}\right)^t H^t\right]^{-1} U\sum U^t \left(A^{1/2}H^{-1}\right) = \\ &= \left[H^t\right]^{-1} A^{1/2}U\sum U^t A^{1/2}H^{-1} = HA^{1/2}U\sum U^t A^{1/2}H^t\end{aligned} \tag{11.8}$$

The same manipulation over (11.5) leads to:

$$\begin{aligned}S_w &= \left[\left(HA^{-1/2}\right)^t\right]^{-1} I\left(HA^{-1/2}\right)^{-1} = \\ &= \left[\left(A^{-1/2}\right)^t H^t\right]^{-1} I\left(A^{1/2}H^{-1}\right) = \\ &= \left[H^t\right]^{-1} A^{1/2}IA^{1/2}H^{-1} = HA^{1/2}IA^{1/2}H^t = HA^{1/2}UIU^t A^{1/2}H^t\end{aligned} \tag{11.9}$$

where the last transition takes account of the orthonormal property of U.

- Finally, deriving the inverse of S_w from (11.4):

$$S_w^{-1} = \left[HAH^t \right]^{-1} = \left[\left(AH^t \right) \right]^{-1} H^{-1} = HA^{-1}H^t \tag{11.10}$$

the matrix $J_{wb} = S_w^{-1} S_b$ can be calculated using (11.8) and (11.10):

$$\begin{aligned} S_w^{-1} S_b &= HA^{-1}H^t HA^{1/2} U \sum U^t A^{1/2} H^t = \\ &= HA^{-1} A^{1/2} U \sum U^t A^{1/2} H^t = \\ &= HA^{-1/2} U \sum U^t A^{1/2} H^t = \\ &= \nabla \sum \nabla^{-1} \end{aligned} \tag{11.11}$$

where $\nabla = HA^{-1/2}U$ was defined, with its inverse:

$$\nabla^{-1} = \left[HA^{-1/2}U \right]^{-1} = U^t A^{1/2} H^t \tag{11.12}$$

This means that J_{wb} can be diagonalized in a stable way, finding its eigenvalues in the diagonal matrix $\sum$, and its eigenvectors in the matrix $\nabla = HA^{-1/2}U$. The only necessary operations are the diagonalization of S_w, providing H and A, and the diagonalization of $(HA^{-1/2})^t S_b (HA^{-1/2})$, providing U and $\sum$.

Appendix B

The following table specifies the organization of the critical bands, according to the Bark scale (taken from [John88]).

Critical bands [Bark]	Lower Edge [Hz]	Center [Hz]	Upper Edge [Hz]
1	0	50	100
2	100	150	200
3	200	250	300
4	300	350	400
5	400	450	510
6	510	570	630
7	630	700	770
8	770	840	920
9	920	1000	1080
10	1080	1170	1270
11	1270	1370	1480
12	1480	1600	1720
13	1720	1850	2000
14	2000	2150	2320
15	2320	2500	2700
16	2700	2900	3150
17	3150	3400	3700
18	3700	4000	4400
19	4400	4800	5300
20	5300	5800	6400
21	6400	7000	7700
22	7700	8500	9500
23	9500	10500	12000
24	12000	13500	15500
25	15500	19500	21000*

* This value is not official, but it was selected in the experiments.

Appendix C

This appendix shows a typical example of frame-by-frame *a-posteriori* probabilities obtained for the little database (3 classes), with a Bayes classifier, and using the Bark spectrogram features (type B). The signal to recognize (shown on the figure below) belongs to the explosion class, which is the second class of the little database. As a consequence, a good classification should be characterized by important probabilities in the second line of each column, each column corresponding to one frame (from beginning to end of the signal). The frame length is 1024 samples (1/8 overlap).

