"Planes, trains and automobiles"

Detection and classification of vehicles by means of sound recognition

Tom-Erik Roos
1071262

Master Thesis in Artificial Intelligence
July 2nd, 2003

Supervisors:

- Dr. T.C. Andringa, *Sound Intelligence*
- Dr. G.P. van den Berg, *Natuurkundewinkel*
- Dr. J.E.C. Wiersinga-Post, *KI/RuG*
- Prof. dr. L.R.W. Schomaker, *KI/RuG*

Kunstmatige Intelligentie
Rijks*Universiteit* Groningen

"Planes, trains and automobiles"

Detection and classification of vehicles by means of sound recognition

Tom-Erik Roos

Table of Contents

ABSTRACT

This thesis documents the development of a vehicle detector based on sound recognition. It starts with an overview of current vehicle sound analysis and its shortcomings. Then a new approach is introduced: a sound source separation and recognition tool based on the principles of human hearing. The research question is whether this technique can be applied to vehicle detection and classification. We have answered this question by implementing a system that performs these tasks, and by evaluating its results. During the course of this project commercial and governmental interest arose and three specific applications were derived from a general system: a vehicle detector for busy city streets, an airplane detector and a vehicle detector for security and monitoring purposes. All the systems, though not fully optimized, show good detection scores. The last application even has such good, commercially acceptable, detection and classification results, that it has been turned into a fully-fledged system that has entered the beta-testing stage.

The research has been conducted at the company Sound Intelligence from September 2002 until July 2003, under supervision of Dr. T.C. Andringa.

SAMENVATTING

In dit afstudeerverslag staat het ontwikkeltraject beschreven van een voertuigherkenner gebaseerd op geluid. Het verslag begint met een overzicht van de huidige geluidsanalysetechnieken en de tekortkomingen daarvan. Daarna wordt er een nieuwe aanpak geïntroduceerd: een analysetechniek om geluidsbronnen te scheiden en te herkennen die gebaseerd is op de werking van het menselijk gehoor. De onderzoeksvraag is of deze techniek toegepast kan worden op voertuigdetectie en -classificatie. We hebben deze vraag beantwoord door een systeem te implementeren dat deze taken uit kan voeren en door de resultaten daarvan te evalueren. In de loop van dit project ontstond er interesse vanuit de overheid (RIVM) en diverse bedrijven. Vanuit een algemeen systeem werden drie specifieke applicaties ontwikkeld: een voertuigdetector voor drukke stadsstraten, een vliegtuigherkenner en een systeem voor beveiligingstoepassingen. Hoewel ze nog niet volledig geoptimaliseerd zijn, geven alle systemen goede detectiescores. De laatste applicatie geeft zelfs zulke goede, commercieel acceptabele, detectie- en classificatieresultaten dat er een systeem van gemaakt is dat klaar is om beta-getest te worden.

Het onderzoek is uitgevoerd bij het bedrijf Sound Intelligence, van september 2002 tot en met juni 2003, onder begeleiding van Dr. T.C. Andringa.

1. Introduction

In the 1979 'Stiltegebieden' Act, the Dutch government declared several rural parts of the country to be 'quiet areas'. In these areas only *area specific sounds* should be noticeable, thereby excluding most sounds caused by humans. Sounds that are not area specific, such as highway and airplane noise, are considered a form of pollution and are unwanted. The Dutch quiet areas areas are spread all over the country, even in the densily populated Randstad, and vary in size from several square kilometers to the whole Wadden Sea.
Quiet areas are valued highly by the public, as a recent study from Stichting Natuur en Milieu (the Foundation for Nature and the Environment) indicates (Berends 2002): for 85 per cent of the 1500 respondents silence was one of the main reasons for making trips into nature.
Unfortunately, in most quiet areas airplanes and cars can be heard, from 10 up to 50 per cent of the time. Also, trains, motorcycles, tractors and boats are frequent sources of noise (Van den Berg 2002a).
Governmental and other agencies try to maintain the quality of quiet areas. In order to do this, they need information about the kinds of disturbances. This knowledge can later be used in for instance new legislation that reduces sound emissions. There are two ways to obtain this kind of information: calculation and measurement. Both face some serious problems: the former works with models and cannot account for unpredictable sound sources on the spot while the latter involves human effort, which makes it expensive when used frequently.
This project seeks to solve this dilemma by moving one step further. Sound Intelligence (SI for short), a spin-off company from the University of Groningen, has developed a model of the human auditory system, originally for speech recognition purposes, that can be used to analyze any kind of sound. Perhaps the application of this model can result in a system that automatically recognizes disturbances and, for instance, sends out an alert. That way, the quiet area maintenance cycle will become faster, simpler and cheaper.
The goal of this project was originally to find out if such a system was possible. However, within the time available in this project applications have been limited to relatively nearby, loud sources. The obtained results can therefore not be used directly for far-off sources such as usually occur in quiet areas.

This report starts with an overview of current practice in the sound measurement and calculation field. Then the SI techniques are introduced and the research question is stated. This report further documents the development of several vehicle detection systems, and the results that were obtained. It is rounded off with conclusions and recommendations for future research.

2. Current practice

Owing to the increased awareness of noise as a form of pollution, (government) agencies are actively trying to analyze the sources of possible disturbances and developing new policies to counter these. In their research they make extensive use of existing measurement and calculation methods, most of which are firmly grounded in environmental law. However, these techniques have been developed for analyzing a single, known noise source over a long period of time. As we will see, they are not so suitable when multiple sources -some of them unknown- are present.

CURRENT METHODS OF MEASUREMENT AND CALCULATION STANDARDS

While the goal of this research project is to develop a new approach to vehicle sound measurement, it will certainly be useful to take an in-depth look at the current measurement and calculation practices. This is done, first of all, to obtain a reference, a (hopefully) reliable framework to compare our own methods and results to. Secondly, the shortcomings of the current practice in the field of vehicle sounds mentioned before are the central points of attention for our method. And finally, a lot can be learned from these 'vintage' methods, which have been perfected over the years to meet their specific purposes.

- First, it will be explained how sound, which is defined as air pressure variations within the frequency range that can be perceived by normal-hearing humans, is measured.
- Most of the agencies that keep track of noise of in the Netherlands (e.g. for sake of environmental law enforcement), do so by *calculating* noise levels, a method far less expensive than on-the-spot measuring. How this is done for three very important sources of noise, will be described next. Traffic noise calculation is explained in some detail, while train and airplane noise calculations (which are very much alike) are described succinctly.
- The final two paragraphs address the methods applied by an environmental agency (RIVM, Rijksinstituut voor Volksgezondheid en Milieu: National Institute of Public Health and the Environment) who, as a part of their monitoring the overall quality of the environment, measure noise levels and have tried to do so in quiet areas as well. It will be explained why their results were unsatisfactory, and concluded what we can learn from that experience.

Measuring sound

When measuring sound, all that is measured are air pressure variations within the normal human frequency range (20 Hz to 20 kHz, Nooteboom and Cohen 1995). The following relationship holds for a given frequency: the larger the pressure amplitude, the louder the perceived sound. In humans this relation is perceived logarithmically: doubling the sound level causes only a small, linear increase (approximately 3 dB) in the perceived loudness. Therefore, the measured air pressure variations are scaled logarithmically in order to obtain a valid result. Also, they are divided by a constant reference value, so that for an 'average human' the sound level is zero when the sound is just audible.

Human hearing is optimal for frequencies in the mid-range (1 to 6 kHz, see figure 1). Lower and higher frequencies are perceived less loud. This effect is taken into account by weighting the different frequency components according to an approximation of the human frequency sensitivity curve (the so-called 'A weighting').

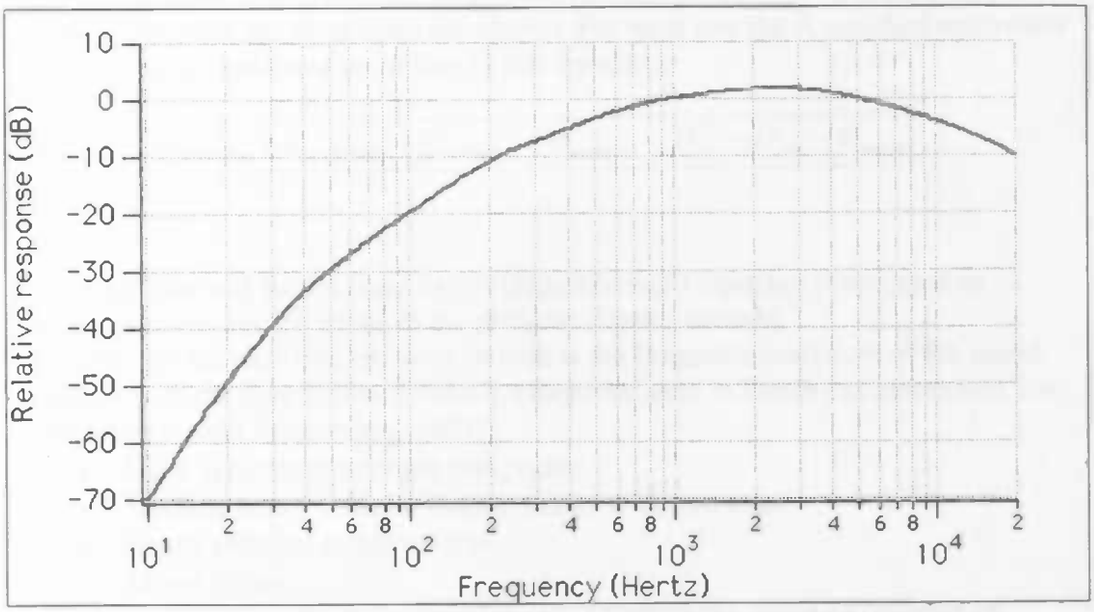

Figure 1: The dB(A) weighting curve

From: http://www.ecse.monash.edu.au/ucourses/ECE3802/Lectures/Auditory/Loudness.html

L. Bahr, 2000

The sound level is measured –and averaged- over a period T. Frequently used values for T are 1 hour, 12 hours, 1 day and 1 year.
In equation form:

$$L_{A,eq} = 10 \times^{10} \log \frac{1}{T} \int_{t=0}^{T} \left(\frac{P(t)_A^2}{P_0^2} \right) dt$$

Equation 1

$P(t)_A$ is the A weighted air pressure at time t. P_0 is the *reference pressure*, 2.0×10^{-5} Pa. $L_{A,eq}$ is called the 'equivalent sound level' and is expressed in A weighted decibels, dB(A).
The averaging step in calculating the sound level has two important consequences:

- Information about individual events is lost. The $L_{A,eq}$ can only be used to characterize the overall sound level.
- Because of the averaging, extremely loud events (or events that occur very close to the microphone) of only a few seconds can heavily influence the $L_{A,eq}$ over 1 hour or even longer periods.

Calculating traffic noise

In calaculating traffic noise, the road from which the immission is determined is divided into one or more 'drive lines'. These can correspond to the road as a whole or individual lanes. The greater the difference (in surface, number of lanes etc.) between parts of the road, the more lines are chosen. For each line the A weighed equivalent sound level is calculated according to this equation:

$$L_{A.eq} = E + C_{surface} + C_{crossroads} + C_{reflection} - D_{distance} - D_{air} - D_{soil} - D_{meteo}$$

Equation 2

The additions and subtractions in this (logarithmical) equation correspond to multiplications and divisions in the physical (linear) domain.
E stands for the total line emission, which is the (logarithmical) sum of the sound emissions of the four different vehicle categories used in Dutch environmental law (Moerkerken and Middendorp 1981):

- Light vehicles (passenger cars, vans).
- Medium heavy vehicles (buses, trucks with two axes).
- Heavy vehicles (other trucks).
- Motor cycles

For each category,these emission values are based on the intensity (number of vehicles per hour) and the average velocity.

- $C_{surface}$ is a correction factor for the type of road surface. An blacktop road has a low $C_{surface}$ value, a cobblestone road a very high value.
- $C_{crossroads}$ is a correction factor that takes into account the effect of a crossroads, should there be one, on the traffic on the road of interest.
- $C_{reflection}$ is included when there are large, acoustically hard surfaces (e.g. a wall), on the other side of the road reflecting traffic emissions.
- $D_{distance}$ corrects for the diminishing of the sound level as one moves further away.
- D_{air} is measure of the damping caused by the air.
- D_{soil} is subtracted when there is a type of soil near the road that may absorb some of the sound.
- D_{meteo} is a factor that corrects for average wind conditions.

When the individual drive line emissions have been calculated, they are added (logarithmically) to obtain a measure for the whole road. This is done during daytime and during nighttime. To the night measure an 'annoyance factor' corresponding with 10 dB(A) is added. The highest of the two measures is chosen to reflect the 24 hour measure, a very important representation of the amount of traffic noise.

Calculating train noise

Train noise calculations are based on emission values that are, in turn, based on the type of railroad track, the type of train, its velocity and whether its brakes are applied. The emission values are calculated over a so called "emission traject" (a stretch of railroad track of where the mentioned variables are more or less constant) and

averaged per hour. The calculations are based on the former state railroad company's timetables and directives (such as when a train should start to brake etc.).
The calculation standard for railway noise distinguishes nine train categories, mainly on basis of the type of brakes and the kind of engine. Braking causes a lot of extra noise which is taken into account by adding a second emission term. The complete equation is as follows:

$$E = 10 \times^{10} \log\left(\sum_{c=1}^{9} 10^{\frac{E_{nr,c}}{10}} + \sum_{c=1}^{9} 10^{\frac{E_{r,c}}{10}}\right)$$

Equation 3

In which E is the total emission for the traject of interest, c stands for the train category, $E_{nr,c}$ is the emission per type of train in the non-braking condition, and $E_{r,c}$ the emission for braking trains. To this emission damping and correction factors are applied as in eq. 2.

Calculating and measuring air traffic noise

In the Netherlands, aircraft noise is expressed in Kosten Units (named after Professor Kosten, they are called Kosten-Eenheden in Dutch, abbreviated to KE) and calculated using the following equation:

$$B = 20 \times \log \sum npf \times 10^{\frac{LA}{15}} - 157$$

Equation 4

B is the noise load in KE; the summation takes place over one year. Of each aircraft overflight during a year two important variables are taken into account: its maximum A weighted sound level (LA) and its Night Penalty Factor (npf), which depends on the time of the day. From 11 p.m. to 6 a.m., the npf equals 10, whereas during most of the day it equals 1.

The RIVM approach: measuring highway and railroad noise

RIVM monitors sound levels in the Netherlands mostly by calculating immissions from highways, railroad tracks, industrial areas etc. and adding these up to obtain a noise contour map for the whole country. However, they also measure sound levels at four specific locations, namely near a highway, a railroad track, a military airfield and in a busy city street, to keep their calculations valid and to measure changes in sound immissions. These shifts can be due to new policies and techniques (more silent tires, railroad material, asphalt etc.) and changing transport volumes.
The measurements are made alongside the A2 highway from Utrecht to Amsterdam and alongside the railroad between these two cities, near Breukelen. The microphones are positioned very close to the sources (approximately 20 meters), in order to minimize noise from other possible sources.