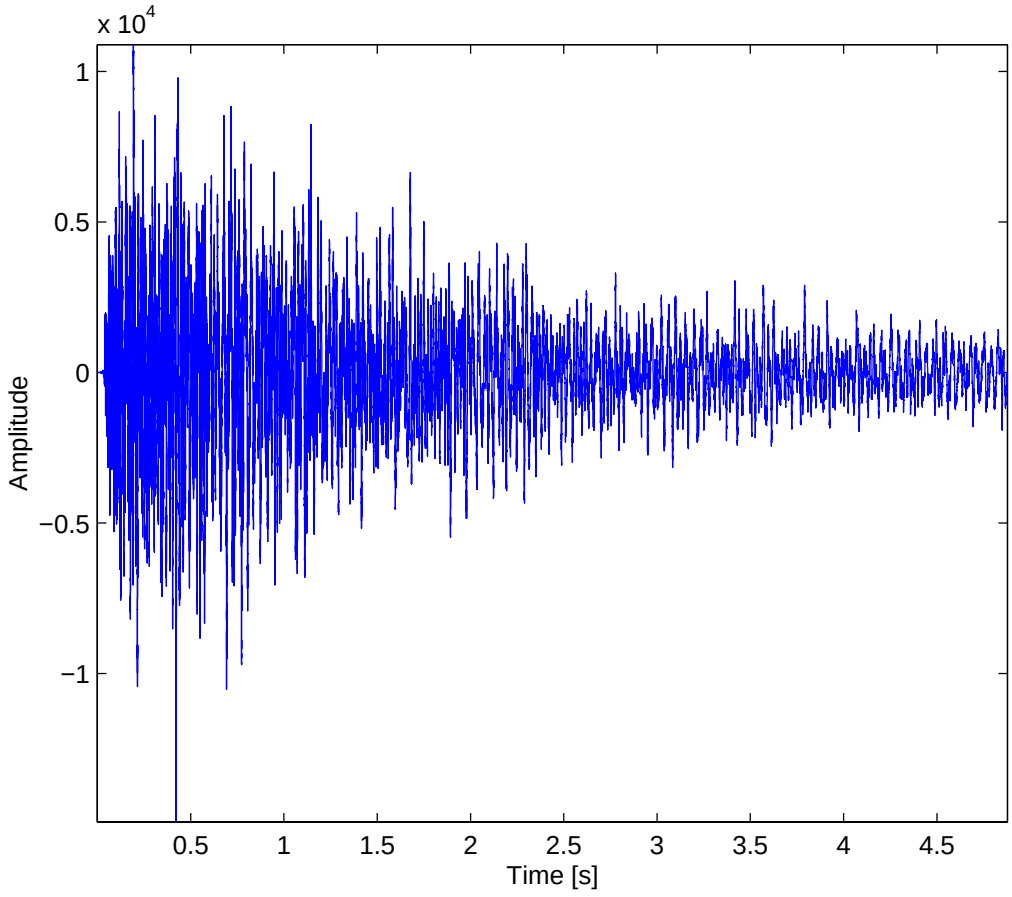


Figure C: *Explosion Signal (Explo056).*

A-posteriori probabilities =

Columns 1 through 7

0.9999	0.9983	0.9984	0.9842	0.9666	0.9964	0.9998
0.0001	0.0017	0.0016	0.0158	0.0334	0.0036	0.0002
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 8 through 14

0.9455	0.6755	0.9549	0.5324	0.3307	0.7065	0.2911
0.0091	0.3244	0.0429	0.4676	0.6685	0.2934	0.7089
0.0455	0.0001	0.0022	0.0000	0.0009	0.0000	0.0000

Columns 15 through 21

0.9132	0.4953	0.0599	0.6044	0.9932	0.6885	0.5676
0.0868	0.5047	0.9401	0.3956	0.0041	0.3114	0.4324
0.0000	0.0000	0.0000	0.0000	0.0027	0.0000	0.0000

Columns 22 through 28

0.5358	0.2383	0.0778	0.2204	0.0727	0.0410	0.1203
0.4642	0.7617	0.9220	0.7796	0.9273	0.9590	0.8797
0.0000	0.0000	0.0002	0.0000	0.0000	0.0000	0.0000

Columns 29 through 35

0.3044	0.3749	0.3139	0.0687	0.1104	0.0075	0.0094
0.6956	0.6251	0.6861	0.9313	0.8896	0.9925	0.9906
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 36 through 42

0.0029	0.0129	0.0007	0.0852	0.0178	0.0500	0.0029
0.9971	0.9871	0.9993	0.9148	0.9822	0.9500	0.9971
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 43 through 49