RIVM measures:

- The $L_{A,eq}$.
- The L_{95}, which stands for the 95th percentile of the sound level, meaning the sound level that is exceeded 95 per cent of the measured time.
- The $L_{A,max}$, the maximum sound level.
- The SEL. The three previous values are computed on a per-hour basis. The SEL (Sound Energy Level), on the other hand, is a (logarithmical) measure of the total energy content of *a single vehicle passage*. In formula:

$$SEL = 10 \times^{10} \log\left(\int_{passage} I(t)dt \right)$$

Equation 5

I(t) is the quotient of the square air pressure and the square reference pressure (cf. eq. 1).

The RIVM approach: measuring quiet area noise

For several months in 2000, RIVM took measurements in a quiet area (Zegveld) near Woerden, province of Zuid-Holland. The $L_{A,eq}$ was measured on a daily basis, as well as the L_{90}. It was found that the area was very noisy on working days: occasionaly sound levels of 58 dB(A) were reached, mainly due to agricultural activities nearby. These local sources caused the measured sound level to differ greatly from the model values. Therefore, not all of the sound could be explained by the official models (which do not take local sources into account). This made it impossible to, for instance, study the small but noticeable effects of aircraft noise. It was concluded that measuring in only one place in a quiet area was not practical with the method used and the Zegveld location was subsequently abandoned.

CONCLUSION: POSSIBLE IMPROVEMENTS

It is clear that in quiet areas current practice (both measuring and calculating) fails where human hearing does not. Human listeners are able to separate sound sources quite easily. As for the problems occurring with the equivalent sound level, humans can distinguish individual passages and are not disturbed by irrelevant but loud sources. For these reasons, possible improvements may be located in the application of knowledge about human hearing.
In the following chapters we will take an in-depth look at a human-based sound analysis technique and the way it can be applied to both natural and non-natural sounds.
It would seem that the combination of existing measurement methods and a human-based sound recognition system is ideal. A possible application is to have the system account for local sources, which can then be corrected for in long-term measurements such as the $L_{A,eq}$. After this correction, the measurements are compatible with, for instance, the RIVM models and can be used for validation purposes again.

3. Theoretical foundations

CONTINUITY PRESERVING SIGNAL PROCESSING

This overview of CPSP (Andringa 2002a, 2002b) will be based on speech recognition, as speech has been the driving force behind its formulation. A distinguishing feature of CPSP is that it separates signal parts of interest (e.g. speech) from any kind of noise and thus facilitates further analysis. Here, noise is defined as everything that is not part of the signal as it was emitted by the sender. In traditional speech recognition, there is no comparable pre-processing step. Noise is simply assumed to be non-existent or to have characteristics that are known.
A second unique CPSP feature is that the assumptions concerning the signal, the noise and (the acoustics of) the environment are kept as weak as possible. Because of this, a CPSP based system will be able to function in a great number of different circumstances.
In CPSP, a signal is assumed to consist of a number of *signal components*: physically coherent signal parts whose frequency, energy and phase develop continuously through time. For each part an onset (the component emerges, it rises above the background noise), often a continuous part, and an offset (when the component can no longer be reliably detected) can be found. Recognizing the components is possible because the physical properties of the source are reflected in the signal. For a good example, see figure 2. In this figure, the fundamental and its overtones are signal components whose frequencies are determined by the length of the tube.

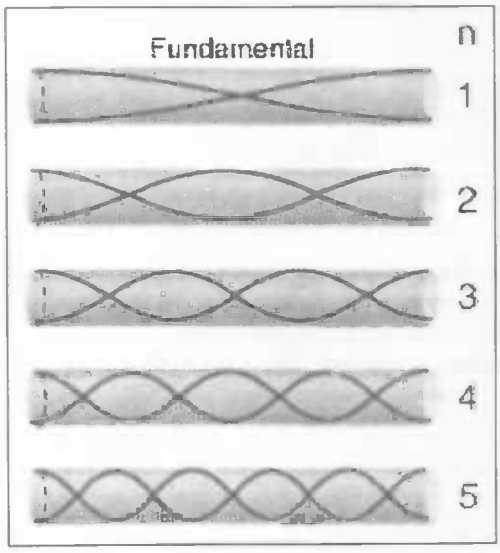

Figure 2: Resonance in a hollow tube with both sides open

The length of the tube determines which fundamental frequency fits in. All the overtones of the fundamental (n=1) fit in as well (in this figure n=2 to n=5).

From: http://hyperphysics.phy-astr.gsu.edu/hbase/waves

For the preprocessing of the signal a model of the human inner ear (most notably the cochlea, see figure 4) is used. Each segment of this "artificial ear" is maximally sensitive to a certain frequency. Because neighboring segments influence eachother,

continuity in time and place is preserved. As each segment of the cochlea corresponds to a frequency range, continuity in the frequency domain is preserved as well. The model is, like the human auditory system, continuous in time because exact frequency measurements can never be instantaneous.

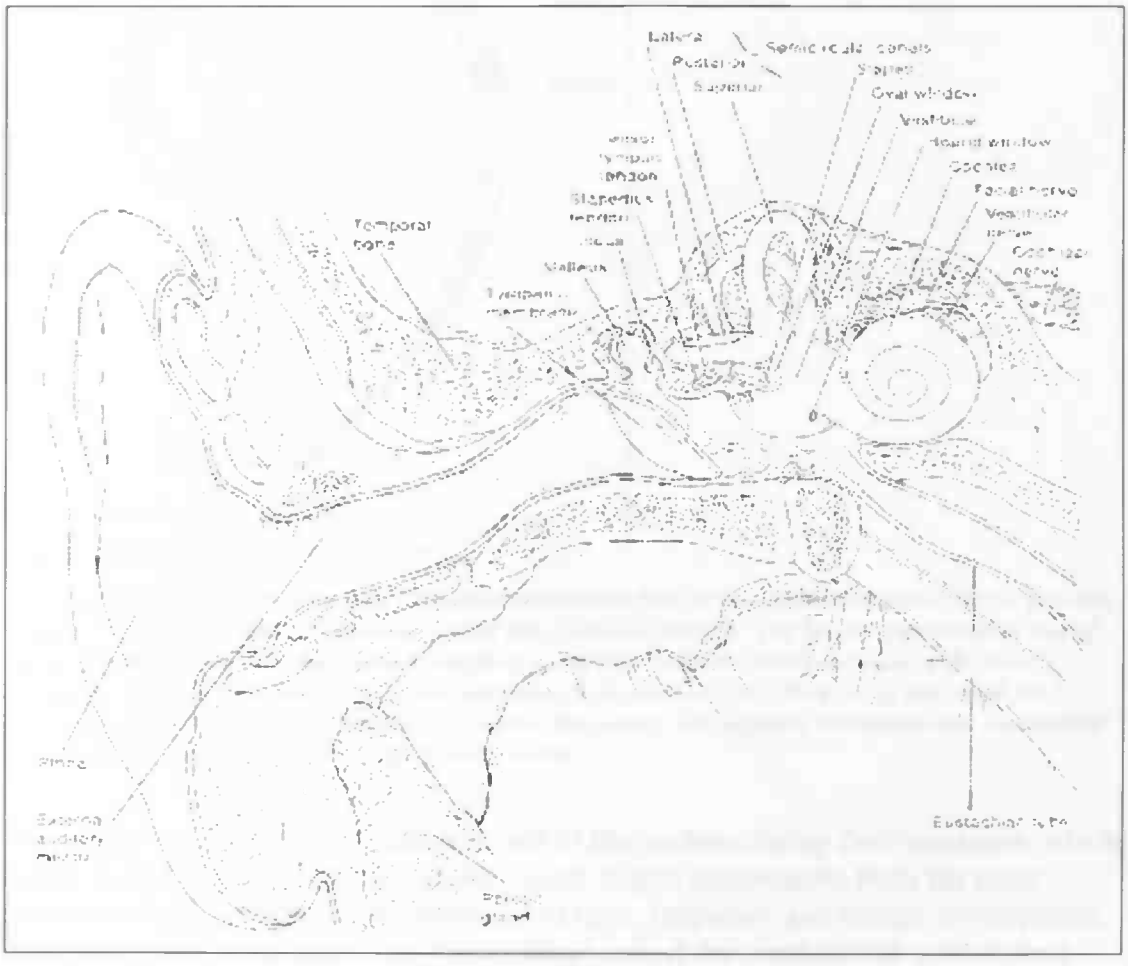

Figure 3: The human auditory system

Sound waves enter the ear via the eardrum (pinna) and are transmitted through the external auditory meatus to the tympanic membrane, which then starts to vibrate. These vibrations are mechanically transmitted and amplified by the ossicles (three connected tiny bones called hammer, anvil and stirrup). Then the sound pressure is transmitted to the oval window, which starts to vibrate and thereby causes the fluid in the cochlea to move.

M. Brödel, 1935

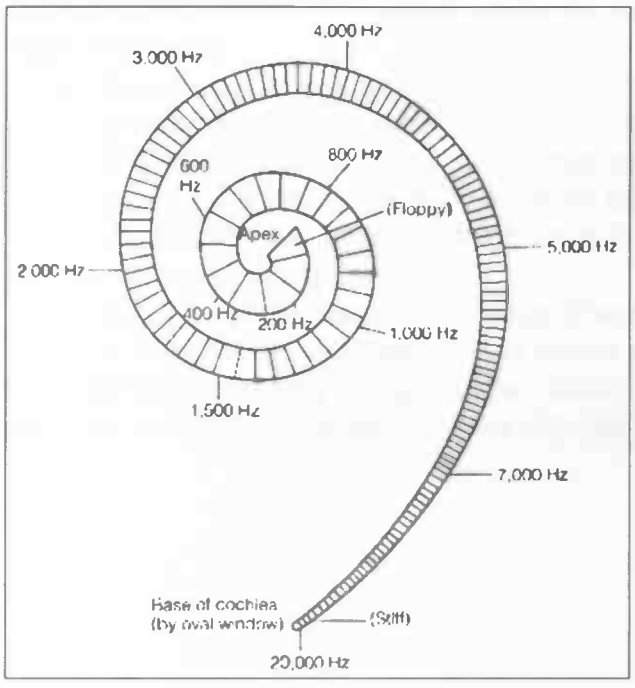

Figure 4: The human basilar membrane

Movement of the oval window causes the uncompressable fluid in the cochlea to move. This is possible because the round window at the other end of the cochlea is flexible. The basilar membrane is one of the membranes that divide the cochlea lengthwise and that starts to vibrate because of the fluid's movement. Its stiffness is not constant but decreases from base to apex. Because of this, each part (segment) of the membrane is sensitive to another frequency. The segment excitations are transmitted to the brain as electric signals by the cochlear nerve.

Strong signal components dominate a part of the cochlea during their existence, which makes it possible to detect them in the signal. Signal components from the same source, such as harmonics, are correlated in time, frequency and energy development. Therefore these components can be combined and, if the combination is consistent with knowledge about a certain class of sounds (for example words or bird songs), the components have been classified.

EXTENDING CPSP

The principles behind CPSP can be used to detect and classify any kind of sound source. It is important to have sufficient knowledge of the physical properties of the sound sources. These properties are reflected in the signals they produce and can therefore be used in the classification process. When a selection of evidence has been made, in other words when a signal part has been assigned to a source in a consistent way, the selection has been detected and classified at the same time.
In the next chapter an overview is given of properties of (a subset of) both natural and unnatural sounds that are relevant for vehicle detection.

One could ask why we would use a *human* cochlea model. For speech recognition, this choice is logical because human performance and communication with humans are its goals. But for (vehicle) sound recognition, using a human cochlea is less obvious. Perhaps there are animal cochleas that have frequency characteristics that are

much better suited for our purposes. However, using a non-human cochlea has three major drawbacks:

- First we would have to find out which cochlea suits our needs best. It could even be that the best cochlea for car detection is another than for truck detection. Researching this would take too much time for this project.
- There is a good chance that there is no model available of the particular cochlea we need. Then we would have to build it ourselves, another time-consuming enterprise.
- Using a non-human cochlea makes it hard to validate the output of the model. It cannot be reliably compared to that of a human listener.

Our conclusion is to keep using a human cochlea model, so we can build upon previous work in this area and compare our results with human performance.

4. Practical foundations

Most of the knowledge in this chapter has been acquired from Frits van den Berg (Van den Berg 2002b).
Many sound sources, e.g. a bird's throat or a hollow reed, contain a cavity. When energy is added to such a cavity, it causes the air inside to start vibrating. The form and size of the cavity determine which frequencies are amplified, or where resonances occur (see figure 2). Sound sources like these behave like physical second order systems (which means their properties can be described by second order differential equations), and an important measure in analyzing the sound spectra of these systems is the Q value: the quotient of the resonance frequency of the sound and the width of the spectral peak at half of the maximum energy level (see figure 5). Therefore, Q is a measure of the resonance width. The narrower the peak, the more specifically the system reacts to the resonance frequency, whereas the broader the peak, the more neighboring frequencies contribute as well.

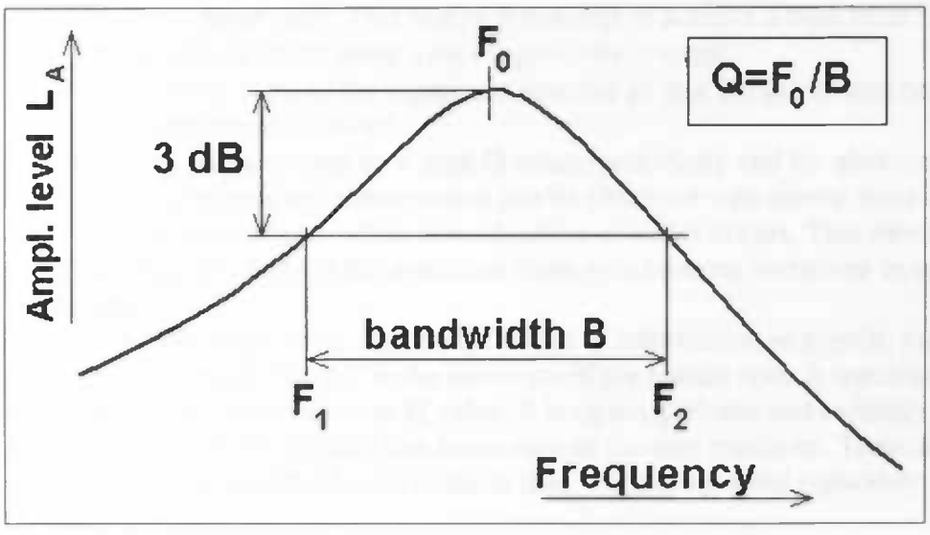

Figure 5: The calculation of the Q value of a frequency peak

From: http://mmd.foxtail.com/tech/qval.html

J. Liljencrants, 2002

KNOWN PROPERTIES OF VARIOUS SOUND SOURCES, BOTH NATURAL AND NON-NATURAL

Wind-induced (background) sounds

Some of the most important ambient sounds are caused by wind, which manifests itself in a couple of ways: on the microphone as a low frequency noise, either in gusts or continuous, and though for example rushing reeds and rustling leaves. Wind sound has a spectrum which is described best by the function:

$$E(f) = \frac{1}{f^3}$$

Equation 6

In this kind of spectrum the lower frequencies contain most of the energy.