0.0116	0.0001	0.0030	0.0002	0.0004	0.0006	0.0216
0.9884	0.9999	0.9970	0.9998	0.9996	0.9994	0.9784
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 50 through 56

0.0150	0.0031	0.0306	0.0099	0.0016	0.0013	0.0123
0.9850	0.9969	0.9694	0.9901	0.9984	0.9987	0.9877
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 57 through 63

0.0002	0.0002	0.0003	0.0002	0.0000	0.0001	0.0000
0.9998	0.9998	0.9997	0.9998	1.0000	0.9999	1.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 64 through 70

0.0004	0.0002	0.0046	0.0006	0.0009	0.0017	0.0038
0.9996	0.9998	0.9954	0.9994	0.9991	0.9983	0.9962
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 71 through 77

0.0008	0.0041	0.0007	0.0018	0.0031	0.0322	0.0115
0.9992	0.9959	0.9993	0.9982	0.9969	0.9678	0.9885
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 78 through 84

0.0019	0.0021	0.0009	0.0003	0.0022	0.0001	0.0011
0.9981	0.9979	0.9991	0.9997	0.9978	0.9999	0.9989
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 85 through 91

0.0026	0.0006	0.0008	0.0051	0.0006	0.0111	0.0083
0.9974	0.9994	0.9992	0.9949	0.9994	0.9889	0.9917
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 92 through 98

0.0063	0.0006	0.0010	0.0026	0.0028	0.0007	0.0002
0.9937	0.9994	0.9990	0.9974	0.9972	0.9993	0.9998
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 99 through 105

0.0007	0.0002	0.0010	0.0016	0.0005	0.0004	0.0013
0.9993	0.9998	0.9990	0.9984	0.9995	0.9996	0.9987
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 106 through 112

0.0016	0.0020	0.0001	0.0000	0.0019	0.0000	0.0019
0.9984	0.9980	0.9999	1.0000	0.9981	1.0000	0.9981
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 113 through 119

0.0018	0.0003	0.0000	0.0001	0.0002	0.0001	0.0000
0.9982	0.9997	1.0000	0.9999	0.9998	0.9999	1.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 120 through 126

0.0003	0.0001	0.0000	0.0002	0.0019	0.0022	0.0109
0.9997	0.9999	1.0000	0.9998	0.9981	0.9978	0.9891
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 127 through 133

0.0003	0.0010	0.0009	0.0028	0.0025	0.0000	0.0004
0.9997	0.9990	0.9991	0.9972	0.9975	1.0000	0.9996
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 134 through 140

0.0003	0.0001	0.0002	0.0006	0.0009	0.0024	0.0001
0.9997	0.9999	0.9998	0.9994	0.9991	0.9976	0.9999
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 141 through 147

0.0003	0.0004	0.0005	0.0021	0.0053	0.0001	0.0007
0.9997	0.9996	0.9995	0.9979	0.9947	0.9999	0.9993
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 148 through 154

0.0009	0.0500	0.0003	0.0179	0.0003	0.0005	0.0002
0.9991	0.9500	0.9997	0.9821	0.9997	0.9995	0.9998
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 155 through 161

0.0007	0.0012	0.0004	0.0001	0.0007	0.0008	0.0003
0.9993	0.9988	0.9996	0.9999	0.9993	0.9992	0.9997
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 162 through 168

0.0004	0.0015	0.0063	0.0200	0.0007	0.0013	0.0038
0.9996	0.9985	0.9937	0.9800	0.9993	0.9987	0.9962
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 169 through 175

0.0001	0.0094	0.0338	0.0012	0.0003	0.0002	0.0009
0.9999	0.9906	0.9662	0.9988	0.9997	0.9998	0.9991
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 176 through 182

0.0000	0.0008	0.0012	0.0069	0.0005	0.0196	0.0115
1.0000	0.9992	0.9988	0.9931	0.9995	0.9804	0.9885
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 183 through 189

0.0071	0.0172	0.0130	0.0171	0.0019	0.0168	0.1013
0.9929	0.9828	0.9870	0.9829	0.9981	0.9832	0.8987
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 190 through 196

0.0201	0.0089	0.0004	0.0027	0.0035	0.0347	0.0065
0.9799	0.9911	0.9996	0.9973	0.9965	0.9653	0.9935
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 197 through 203