Natural, communicative sounds

By natural, communicative sounds we mean sounds that are caused by animals and humans, with the purpose of conveying information. These kinds of sounds are typically characterized by:

- *Noise robustness.* The most important parts contain most of the total energy and are transmitted by frequencies that are perceived best by the auditory system of the recipients, given the acoustic environments the individual communicates in.
- *Narrow bandwidth.* This makes it possible to achieve a high local signal-to-noise ratio without using a lot of -precious- energy.
- *Repetition.* Parts of the signal are repeated so that the sender can be more sure they are indeed received.

Bird songs are characterized by a high Q value, periodicity and the absence of harmonics. An interesting phenomenon can be observed with crows: their croaking is aperiodic and narrowband, while *comodulation of bands* occurs. This means that different frequency parts in the spectrum show synchronous variations in their amplitudes.
When one looks at the most important bearers of information in speech, viz. the formants (the typical 'lumps' in the envelope of the human speech spectrum; see figure 6), speech has an average Q value. It is (quasi)periodic and certainly the voiced parts (such as vowels) contain few harmonics at formant positions. These harmonics themselves are narrowband contributions that, when considered separately, have high Q values.

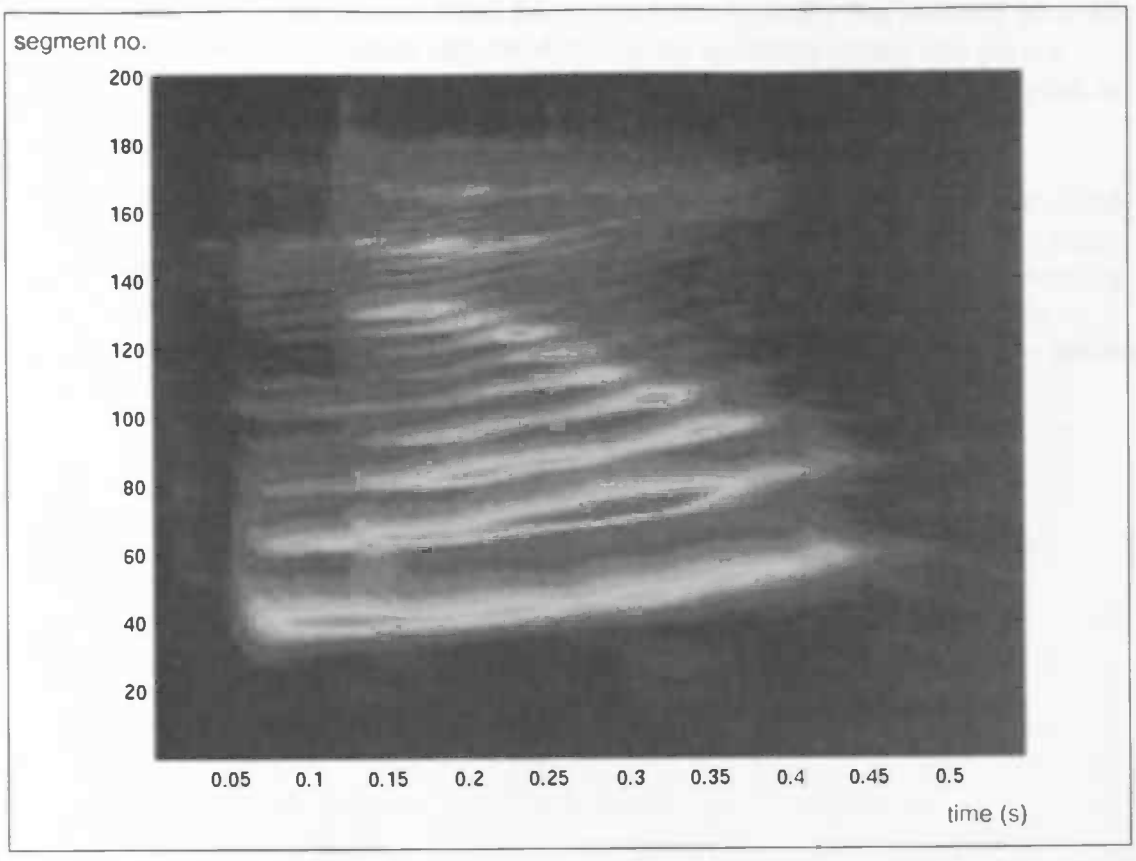

Figure 6: Voiced speech components

'Cochleogram' of the Dutch word /nul/ (zero). The segment axis is logarithmical and corresponds to 20 Hz (segment 20) – 6 kHz (segment 200). Energy values are depicted as colors, ranging from low (blue) to high (red). We see a fundamental at segment 40, with at least 14 harmonics. There are four formants visible: the second harmonic, a 'gliding' one formed by the energetic tail parts of harmonics 3-9, harmonics 12-14 and the highest harmonic. Harmonics and formants are signal components.

T.C. Andringa, 2003

Non-natural sounds

For this introductory description of non-natural sounds we look at planes, cars and trains, all three important sources of noise (see chapters 1 and 2). The values mentioned in this paragraph are based on an observer to sound source distance of 1000 meters or more, as would be the case in a quiet area. They will have to be adjusted the distances we will work with, where sound dispersion and deformation are much less or even absent.

- Jets cause a peak in the spectrum between 200 and 500 Hz, which is filled with noisy components from the gas stream, is hardly periodic and has an average to high Q value. Besides this the more or less tonal sound of a turbine is often present. Propeller planes on the other hand produce a very periodic sound with harmonics with a high Q value, because of the steady turning of the propeller blades. Helicopter sound consists of short periodic pulses.
- Highways emit a relatively constant, continuous sound that, for roads further away, can be heard best in the morning and evening since the atmosphere is more stable at those times of day. At long distances the engine noise is

observed most clearly. It has a frequency that typically lies between 60 to 80 Hz (for a four cilinder engine). Closeby the tire noise comes into play, a "hissing", high frequency (1000 to 2000 Hz) sound whose frequency peak is mainly aperiodic. Because of its high frequency it does not carry far.

- Trains have a broadband spectrum that ranges from 100 to 2000 Hz. Therefore, the Q value is low and the signal is not periodic, unless the sound from the locomotive engine is dominant. From closeby, the sound of a train first rises in intensity, then remains constant for awhile as the train passes by and then becomes less loud again. When observed from farther off, the constant part becomes increasingly shorter while the rise and fall parts become larger.

5. Research question

RESEARCH QUESTION

The CPSP method has proven to be able to select periodic speech components from a noisy signal very well. In that process knowledge about the properties of speech and speech sources is used intensively (Andringa 2002a).
The goal of this project is to try and apply this method to vehicle sounds. *Vehicle sounds* are sounds caused by human-built transportation devices (e.g. planes, passenger cars, trucks and scooters). Some examples of vehicle sounds have been outlined in the previous chapter. However, in this project we will focus upon nearby vehicle passages only. Because they occur closeby, their sound levels are relatively high, compared to sounds from other sources, which makes them easier to detect. Furthermore, there are less dispersion effects (loss of higher frequencies etc.).
All of this is translated into the following research question:

Is it possible to build a CPSP based system that can detect and classify vehicle passages?

- The first step, CPSP, entails selecting signal components (signal parts correlated in time, frequency and phase) in the signal, thus allowing for the subsequent detection of sound events and their classification.
- By detection we mean that the system decides there is enough evidence to declare an event to be a vehicle passage.
- Classification means ascribing the detected passage to a vehicle class.

The last two definitions are elaborated further in chapter 9.

The implemented system will be tested in real conditions, near a road, where its recognition and classification performance can be compared with human annotations. Another way of testing the system would be comparing its output with that of an artificial neural network. Most notably with airplanes, neural networks are already used for vehicle detection.

APPROACH

Many problems in cognitive science are classification problems and can be approached from two directions: *bottom-up* and *top-down*. The bottom-up approach uses very little a priori knowledge and structures. All patterns and regularities needed to solve the problem arise, through a learning process, from the unordered data itself. The top-down approach, on the contrary, makes extensive use of pre-existing knowledge (or assumptions) in ordering the data.
A possible course of action for this project is the bottom-up route. A lot of raw data (sound samples) will have to be gathered and analyzed without making any prior assumptions. This can be achieved by using a self-organizing neural network architecture, such as a *Kohonen map*.
A Kohonen network (Haykin 1999) uses an unsupervised form of learning. Every node in the network is connected to all input neurons. Initially, all the weigths are given a random value. Then, training samples are offered and the learning process

begins. All nodes are in competition but only the one with the highest activation 'wins'. The weights between the winning node and the active input neurons are strengthened. The same thing happens for all nodes in a neighborhood around the winning node. The size of this neighborhood decreases gradually with time, but never goes to zero. Therefore, during learning, there is a spread of activation around each node, which gets smaller as the network is trained more and the distinctions between classes are learned better. Eventually, after training is complete, the Kohonen map represents an ordering of the data. Each node responds maximally to a different input class, and neighboring nodes repond to similar classes (*topological ordering*).
The inputs for a Kohonen network in the context of this project could be *cochleograms* (see chapter 7) of sounds events, represented in vector form.

In this project a mixed approach is taken. The completely top-down route is rejected because not enough information is available about vehicle sounds analyzed with CPSP, mainly due to the lack of prior research. It is impossible to predict exactly what kinds of structures we will encounter. However, it is decided not to work completely bottom-up either, and to use the available knowledge for bootstrapping the system. This choice has been made for the following four reasons:

- First of all, there is a lot of *general* knowledge about vehicle sounds available (see chapters 2 and 4 for an overview) and it would be unwise to ignore this. The application of this knowledge is justified because the project domain is limited. We have a fairly good conception of the (vehicle) classes that will occur in our data.
- The goal of this project is to implement a working system. Time is limited, so we have to make choices. One of these has been not to spend time on deriving our classes bottom-up, but to make use of already existing classes.
- The third reason stems from cognitive science, which aims to apply knowledge about human problem solving (which includes sound recognition and classification). Humans are versatile and reliable sound recognizers, so why not apply some of the methods they use to the problem at hand? This involves insights from human sound processing, hence the use of CPSP, which is a form of signal processing that shows a close correspondence to the human auditory system, and from human sound classification, hence the use of knowledge about sound categories.
- The last reason is technical. The techniques comprising CPSP are able to explain most of the information in a signal using only a few structures. For instance, a complex of harmonics can very efficiently be described just by the fundamental frequency and the number of harmonics. This amount of data reduction will be very hard to improve upon, especially by methods that use no prior knowledge.

It is because of the cognitive background of this project and the availability of reliable techniques and domain knowledge, that we take a mixed approach. However, it remains very important to use as much empirical verification as possible.
Certainly, as remarked earlier, the use of a Kohonen map could be useful to validate our mixed approach. If the classes we use prove to be unsatisfactory, we could also use it to derive better distinctions. Furthermore, it could be used in the stage after the CPSP preprocessing, in order to classify sounds based upon superstructures of

features derived by CPSP.

SYSTEM REQUIREMENTS

Now that the research question has been stated and the way to approach it has been identified, it will be useful to address the system requirements. What are the demands imposed on this particular vehicle recognition and classification system and in what ways could/should it function?
Ideally, the system should be able to recognize -besides vehicles- all possible sounds made by humans, machines, animals, wind etc. If the system is familiar with each sound event that could possibly occur, it will never fail: it never misses disturbances and it never evokes a false alarm. Unfortunately, such an ideal system is impossible to build, simply because there are too many sounds to account for. Therefore, some restrictions must be made: the system needs not recognize all sounds, but it must be able to detect the sounds of interest (in our case: vehicles) as they occur at the site where it is positioned. Because the other sounds that may be noticeable at the location are unpredictable up to a certain level, the system must maintain a *background model* that is adaptive. The statistical properties of noisy background sounds are known: they often develop slowly and show little variance (Andringa 2002c). Therefore, they are relatively easy to model. Non-stationary sound events are harder to model. They can be excluded by making our classifiers very specific.
In order for the system to function optimally, it needs at least a number of sound event classifiers. How the cooperation between these could be managed, is now given some thought.
Suppose every second the latest ten seconds of the incoming sound are analyzed and reduced to a list of feature values. This list forms the evidence against which the classifiers are matched. They are made up of rules that state which feature values are valid for the sound they represent. If their criteria are met well, a high probability is given for the corresponding sound event. If there is little evidence, a low probability is given. The classifier with the highest probability value 'wins' and is chosen to explain the features it matched. These features are then removed from the list and the 'bidding' starts again. This procedure repeats itself until all the evidence has been explained or the highest probability value drops below a certain value or until another stop criterium has been met. Note that a winning classifier is not excluded since its sound could be present more than once.

6. Data

A good starting point of the implementation phase of a project such as this is finding the right data. In this case, that means making recordings of vehicles in all sorts of environments and weather conditions. The data are used to build models (the training phase) and to test the system in which these models have been implemented (the test phase). It is good practice to separate training and test data, so the system cannot just learn the data by heart.
An overview of all the data used in this project can be found in Appendix I. All the sound data have been recorded using a DAT (digital audio tape) recorder and a high-quality mono condensor microphone with a wind shield. The sample frequency is 44 kHz.
The recordings have been fed into a computer and saved in wave form, a way of storing sound data that does not use compression (hence there is no quality loss). Each recording has been annotated in a specific format, decribing per minute, and if necessary per second (as is the case with vehicle passages), all relevant sounds. This is an example from the VCS recording:

min.	**sec.**	**event**
27	35	scooter heading for VCS parking
28	11	motorcycle on dike
28	16	jeep
28	35	car headed for VCS front parking space
29	15	airplane noticeable
29	25	skidding car
29	28	car
29	33	accelerating car
30	6	car
30	30	airplane
31	6	truck braking hard on dike
31	32	car
31	51	car (20 kph)
31	58	accelerating truck on dike

Table 1: VCS annotations (partial)

These annotations are used to compare the output of the system with and compute how well the system performs.

7. General system development

In an early stage of this project, when quiet areas were still the main focus, contact was made with RIVM (see chapter 2). This was done because cooperation seemed to be of mutual interest. RIVM's own methods did not work well in quiet areas, so they were interested in a method that would. This project could, in turn, benefit from RIVM quiet area data and know-how. SI encouraged the collaboration because they were very much interested in RIVM as a possible customer of sound detection products and supplier of data.
RIVM demanded a 'proof of concept': proof that the combination of CPSP and knowledge about sound sources can indeed be used to recognize vehicle events. They proposed the following: a system that could detect and classify vehicle passages in a busy city street. At the same time, the companies VCS and Lochard (see Appendix I) expressed their interest in similar systems, respectively for cars and airplanes. In this phase it was decided to focus completely on nearby vehicle passages (also for the reasons mentioned in chapter 5). A lot of vehicle data was already available, and it was decided to first build a general vehicle detection system and then specialize it for the three different applications mentioned above. This was done not only to persuade VCS and RIVM, but also to gain insight into the workings of sound detection and classification.
We will first look into the general vehicle detection system as a whole (functional design) and then we will examine its components (technical design). In the next chapter, we will review the different applications in the same fashion.

FUNCTIONAL DESIGN OF THE VEHICLE DETECTION SYSTEM

Input

The system is able to deal with sound recordings in wave form (see chapter 6). The length of these files is immaterial, because when they are too long to be analyzed as a whole, they are cut into pieces that can be processed.
The system should be able to function in any outside environment, as long as the vehicles pass closeby (20 meters or less), their sound levels exceed the background noise and non-vehicle sounds are either predictable or distinguishable from vehicle sounds.

Processing

The digitized sound is analyzed by an artificial cochlea (see chapter 3). This model gives as output a measure of the energy of every cochlea segment per time frame. This structure, a cochleogram, is used as the basis for detection.
The detection module will be adaptive. Background sounds, which are often slowly-changing with little variance (Andringa 2002c), are discarded. To be able to make the distinction between foreground and background, a background model will be maintained. If the background sounds change in a qualitative or quantitative way, the model is adapted to accommodate for this, for instance by incorporating persistent sounds.

In the foreground the system looks for evidence for vehicles. For the general system, we define a closed subset consisting of three classes: cars (including vans), scooters and trucks. Later we will also include buses.
The distinction between cars and trucks is based on visual observation and corresponds the official light (cars) and medium/heavy (trucks) categories (see chapter 2). There is no scooter category in this system, whereas we do not recognize motor cycles (because none were available in our data sets). The differences and similarities between the two classifications are summarized in table 2.

Visual categories	**Our system**	**Environmental law**
car, van	car	light vehicle
scooter	scooter	-
small truck, large truck	truck	medium, heavy vehicle
bus	bus	medium vehicle
motor cycle	-	motor cycle

Table 2: comparison between our and the official vehicle categories

Output

The output of the system will be a classification result (if one can be made) together with the time of detection, measured from the beginning of the input (file).

TECHNICAL DESIGN OF THE VEHICLE DETECTION SYSTEM

The system described here has been implemented in Matlab. The cochlea model that is used, has been developed by SI programmers in the C language.

In the course of this paragraph, we will use figure 7 as our guide. It is a schematic overview of all modules of the general system. In each subparagraph, the relevant part(s) of the figure will be mentioned in bold face. This way, we do not lose track of the larger structure while exploring the details of design.
The interpretation of figure 7 should start at the bottom. The recording location and all its sounds are part of the real, physical world. When making a recording, we enter the 'information world'. There, the cochleogram is the central representation. It is manipulated and analyzed on an increasing level of abstraction (going upwards in the figure). Finally, this results in output, as depicted in the top left box.
In the figure, squares represent signal processing steps, circles indicate signal parts and circles with a thick edge stand for knowledge application steps.

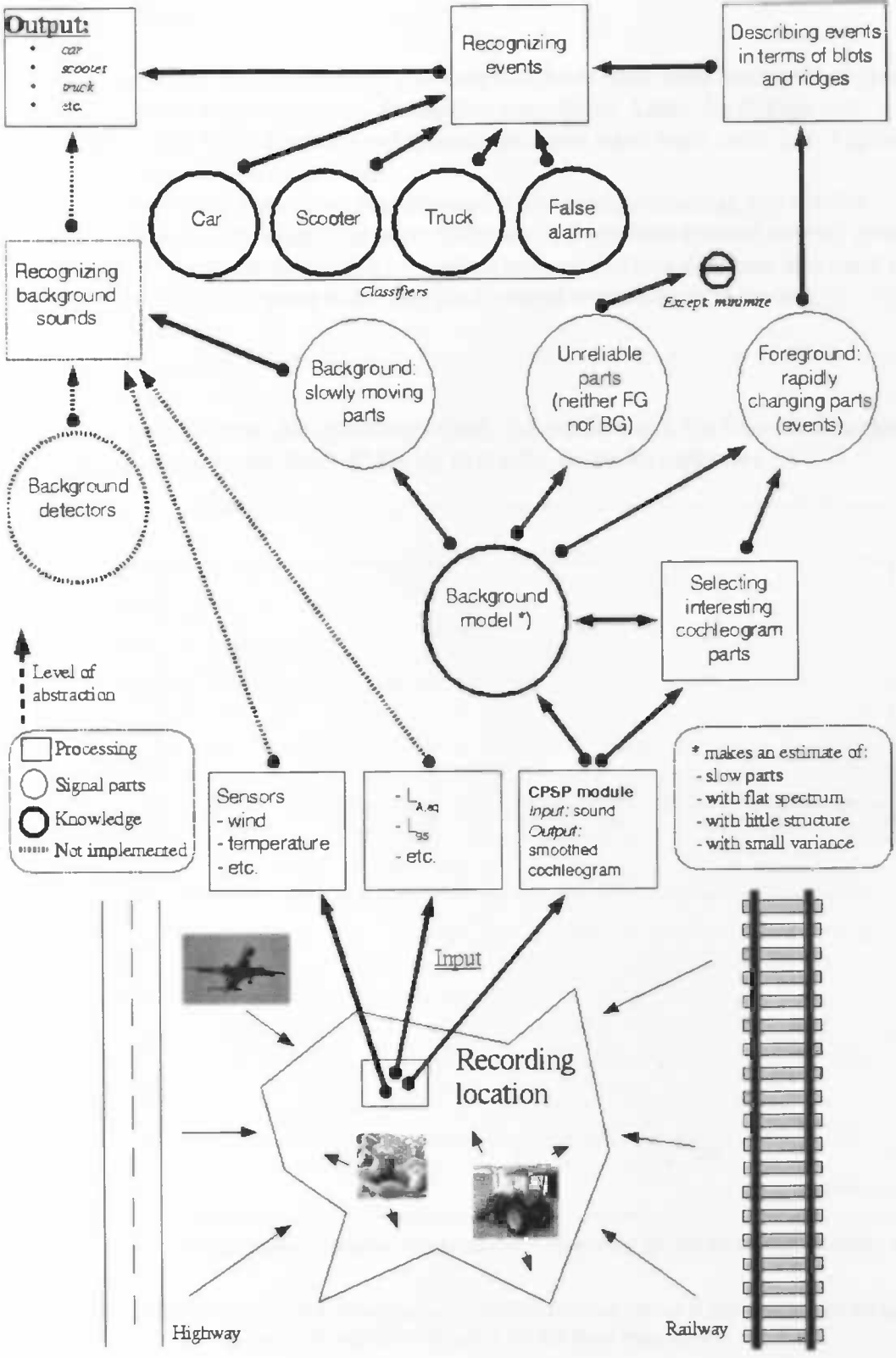

Figure 7: System overview

An outline of all modules of the general vehicle detection system. The different parts of the figure are explained and described in the text.

The database

First there is the **recording** step. The data that have been used during development come from the Maarhuizen and Paddepoel recordings. Later, for testing and finetuning, the VCS, Lochard and Utrecht data sets have been used. See Appendix I for more information on the data.
Those parts of the recordings that showed individually occurring and readily recognizable vehicle sound passages (with an average time span of several seconds) from one of the three categories of interest were added to a database and used in the training phase. Also, parts with only background sounds were included.

Cochlea model

In order not to distort the incoming signal, the model has a flat transfer function. It represents frequencies from 20 Hz up to 6 kHz, using 80 segments.

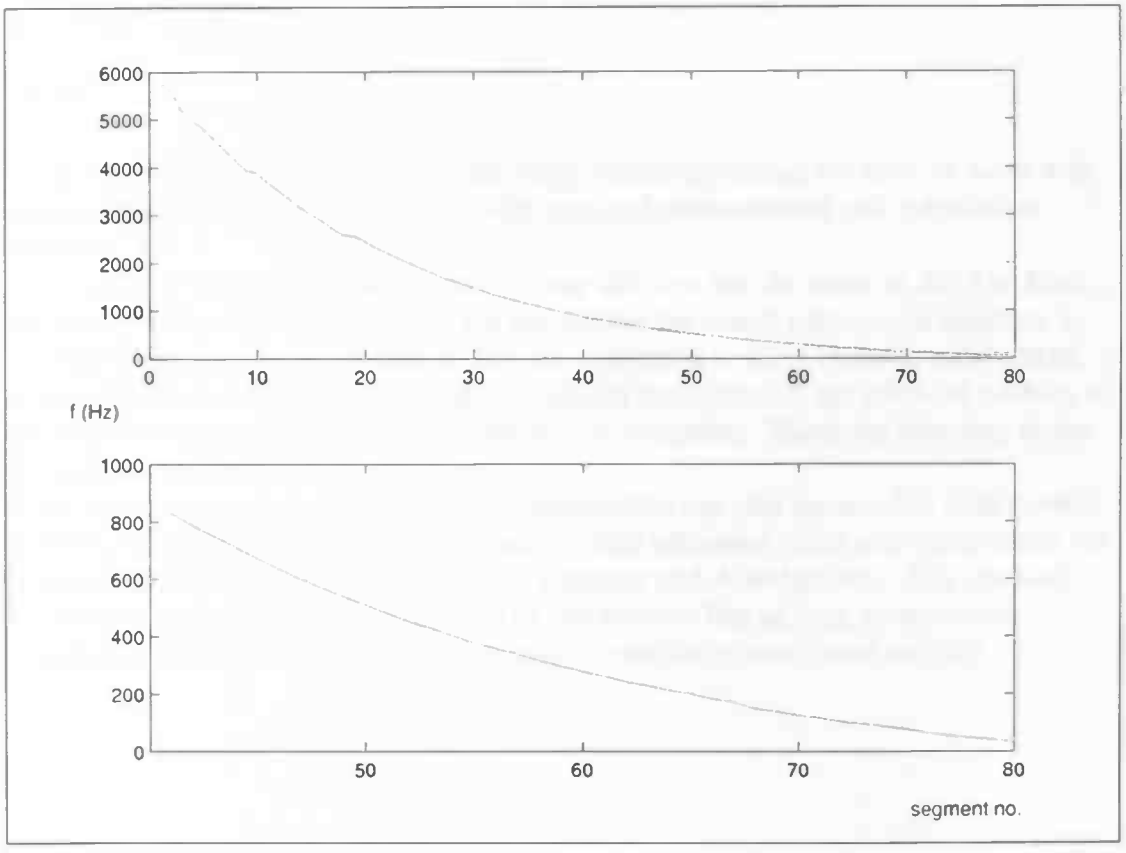

Figure 8: The correspondence between segments and frequencies for the 80 segment cochlea used in this project

In the top panel the relationship (also known as the Greenwood curve) is depicted for all 80 segments. In the lower panel, the part with segments 40 up to 80 has been magnified.

The signal is sampled with a sample frequency of approximately 200 Hz. One sample period (5 ms) is, as we have seen, called a *frame*. The model computes energy values from the (simulated) basilar membrane excitations. These are squared and *leakily integrated*. Leaky integration is a weighted form of integration in which information

about earlier values is gradually lost:

$$E_s = E_s(t-\Delta t) \times e^{\frac{-\Delta t}{\tau}} + x_s^2(t)$$

Equation 7

The subscript 's' indicates that the equation is applied per segment. Δt is the sample period, τ the time constant and x_s the basilar membrane response at segment s.
A Matlab script is used to read in the cochleogram and process it further. When displayed on screen, all energy values are multiplied by a factor ranging, continuously, from 10 for the highest frequencies to 1 for the lowest (thus boosting the highest frequencies), in order to aid visual inspection. After that, all energy values are scaled logarithmically and converted to dBs, using a simplified version of the sound level equation from chapter 2:

$$E_{dB} = 10 \times^{10} \log E$$

Equation 8

Using logarithmical values restricts the range of energy values we have to work with and makes our methods comparable with standard measurement and calculation practice.
However, it is very important to note that our dBs are not the same as dB(A)s! First, the energy values from the cochlea are not, unlike the sound pressure in equation 1, divided by a reference value before they are converted to dBs. Therefore, they have no real physical meaning. Second, the weighting performed in the artificial cochlea is not exactly the same as the –more primitive- A weighting. Third, the boosting factor also influences the sound level values.
If we want to make our system not only comparable but also compatible with current systems, we should incorporate a physically valid reference value and compensate for boosting and the differences between the cochlear and A weightings . It is certainly worthwhile to do so in a future version of our system. But as long as we do not interchange them with dB(A)s, we can safely continue to work with our dBs.

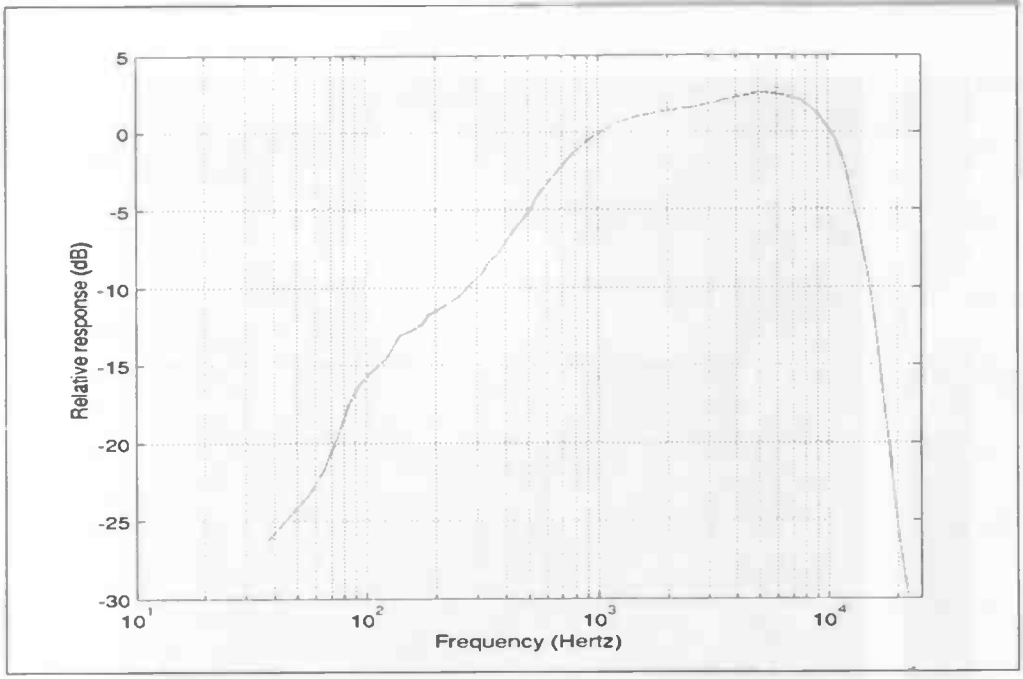

Figure 9: The artificial cochlea weighting curve

The response has been computed by presenting pure sinuses to the cochlea. To aid inspection, the resulting curve has been smoothed with a moving average of 10 segments. The shape of the curve and the responses are roughly the same as the A weighting curve from figure 1, but dissimilarities arise at the higher frequencies (10^4 dB and higher).

The cochleogram is plotted in a two-dimensional plane, with the frequency (the corresponding segment number) on the Y axis, and the time (frame number) on the X axis. The absolute energy value of each point (X,Y) is shown using a color value, ranging from dark blue to dark red, corresponding to respectively the lowest and the highest energy value in the cochleogram.
Applying the cochlea model is part of the **CPSP module** in figure 7.