0.0417	0.1687	0.0004	0.0079	0.0063	0.0101	0.0015
0.9583	0.8313	0.9996	0.9921	0.9937	0.9899	0.9985
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 204 through 210

0.0145	0.2408	0.0098	0.0012	0.0988	0.0052	0.0001
0.9855	0.7592	0.9902	0.9988	0.9012	0.9948	0.9999
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 211 through 217

0.0050	0.2033	0.0067	0.1028	0.0043	0.0040	0.0569
0.9950	0.7967	0.9933	0.8972	0.9957	0.9960	0.9431
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 218 through 224

0.0520	0.0043	0.0183	0.0001	0.0008	0.0088	0.0102
0.9480	0.9957	0.9817	0.9999	0.9992	0.9912	0.9898
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 225 through 231

0.0500	0.0260	0.0560	0.0726	0.0120	0.2937	0.0305
0.9500	0.9740	0.9440	0.9274	0.9880	0.7063	0.9695
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 232 through 238

0.0608	0.0327	0.0381	0.1126	0.4403	0.0526	0.5651
0.9392	0.9673	0.9619	0.8874	0.5597	0.9474	0.4349
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Columns 239 through 245

0.2834	0.4217	0.3086	0.3051	0.2809	0.6016	0.1205
0.7166	0.5783	0.6914	0.6949	0.7191	0.3984	0.8795
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Appendix D

This section illustrates the different forms of statistical distributions that can be obtained using GMM models. Both 3-D surfaces and level-curves are shown on Figures D.1, D.2, and D.3, for GMM PDF built using respectively 1, 2 and 4 gaussian components. All curves represent the same set of training signals. In this way, the evolution of the distribution forms, being more and more specific to the concerned training signals can be observed and compared. The features scheme is the Bark spectrogram (type B, N=1024). A PCA+LDA reduction process to 2 dimensions is performed, to allow such representations.