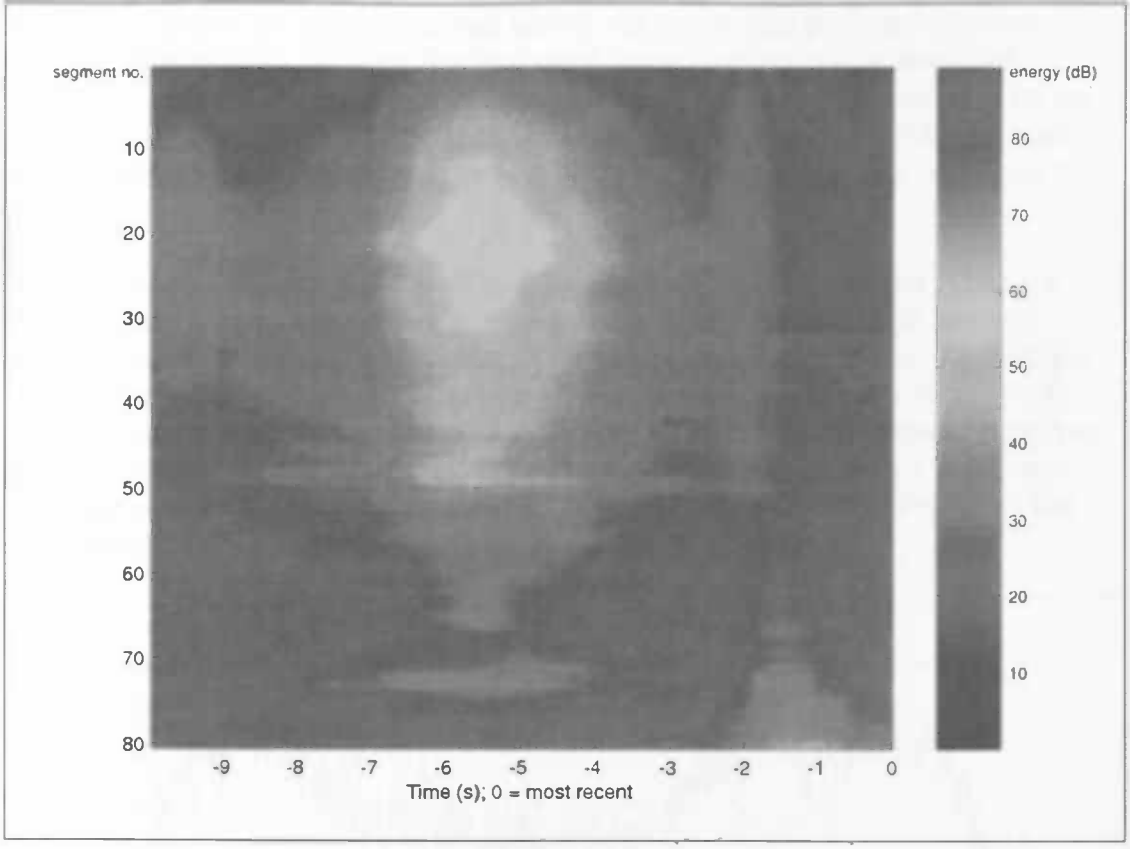

Figure 10: 10 second cochleogram showing a car passage

Smoothing

After studying the database, it was concluded that a form of smoothing would make the signals easier to analyze. There are many small variations in the energy development that are irrelevant for our purposes. For instance, suppose that we are looking for a series of gradually increasing energy values. There may be areas that have an increasing *trend*, but are not continuously rising. Analysis is facilitated if the small variations are removed from the signal and only the trend remains.
A crucial factor when smoothing is the *time constant*. This is a measure of the time scale on which the signal of interest changes. Speech, for instance, changes rapidly, in the order of tens of milliseconds, whereas a train passing in the distance can be accurately described on a scale of seconds. Therefore, speech has a low time constant (the exact value dependent on the speaker, acoustics etc.) and a train in the distance a higher one. In the process of smoothing, which involves taking the mean value of a signal over a period of time (and thereby eliminating information about change in that period), it is essential not to average over periods longer than the time constant. If one does so, important information, represented by changes in the signal, can no longer be detected and is therefore lost.
We will now try to derive, by obeservation, a time constant for close-distance vehicle passages. The average period that a vehicle passing at a distance of several meters is audible, is determined by the wind strength and direction, type and speed of the vehicle and the road surface. However, in our data sets 8 seconds appears to be a good approximation. The vehicle sound is not constant, both qualitatively and

quantitatively, during this period. There are two causes for this: the movement of the vehicle relative to the microphone and actions the driver may perform (accelerating, braking, shifting gears etc.). As for the former, these changes can be described accurately with a time constant of 500 ms (0.5 seconds). The same holds true for the latter: the driver needs time to perform the action and the vehicle does not respond immediately as well. Together this results in an average response time of 500 ms or more.

The signal is divided into parts with lengths equalling the time constant. Then for each segment the mean energy value over this period is computed. This is a very effective form of *resampling* that reduces the signal many times in size. Resampling vehicle sounds reduces 100 frames to only 1 frame, the reduction factor being 100. In principle it is possible to work with these very efficient reduced signals. However, there is a drawback. The designer is forced to work on a time scale that is different from the 'normal' time scale. For instance, problems may arise with parameters that have non-intuitive values and are therefore hard to set correctly.

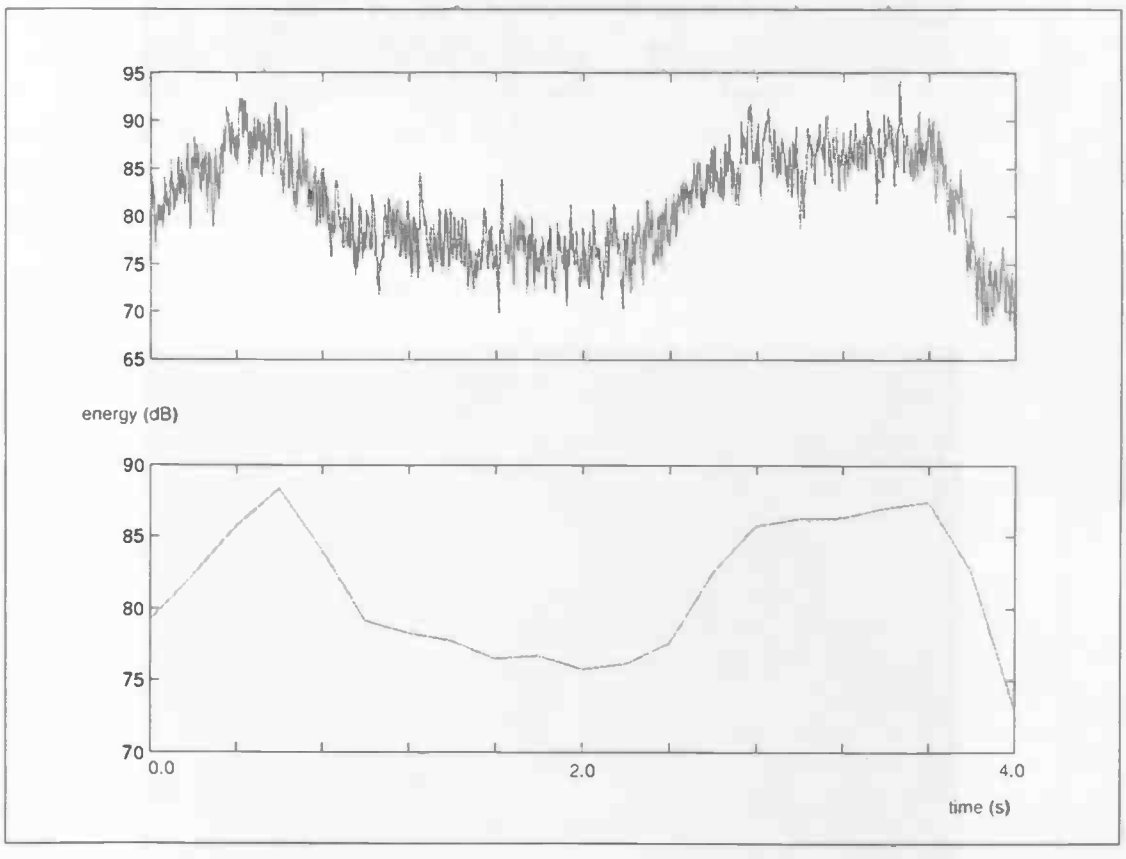

Figure 11: The resampling and smoothing process

In the lower panel the result of smoothing a 4.0 s signal part from segment 40 with time constant 0.5 s is depicted.

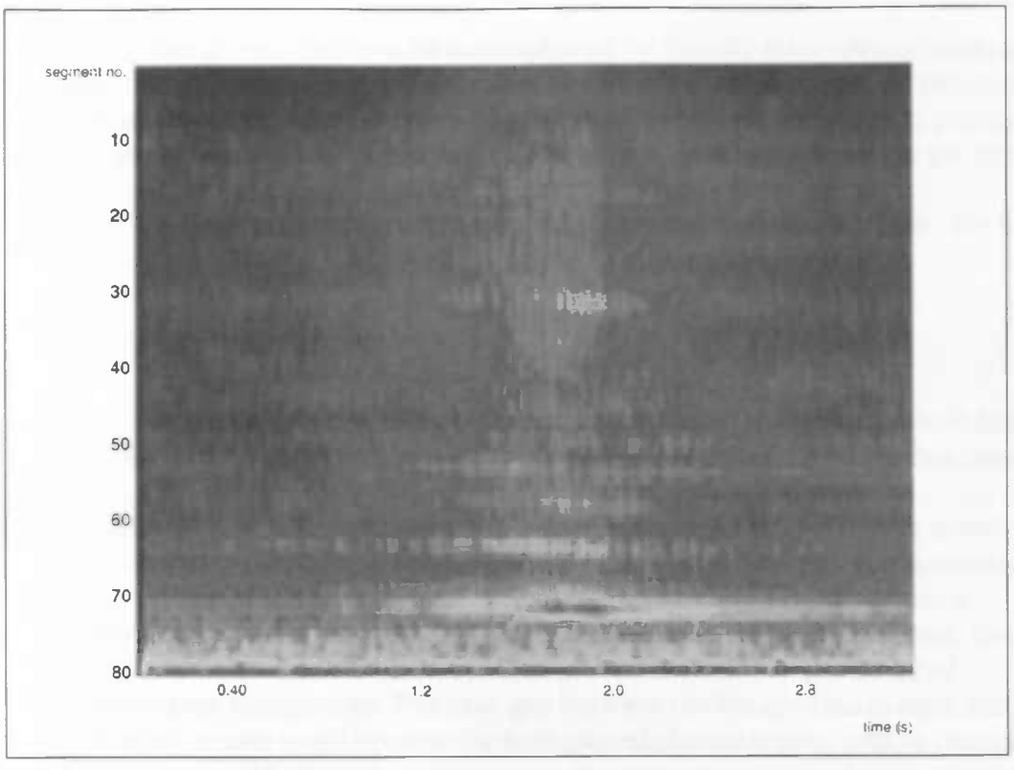

Figure 12: Unsmoothed cochleogram showing a scooter passage

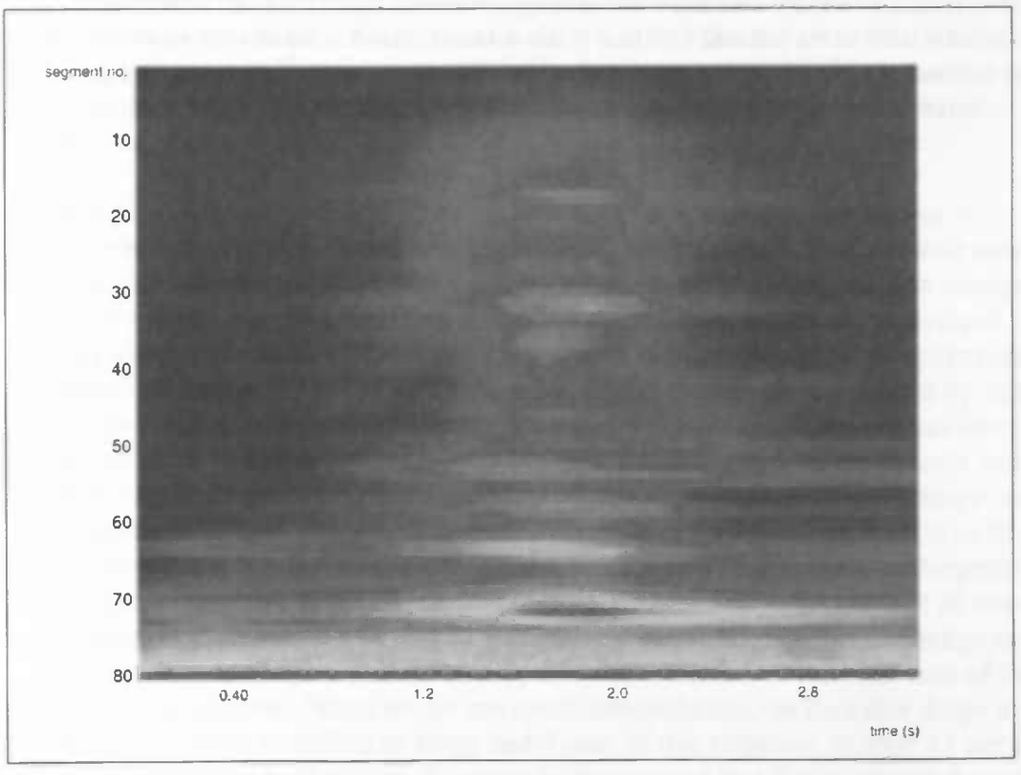

Figure 13: Smoothed version of the same scooter passage

Irrelevant signal parts have been smoothed out, resulting in a much clearer signal. No important information has been lost in the process.

The *smoothing* itself can now be accomplished by linearly interpolating between the means. This way, for each frame the energy value is a weighted sum of two means where the previous mean becomes gradually less important and the next gradually more important. We interpolate in such a way that the distance between the means is again the time constant, so the original sample period is preserved.
Smoothing takes place in the **CPSP module** box of the system overview. We now move on to the **Background model** and **Foreground parts**.

The background model

The next step is to select relevant signal components and to eleminate those belonging to the background. For most of the signals used, background noise is often caused by the wind and the recording equipment.
A possible way to filter out background noise is the following. For each sound fragment analyzed, a human listener tries to find closeby in time a few seconds of relative silence, during which only background noise can be heard, makes a cochleogram out of it, averages that per segment and subtracts it from each frame of the signal to be analyzed. This has indeed proven to be a very good way of eliminating the background. The time gap between the background sample and the signal must remain small because the background characteristics tend to change over time (varying wind strength, moving agricultural machines etc.). All frames of the sample are averaged in order to smooth out small changes.
This approach has two main disadvantages: not always can we find a signal sample that contains almost only background sounds and that has the same characteristics as the signal part we are currently analyzing. Also, human intervention is needed to accomplish the background filtering, making it impractical for our automated detection system.

Statistical properties

However, a way can be found around this problem, by using a **background model**. Some important properties of the background are known: its components change slowly, on a time scale of minutes instead of seconds. Furthermore, analysis of background samples has shown that their energy distributions have an approximately normal (Gaussian) distribution. Background energy values are influenced by many - often intermixed- sound sources, each with its own probability density function. The *Central Limit Theorem* from statistics states that the sum or average of such variables will tend towards a normal distribution, provided there is a 'sufficiently large' number of samples available. Hogg and Tanis (1997) state that a sample size of 25 or 30 gives a good approximation. The smoothed (resampled and interpolated) cochleogram contains, per period, 80 * 100 =8,000 sample records. Suppose only half of this is background, then still our sample is 'sufficiently large'. The Gaussian background assumption is no longer valid when only 25 / 8000 * 100 % =0.31 per cent of the signal is background. When we do not apply interpolation, the data size drops to 80, still large enough according to Hogg and Tanis. In this situation, at least 31 per cent of the signal must be background. It cannot be guaranteed that this is always the case, especially when the signal is dominated by a loud source. In practice, when working with resampled signals we will not use just one column but for instance 8, so the

obligatory background percentage drops to 3.9 per cent.
This information has been used to build a background model. It contains, per segment, the mean (μ_s) of the background. For smoothed signals, the standard deviation (σ_s) has been observed to be almost always close to 1. From the means, the limits of what is background and what is **foreground** (all the fast-changing, energetic events) are calculated. To compensate for the often changing background, the model has adaptive properties.

The histogram
First, for each consecutive part of 500 ms (one *cycle*) in the signal, all energy values of a segment ranging from (μ_s – 3 dB) to (μ_s + 3 dB), are selected. The 3 dB bounds have been empirically ascertained but they are also statistically justified. A well-known rule of thumb from statistics states that 99.7 per cent of the data points from a normal distribution lies within 3 standard deviations of the mean. This entails that our limits contain almost all of what we assume is background.
The selected values are considered to be background by definition and are allowed to update a *background histogram* (figures 14, 15) that contains, per segment and per energy bin (there are bins for every energy value between 0 and the maximum energy level), a measure of the number of energy values belonging there. Because it is a relative histogram, the sum of the bin values per segment must always remain 100 per cent, so once a value has been added to a certain segment-bin combination (which is done by increasing the segment-bin value with 0.01), all the bin values in the segment, including the one just updated, are divided by 1 + 0.01.

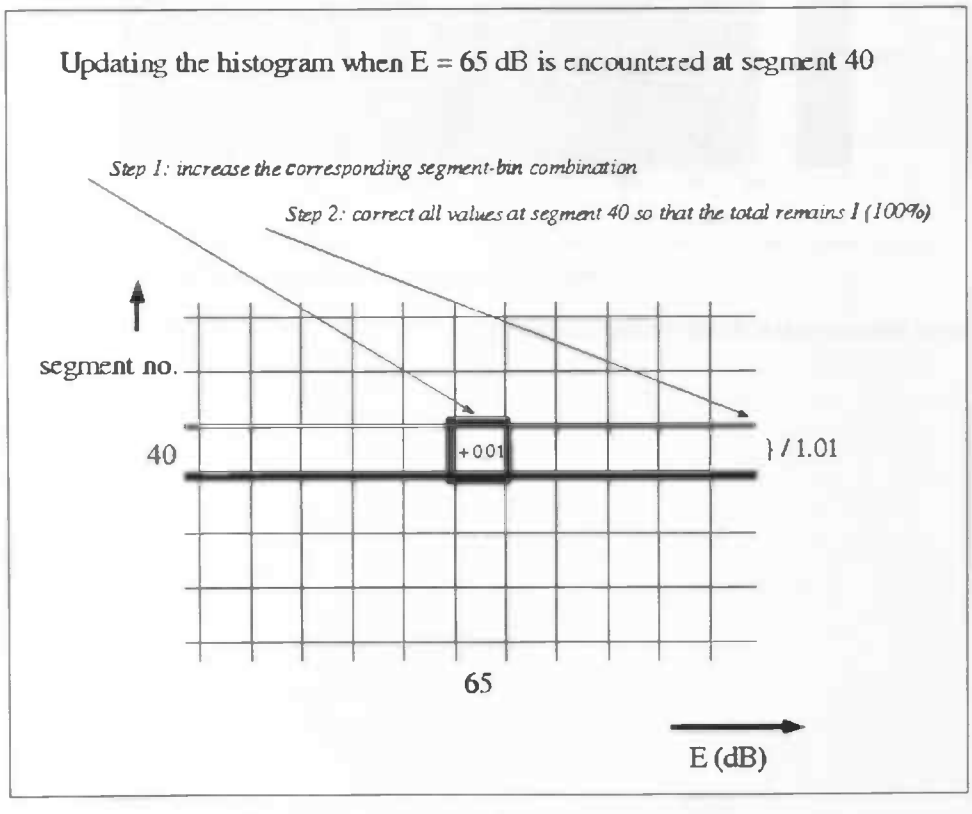

Figure 14: Updating the histogram explained

Every energy value above the higher background limit but below foreground (μ + 5 dB), is considered uncertain. It is not sure whether it belongs to the background or the foreground. Nothing is done with it, except that the number of uncertains, which functions as a measure of the model quality, is incremented. See figure 16 for a schematic overview of the relation between background and foreground.
At the end of each 500 ms (100 frames), new μ values are calculated from the histogram. This is done by taking for each segment all the bin values exceeding 0.01 (the start value for each segment-bin combination) and computing the means of their corresponding energy values. These μs are then used in the next cycle, and so on.
The output of the background model are the signal parts whose energy levels are above the foreground limit. All the other signal parts are *masked* (set to zero).

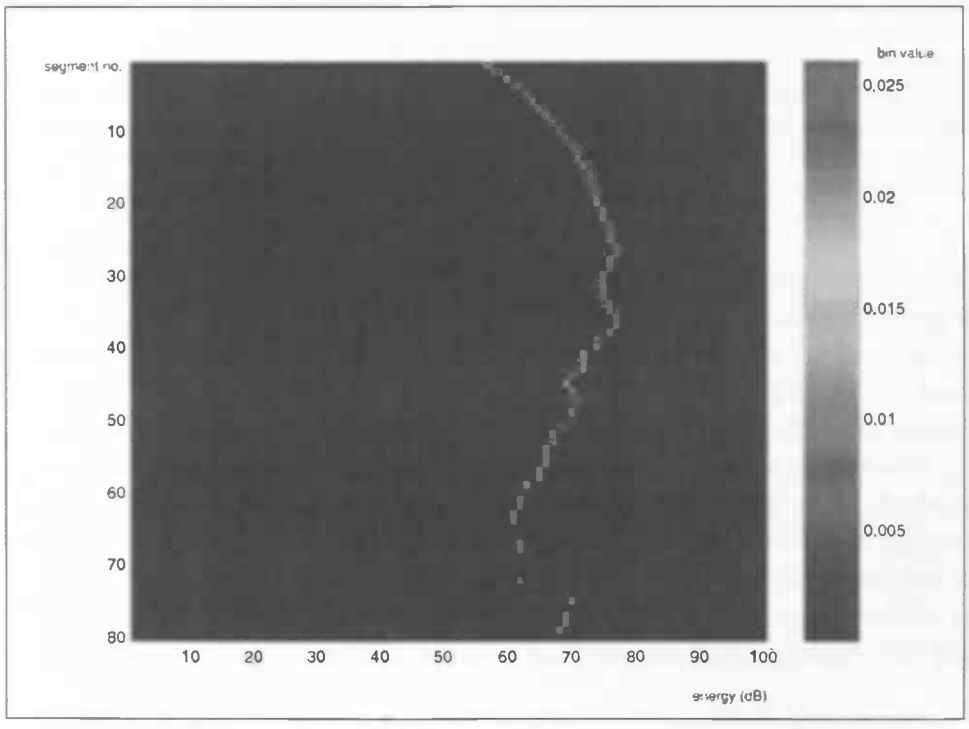

Figure 15: Typical example of a background histogram

It can be seen that the background mean is segment dependent and that the variance is very low.

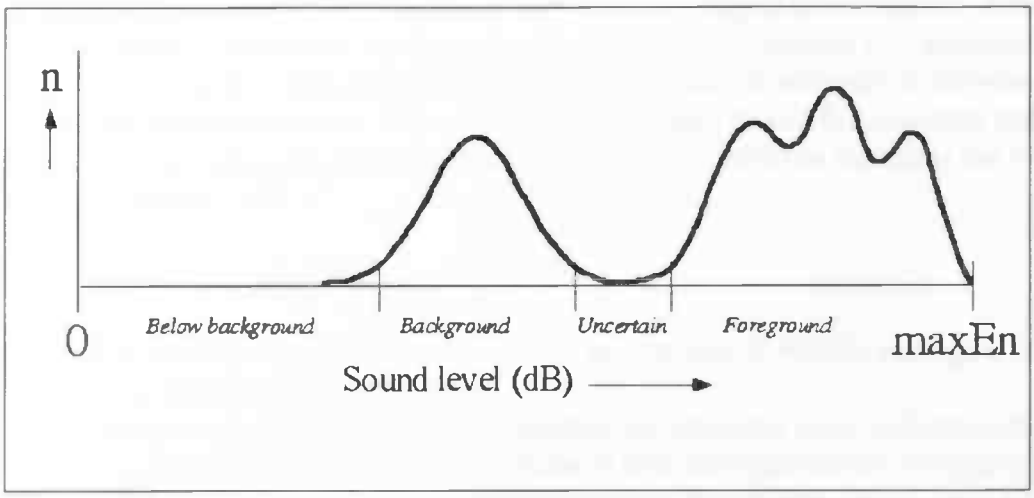

Figure 16: Placement of background and foreground parts along the energy axis, for a given segment. The Y axis represents the number of times an energy value occurs. The energy distribution is purely hypthetical.

Backup model

To avoid changing the model by erroneous inclusion of foreground parts, a second model is introduced as a 'backup option'. This model is always one step behind the first model. Each cycle, histograms and uncertainty values are computed for both models. The model with the lowest uncertainty measure is chosen to make the background-foreground distinction and is subsequently updated as described in the previous paragraph. This updated model then serves as the first model in the next cycle. The not-updated version of the chosen model functions as the second model. This way, should sudden changes occur in a model (e.g. a shift to higher mean energy values caused by a lot of energy values that are actually foreground but lie -for a short period- close to the higher background limit), they can be corrected for in the next cycle by choosing the second model which has not been updated after the shift.

Initialization

The models are initialized on respectively the second and the first second of the signal to be analyzed. The initialization time is kept short so as not to overtrain the models. Since there are no μ values known yet, a different background criterium is applied: every value below 70 per cent of the maximum is considered background. The increase for segment-bin combinations that are present is higher in this stage (0.1), to speed up the learning process. It is important that the initialization part of the signal contains a minimum number of foreground parts, since these can distort the background model. For the final system this will not be hard to accomplish because initialization needs to be done only once, so it can be timed well.

Maintaining continuity

An extra security measure has been built in for the rare situations where the background level either rises or drops abruptly, for instance when a strong wind starts to blow or dies out suddenly. For each segment, at each cycle, the number of values that fall below background is counted. When it is found that this number is over 50 per cent of the total number of values, both background limits for that segment are lowered with 2 dB.

In the (artificial) cochlea the continuity of the original signal is maintained. Although strictly speaking continuity has been lost in the smoothing process (we averaged the signal but we used a well-chosen time constant), we still do not want to introduce any unnecessary discontinuities. To prevent the background model from causing sudden shifts between segments, the maximum difference in μ between segments has been set to plus or minus 1 dB.

Forming ridges and blots

In the foreground we look for structures that may be part of vehicle passage sounds (see the top right box in figure 7).

- **Ridges** are strings of energy values that are relatively high (compared to neighboring values), connected through time and continuous in segment position (only small deviations are allowed). Ridges are used to track periodic signal components, which -as we have seen in chapter 3- have a high Q value and thus dominate a very specific part of the basilar membrane.
- Resonances, signal components with a low Q value, can be described using **blots**. 'Blot' is the shorthand term we will use, instead of the more complete but too complex *synchronous multi-segment structure*. Whereas ridges entrain only one cochlea segment, blots span whole groups of neighboring segments (from 10 up to 60). The energy development is similar for all the blot's segments, since they are all dominated by the same source.

In order to find ridges, first all the *peak values* must be selected. A segment can show a peak at a certain point in time when it has an energy value that is higher than both its neighbors' energies or higher than one and equal to the other. These peaks are connected through time, allowing maximally two shifts in peak segment position per time step. If there is more than one candidate peak to continue the ridge, the one with the energy value closest to that of the most recently added peak is chosen, for reasons of continuity. Not all strings of peaks are allowed to become ridges: a minimum ridge 'length' (equalling the time constant) is imposed. This is a good way of eliminating ridge-like structures that exist shorter than the kinds of signal components we are interested in.

When analyzing the database, we see that most blots are formed during the passing of vehicles. They are caused by the contact of the tires with the road surface. The signal to noise ratio is maximal at the moment of passing, when the vehicle is closest to the microphone. This ratio is therefore a good starting point for finding blots.

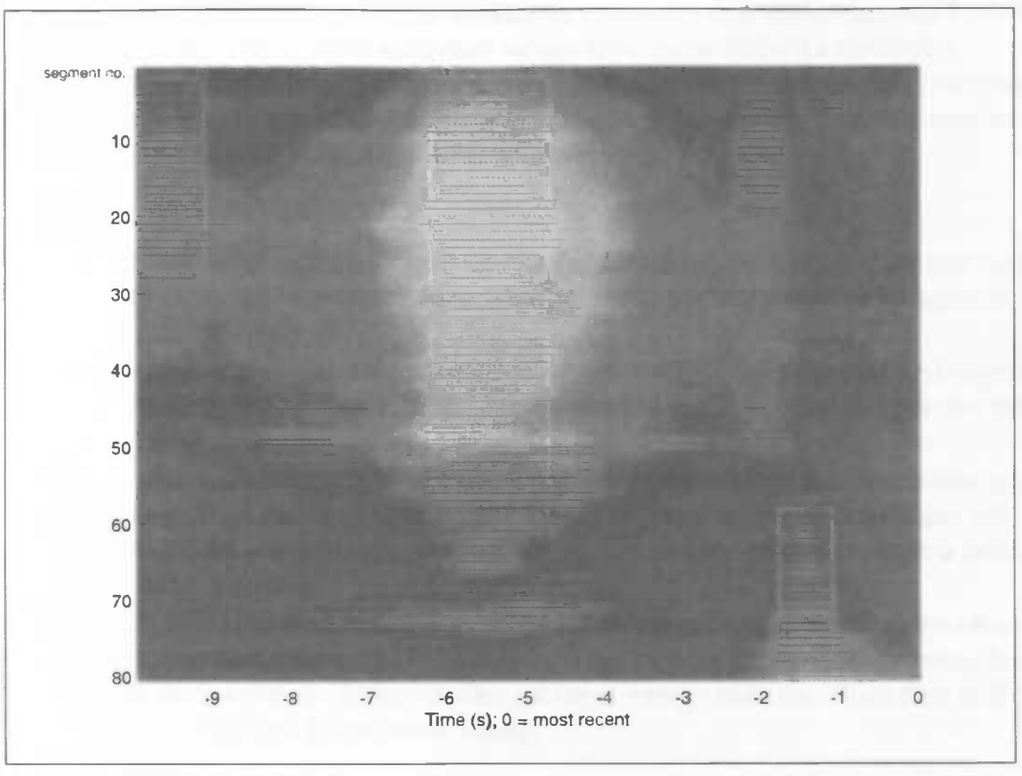

Figure 17: The formation of synchronous multi-segment structures (blots)

The black lines are linked areas, the magenta rectangles blots. A minimum blot height of 20 segments has been imposed.