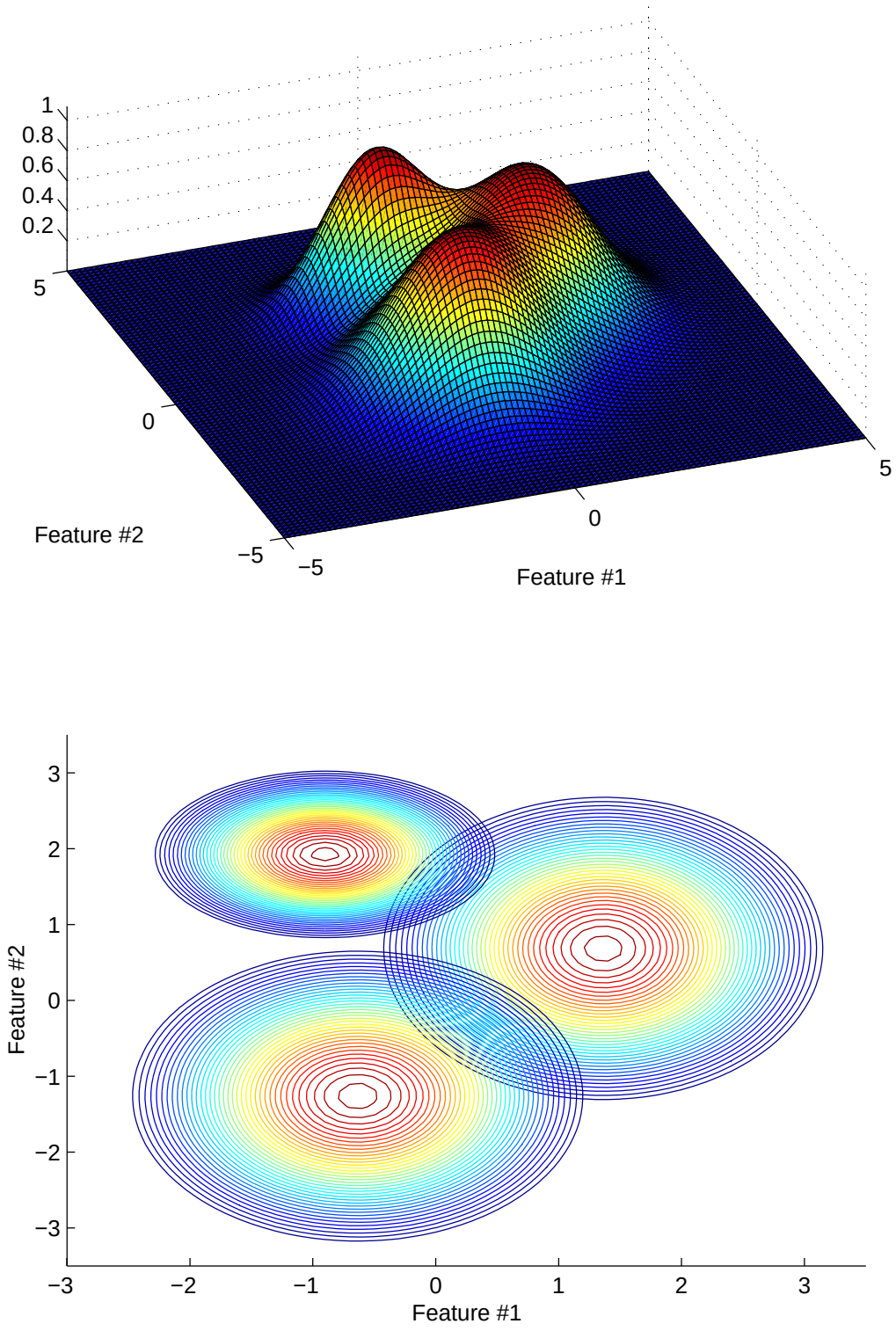


Figure D.1: *One gaussian per class*

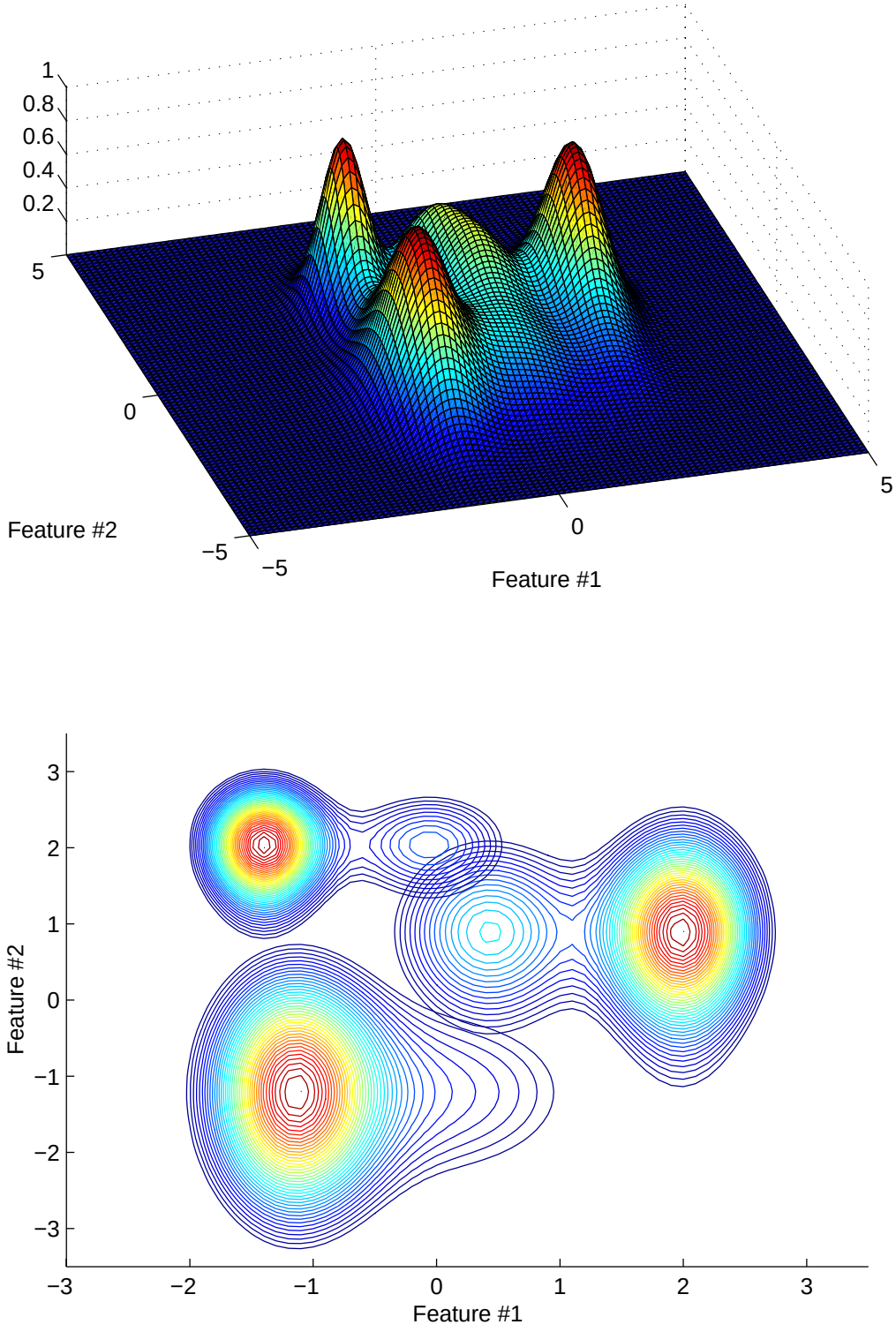


Figure D.2: *Two gaussians per class*

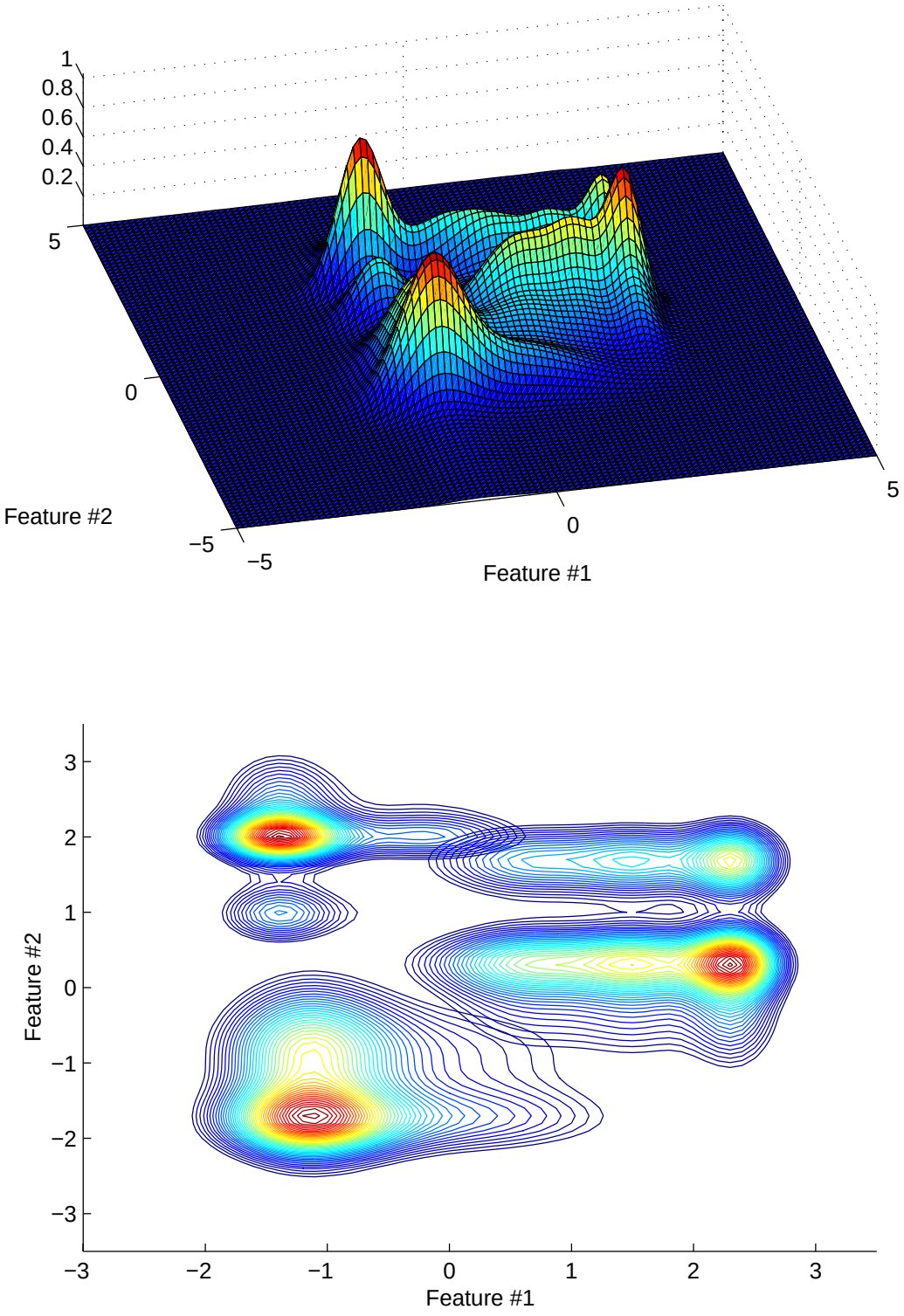


Figure D.3: *Four gaussians per class*

Alain Dufaux
Av. Belmont 23 C
1820 Montreux

Phone: +41 32 755 74 45
Fax: +41 32 755 74 01
E-mail: alain.dufaux@a3.epfl.ch

Alain Dufaux

Native of Montreux, Switzerland; Born December 23th, 1968; Unmarried.

Languages:

French, English, German

Educational Degree:

1988-1994: Swiss Federal Institute of Technology, Lausanne (EPFL),

Dipl. Electrical Engineer EPFL

Working Experience:

Since 1994: Chair of Electronics and Signal Processing, Institute of Microtechnology, University of Neuchâtel, Switzerland.

Research Assistant

Domains of Research and Supervision of Student Projects:

- Pattern recognition techniques applied to automatic classification of impulsive sounds and automatic speaker recognition.
- Speech and audio compression techniques: Algorithm design and DSP implementation.