First, the foreground is divided into two parts: a part with a positive derivative (in which the energy values rise along the time axis) and a part with a negative derivative. Foreground parts with derivative zero are ignored. In both parts we search for *linked areas*.

Linked areas are horizontal strings of energy values that span exactly one segment. Small gaps may occur in these strings, as long as they are not broader than a certain maximum value. For the initial system, 50 milliseconds (10 frames) is a reasonable value: no strings are connected that do not belong together. A linked area's top value (the last value for a rising part and the first for a declining part) and lower limit (the frame closest to the top which has a 6 dB lower energy level than the top value) are stored. Linked areas that do not have a lower limit (because they contain no values lower than the top energy minus 6 dB) are considered unreliable and are dismissed. Then, rising and declining parts that *match*, i.e. when the second has its beginning one frame next to the end of the first, are combined. The result is a string of energy values at a certain segment that rise, reach a maximum value 6 dB higher (which corresponds to a quadrupling of the energy in the linear domain) and fall 6 dB again. This maximum corresponds to the moment when a vehicle passes the microphone.

Then the peaked structures from the previous stage are combined along the segment axis. Not only is there is a maximum shift of the center frame between segments (again the time constant), but also a restraint on the mean segment energy level, which is not allowed to drop 6 dB below the energy of the first (blot formation starts in the highest frequency range and then works downwards) blot segment. These demands

are imposed because blots are per definition structures that are continuous both in time and in frequency. All mentioned values have been derived empirically.
When all the blots have been formed, they are made visible as rectangles ranging from the blot's average left limit (its beginning in time) and the highest frequency to the blot's average right limit (its end) and the lowest frequency.

Summarizing:

- *Peak:* segment-frame point that has either a higher energy value than both its neighboring segments or a value that is higher than one of the neighbors'and equal to the other.
- *Ridge:* time string of peak energy values, continuous in energy and segment position. Maximum shift: 2 segments per frame, minimum length: the time constant.
- *Linked area:* strictly horizontal string of energy values that either rise or fall in time. Top: the maximum value, limit: the frame where the top value – 6 dB is reached for the first time. Every rising linked area is coupled with a falling one.
- *Blot (synchronous multi-segment structure):* combination of linked areas in the segment direction (from high to low frequency). Maximum center frame shift: the time constant, maximum mean energy shift: no more than 6 dB below the first frame mean energy.

Detecting vehicles

In the detection step (**Recognizing events** in the system overview), our system uses the two most important sounds originating from a passing vehicle: the engine noise and the tire on surface noise. The first is found in the cochleogram as a ridge in the lowest frequency range. Say the engine revolves with 1800 to 2400 cycles per minute, then it causes sound at a frequency of 60 to 80 Hz (strokes per second). This shows up in the cochleogram as a ridge around the segment corresponding to 70 Hz. Therefore, the system looks for ridges in that area.
The air-tire sound is translated into a blot between 120 and 3800 Hz that spans at least 20 segments. When around a frame with a maximum signal to noise ratio (averaged over all segments) a blot and an engine ridge can be found, the system claims to have detected a vehicle. What kind of vehicle is not yet known at this stage.
This relatively simple algorithm works quite well: all vehicles in the 'Maarhuizen' recording are detected, so no *misses* occur.
Before going on by implementing the classification process we will first make some improvements to the algorithm.

Additional system features

During passage, sound from the vehicle may be blocked by obstacles for a short period of time. Therefore, in the improved system ridges are allowed to have gaps in them, as long as the ridges are present during at least one, preferably two (or three, of course), of the three following moments: start of the blot, point of maximum signal to noise ratio and end of the blot.

To enhance recognition, several extra features have been introduced:

- The maximum energy within a blot is stored. It is computed as the maximum of the maxima of the linked areas that are part of the blot.
- The mean energy, the mean of the linked areas' means, is also stored.
- The blot height, in segments, and the blot width, in frames, are stored.
- Blots are continuous structures, but there are energy variations in the segment direction. Peaks and ridges can therefore also be found *within* blots. "Ridginess" measures the total number of ridges that exist at the center frame, between the top (highest frequency) and the bottom (lowest frequency) of the blot. This number is divided by the blot height and thus gives an estimation of the extent to which the blot is dominated by ridges.
- "Peakedness" is the average distance (in dB) between maxima and minima in a blot at the moment of maximum signal to noise ratio. The extremes are calculated in the segment direction. Peakedness is a measure of the form the blot's energy distribution (with peaks, flat or intermediate). It can be used in separating passages and non-target sounds. Non-target sounds tend to have relatively high peakedness values (above 6 dB). There is not yet a good explanation available of why this is so.
- The energy-width quotient, calculated by dividing the mean blot energy by the blot width (in frames), acts as a measure of the energy distribution over time. For a blot that exists only for a short time, the energy-width quotient will be lower than for a blot that exists longer and has the same mean energy. As with peakedness, this measure can be used to discern between targets and non-targets.
- The average segment shift is the difference between the average position of all ridges (in segments) during the start of the blot and during the end of the blot. One would expect this shift to be always negative, because of the Doppler effect. However, it turns out that the segment shift is either positive or negative. False alarms tend to have very small values on this feature. There is no satisfactory explanation for this phenomenon yet.

These features are all used in the recognition script. When a blot and simultaneous engine ridge have been found (satisfying the conditions mentioned in the previous paragraph), first a check for **false alarms** is performed. If the sound is considered a target, then the classification algorithms are activated.

False alarms

False alarms are mainly caused by traffic that is not on the target road. Because they are further away, their passages contain less energy than passages on the target road and their blots are peaked more (as was mentioned before). False alarms can also be caused by artifacts in the cochleogram, for instance pulse-like structures that have been 'smeared out' in the smoothing process.
There are 7 checks that are performed on the blot. If 3 or more turn out positive, the sound is considered a false alarm.

- Is the blot height lower than 32 segments (50 is average)?
- Is the blot width lower than 220 frames (300 is average)?
- Is the peakedness higher than 3 dB (2 is average)?

- Or is it lower than 0.2 dB?
- Is the maximum energy lower than 90 dB (95 is average)?
- Is the mean energy lower than 85 dB (90 is average)?
- Is the energy-width quotient greater than 0.6 dB/frame (0.4 is average)?
- Is the segment shift between -1 and +1 (-4 / +4 average)?
- Can a ridge be determined only once, out of a possible three times during passage?

These values have been derived empirically and give good results (see chapter 9). It is not yet clear what their physical meanings are, so this will have to be researched further.
The above consideration also holds for the parameter values mentioned in the rest of this work.

Classifying vehicles

In this subsection, we take a closer look at the rest of the **classifiers** in the top of figure 7.

Scooters produce blots, combinations of ridges caused by the engine. Scooter sound mostly consists of these ridges, harmonics of the fundamental engine frequency (which can lie between 40 and 80 Hz). Two features inspired by this fact have been tried to discern scooters from cars (and trucks): peakedness and ridginess. The former has not been able to improve scooter recognition very much, because the peakedness of car blots is only slightly lower or even the same. The latter, however, is a useful feature.
Another important difference is the energy of the blot as a whole. It has been observed that scooters have a average blot energy of more than 95 dB. The maximum value often exceeds 100 dB. Cars on the other hand are much more silent, with mean energy levels of 90 dB and maximum levels almost always below 100 dB. When these criteria are applied, recognition of scooters improves dramatically.
The eventual result is the following algorithm. If three or more of the following criteria are met, the system claims to have detected a scooter:

- The mean energy is greater than 97 dB.
- The maximum energy is equal to or greater than 100 dB.
- The ridginess of the blot exceeds 0.15 (the average for cars is 0.10, for scooters 0.17).
- The blot has an average segment position lower than segment 25 (average for scooters: 20, for cars: 30). Since this is the most distinguishing feature, it is weighted double.

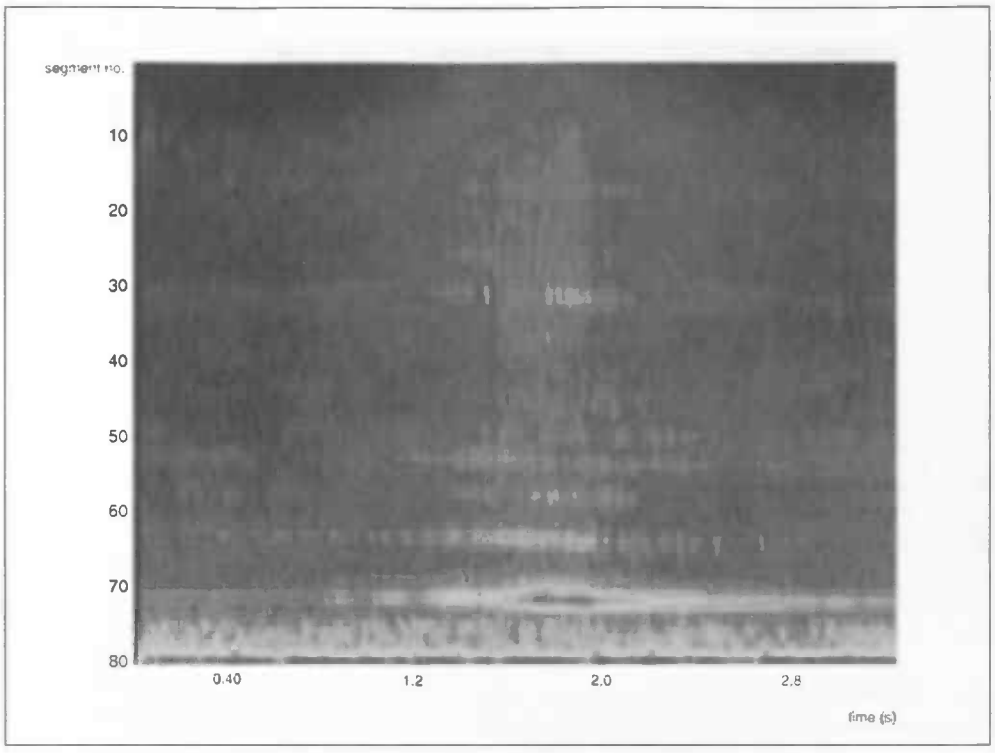

Figure 18: Cochleogram of a scooter passage

A **truck** is classified solely on the basis of the width of its blot. When there is a passage with a duration of 2.5 seconds (corresponding to a blot over 500 frames wide), and an engine ridge can be found during the same period, a truck has been found. Trucks are very long vehicles (certainly compared to cars and scooters), so their passage takes longer and their blots are wider.

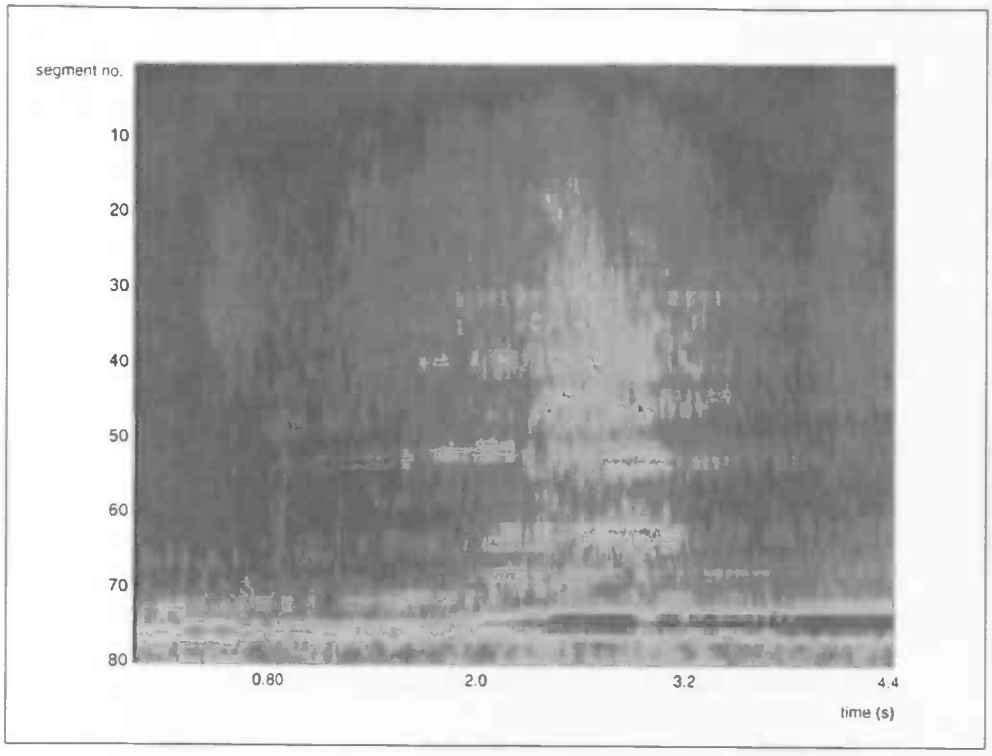

Figure 19: Cochleogram of a truck passage

The black lines are ridges.

There are three kinds of 'meta-classifications' we can make: no passage, false alarm or vehicle. Because our domain is closed, we may assume that every valid detection that is not false is a vehicle and thus either a car, scooter or truck. This implies that if a vehicle cannot be recognized as a scooter or truck, it must be a **car**. Therefore, every vehicle that does not have the special features of a truck or scooter, is a car. 'Car' is the default classification of this system. This implies a risk for faulty classifications: vehicles that are detected but are not of the class car, scooter or truck, will be misclassified as cars. Solutions are to make the detection algorithm more specific or to add extra classifiers.

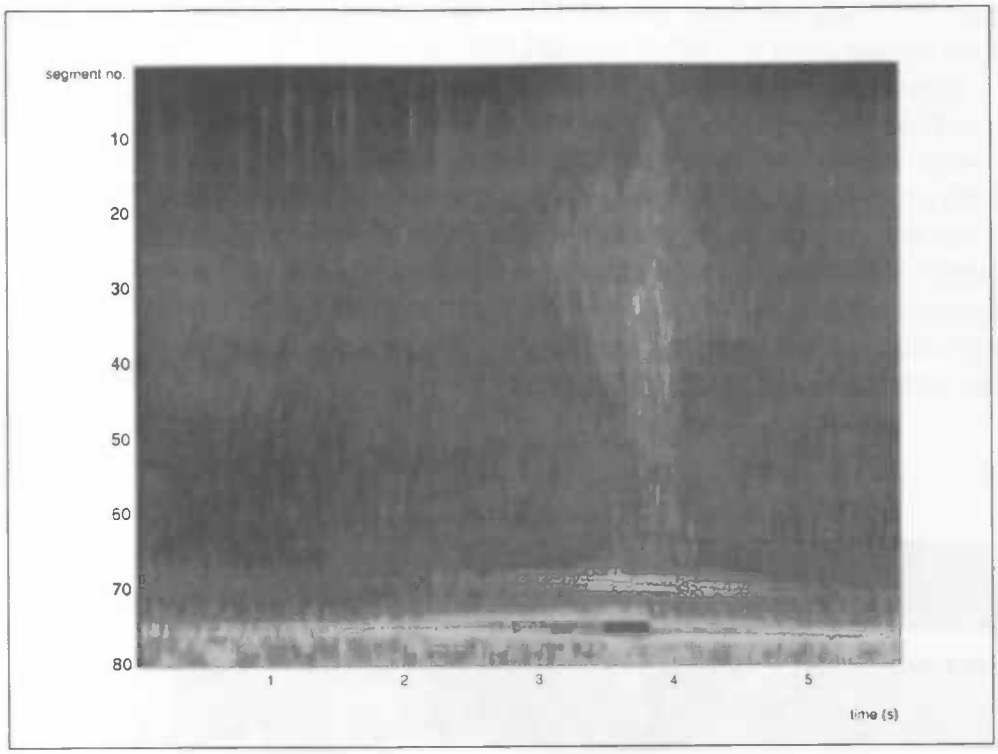

Figure 20: Cochleogram of a car passage

The engine ridge and its first harmonic are shown as ridges.