Alain Dufaux
Av. Belmont 23 C
1820 Montreux

Phone: +41 32 755 74 45
Fax: +41 32 755 74 01
E-mail: alain.dufaux@a3.epfl.ch

Main Publications:

- A. Dufaux, L. Besacier, M. Ansorge, F. Pellandini, "Automatic sound detection and recognition for noisy environment", in Proceedings of *EUSIPCO 2000, European Signal Processing Conference 2000*, Tampere, FL, September 5-8, 2000.

- S. Grassi, A. Dufaux, L. Besacier, M. Ansorge, F. Pellandini, "Speaker Recognition on Compressed Speech", in Proceedings of the *International COST 254 Workshop on Friendly Exchanging Through the Net*, pp. 117-122, Bordeaux, F, March 23-24, 2000.

- L. Besacier, A. Dufaux, M. Ansorge, F. Pellandini, "Automatic sound recognition relying on statistical methods, with application to telesurveillance", in Proceedings of *COST 254, International Workshop on Intelligent Communication Technologies and Applications, with Emphasis on Mobile Communication*, Neuchâtel, CH, May 5-7, 1999.

- D. Manetti, A. Dufaux, E. Meurville, M. Ansorge, F. Pellandini, P. Ryser, D. Wieser, "Audio Coder for Telesurveillance Applications with Real Time Implementation on a Multimedia DSP", in Proceedings of *COST 254, International Workshop on Intelligent Communication Technologies and Applications, with Emphasis on Mobile Communication*, Neuchâtel, CH, May 5-7, 1999.

- A. Dufaux, L. Besacier, M. Ansorge, F. Pellandini, "Automatic detection and classification of wide-band acoustic signals", in Proceedings of *ASA 99, 137th Meeting of the Acoustical Society of America and Forum Acusticum 99*, Berlin, G, March 14-19, 1999.

- S. Grassi, A. Dufaux, M. Ansorge, F. Pellandini, "Efficient Algorithm to Compute LSP Parameters from 10-th order LPC Coefficients", ICASSP'97, *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 3, pp. 1707-1710, Munich, G, April 21-24, 1997.

- A. Dufaux, "Anatomie et Physiologie des Radios Locales Suisses", *"Homme, Technique, Environnement (HTE)"* Project, EPFL Internal Report, 1994, "Prix de la Commune d'Ecublens" for EPFL Academic Year 1994.