Output

The result of the detection and classification steps, along with the most important feature values, is written to an information file. An example:

```
---- Analyzing signal from 2.25 to 2.35 (min:sec) ----
Resampled matrix saved as /home/students/erik/matlab/test/graphics/vcs/vcs29-
11_2:25
*TRUCK detected at 2.30 (min:sec)
  Blot has width 495, height 62, mean energy 87, max energy 95, energy-width
fraction 0.18 and mean peak height 1.8
  Blot has center segment 32 and average segment shift 4.1 during passage
  Passage has ridginess 0.15
  Ridge has length 1999, mean energy 87 and mainly dominates segment 75
  Ridges have mean energy 87
```

The continuous analyzer

As during training the signals became increasingly longer the need arose for a tool that could analyze signals in manageable parts instead of as a whole. Eventually, keeping the goal of this project in mind, this tool should work in real-time on live data. In the initial system, the tool works slower than real-time (up to 10 times as slow on a 650 MHz machine, which is mainly due to the computational load of the cochlea model used; the additional scripts are real-time or faster) and only on static data (already existing wave form files).
The total length of the signal at hand is computed. Then the signal is analyzed in

blocks of approximately 5 seconds (1000 frames), until less than 1000 frames are left. The resulting 'residue' is not analyzed because in the live input system we seek to develop there will be no residue as the data flow goes on forever (ideally).
Once a 1000 frame block has been analyzed by the CPSP module and has been smoothed, it is written to the screen in consecutive sub-parts of 100 frames. Each time, the first 100 frames of the plot, which always depicts two blocks (2000 frames) in total, are removed and the latest 100 frames are added at the end of the plot. This is done to create a sense of temporal development. Having done this 10 times, at the end of the block, the resulting cochleogram is analyzed further in terms of ridges and resonances. Because the cochleogram is made up of two blocks, structures that are not yet complete after one block and can therefore not be analyzed properly, can be analyzed 5 seconds later.

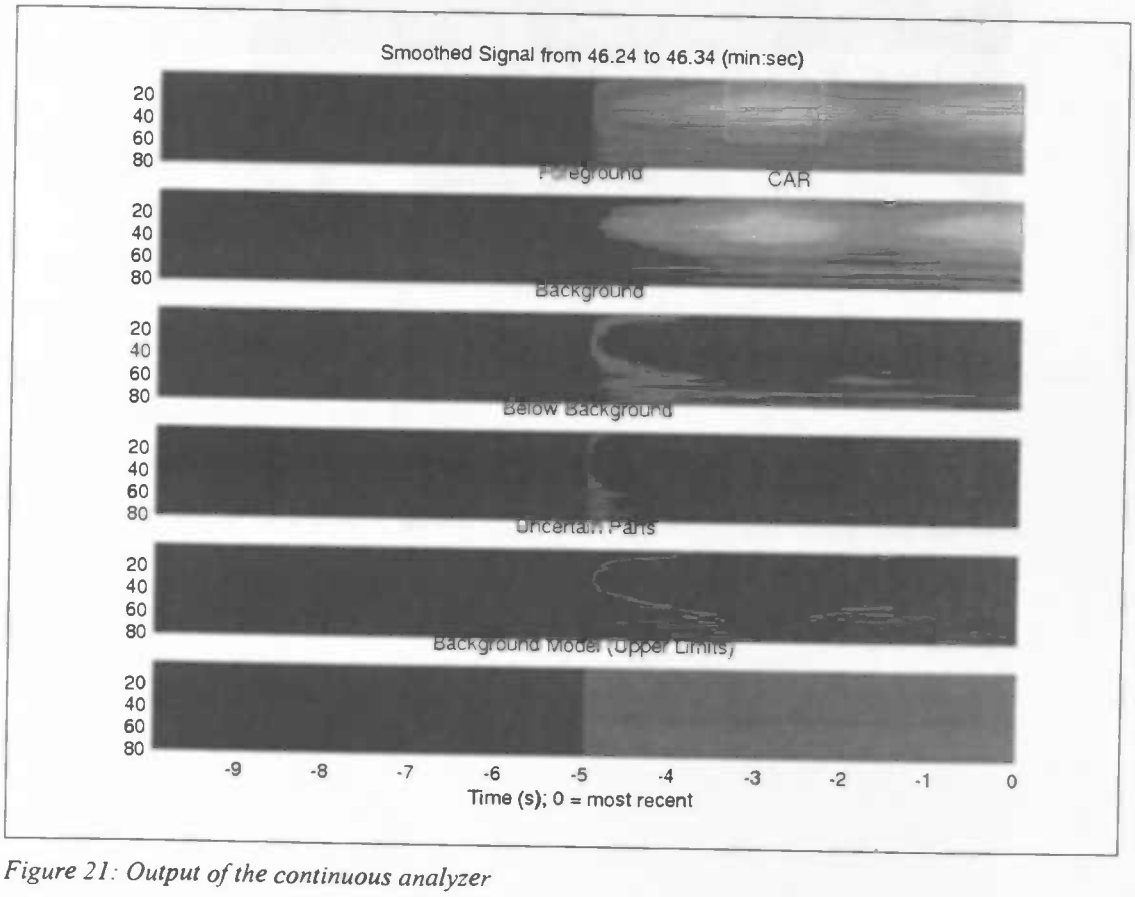

Figure 21: Output of the continuous analyzer

The top panel depicts a smoothed cochleogram of a signal that started 5 seconds ago (X axis: time in seconds, Y axis: segments). The panels below show, respectively, the foreground, the background, the parts that fall below the background, the parts that fall between background and forground and the upper limits of the background model. Panels 2 through 5 again form the whole signal when added up. Note the blots, ridges and classification result in the top panel. The analyzer has been designed to depict two blocks at any given time, but because there are not yet 10 seconds of signal, the missing part has been filled with zeros.

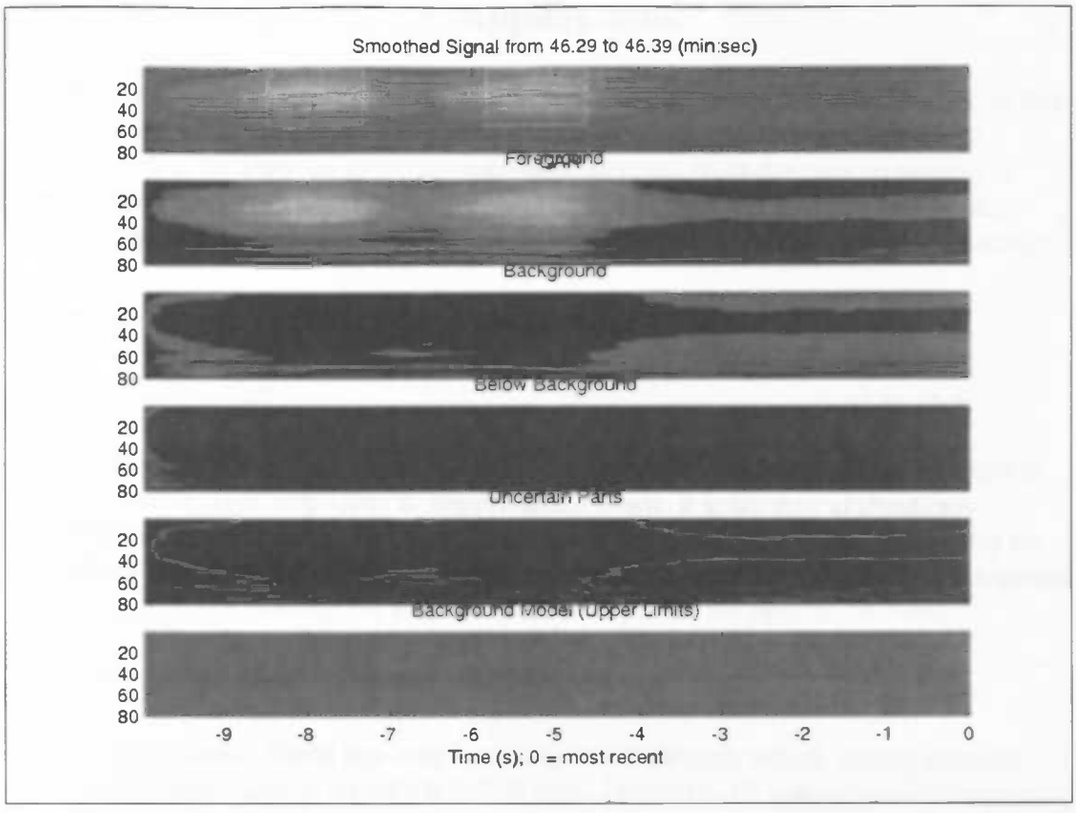

Figure 22: The same signal, 5 seconds later

Now there is enough signal (10 seconds, two blocks) to fill the whole figure.

8. Applications

As was mentioned at the beginning of the previous chapter, soon after the start of this project three different parties expressed their interest in a version of the vehicle detection system. These systems have all been derived from the general system as described earlier, each adjusted to the particular demands of the customer. In this chapter, the adjustments are explored in full detail, both functionally and technically.

APPLICATION I: CITY STREET VEHICLE DETECTOR

Functional design

The first application, a vehicle detector for a busy city street in Utrecht (the 'Utrecht system' for short) is very close to the original system. It is used to analyze pre-recorded sounds and detect and classify vehicles if present. As was mentioned in the introduction of chapter 6, the performance of the system on the Utrecht data set would serve as a proof of concept for RIVM.

Situation

The Constant Erzeij Street is a very busy city street through which, during daytime, outside rush hour, approximately 600-700 cars, 25 trucks, 10 scooters and 10 buses pass per hour. Approximately 200 meters from the RIVM measuring post, there is a T crossing with traffic lights, so the vehicles coming from that side tend to be in groups. There are houses on both sides of the road. Background sounds include distant traffic noise, airplane overflights, street noise etc.
Because of the amount of passing cars, only the special vehicles (scooters, buses and trucks) were annotated. A representative part of the recording was annotated afterwards by a human listener (see chapter 9).

Background model

In the Utrecht system, in fact in all the application systems, the background model has been slightly simplified. The backup model put a heavy computational load on the system but was almost never used. If has therefore been removed, without hardly any impact on the quality of the model.

Detecting scooters

Although RIVM does not provide for scooters in its categorization (see chapter 7, 'Functional design'), they are interested in detecting and classifying them. One reason is the fact that scooters often influence the $L_{A,eq}$ without being accounted for in the official statistics. These are therefore less reliable than they would be when corrected for the amount of passing scooters, as ascertained by our system. A second reason are the possibilities that analyzing scooter sound with CPSP offers: for instance, we could determine the percentage of scooters that have been tuned up.

Detecting buses

In Utrecht, the first recordings were made of buses. Recall that for RIVM, buses and medium heavy trucks form one category. For a human listener however, it is possible to discern buses from other vehicles, so we think our system should be able to as well. These are the criteria to classify a passage as a bus:

- The blot width must be between 240 and 440 frames.
- The height must lie between 45 and 70 segments.
- The mean energy must be at least 75 and at most 90 dB.
- The maximum energy must be between 96 and 104 dB.
- The center segment lies between 24 and 34.
- The blot's peakedness must be at least 1 and at most 3 dB.
- The mean ridge energy should not exceed 80 dB.

For both this application and the next (the airplane detector), a new blot (and ridge) feature was introduced: the energy standard deviation. It is calculated with the unbiased standard deviation equation for samples (N is the sample size, E(n) the energy for sample record n and E-bar the mean blot or ridge energy):

$$s = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N} E(n) - \bar{E}}$$

Equation 9

- For a bus, the standard deviation must be between 5 and 8 dB.

These criteria have been derived by carefully studying bus passage characteristics. However, it is often not easy to distinguish between buses on the one hand and cars and trucks on the other hand. Because of that there are so many -and very strict- criteria.

APPLICATION II: AIRPLANE DETECTOR

Functional design

The airplane detector (developed for Lochard Inc., therefore also called the 'Lochard system') has been implemented to analyze sound samples containing sounds of overflying aircraft. It can detect and distinguish between two types of airplane: jets and propellers. With jets, it can -if necessary- make a further distinction between ascending and descending planes.
The system itself works real-time on 650 MHz machine, which has been achieved by first performing the cochlear analysis separately and by working with a down-sampled version of the cochleogram. The time constant used here is 1 second, which means that for each segment, 200 frames are averaged into 1 (see chapter 7 for more details). Airplane sounds change slower than other vehicles' and 1 second has proven to be an accurate estimation of the time constant. This has not yet been physically founded.
We introduced new feature that makes the system more adaptable. The user can now

set two important parameters:

- One is the maximum energy, 'Emax', in dB, representing the upper limit of our measuring range. Every value above this is removed and replaced by the maximum value. This is done to avoid introducing oversteered signal parts into the analysis. *Default value:* 120 dB.
- Another is the expected energy level, 'enLevel', in dB. This is a measure of what the mean sound level is at a certain location, under typical circumstances. In the Lochard system, this value is used as a reference in the detection process. For instance, the jet ridge energy must be 5 dB above this value. In the current system, it is up to the user to ascertain (by measurement) a sensible value for this parameter and put it in the configuration file. In the future, the system will have to be able to make a good estimation on its own. *Default value:* 80 dB.

Situation

The plane detector should, like the other applications, function in a wide range of environments. The data sets include plane overflights with almost no noise, with some vehicle noise and with highway noise that is as loud as the overflights.

Detecting jet planes

The most important characteristic of jet planes passing at a distance of 50-500 meters is a high-energy (higher than the expected level) ridge around 2400 Hz (segment 20), caused by the turbine. If such a ridge is found, the following criteria are applied:

- Is the ridge length greater than 5 but smaller than 20?
- Is the ridge energy higher than 'enLevel' +5 dB?
- Does the ridge standard deviation exceed 3 dB?

If all three criteria are met, the system assigns the ridge to a jet. When we want the system to distinguish between ascending and descending planes (as with the Schiphol recordings where every plane is either landing or taking off), another two criteria are subsequently applied.
In the case of ascending planes, we look for:

- At least 4 ridges in the area above segment 70 that 'rise' (their end segments are lower than their start segments). See figure 21.

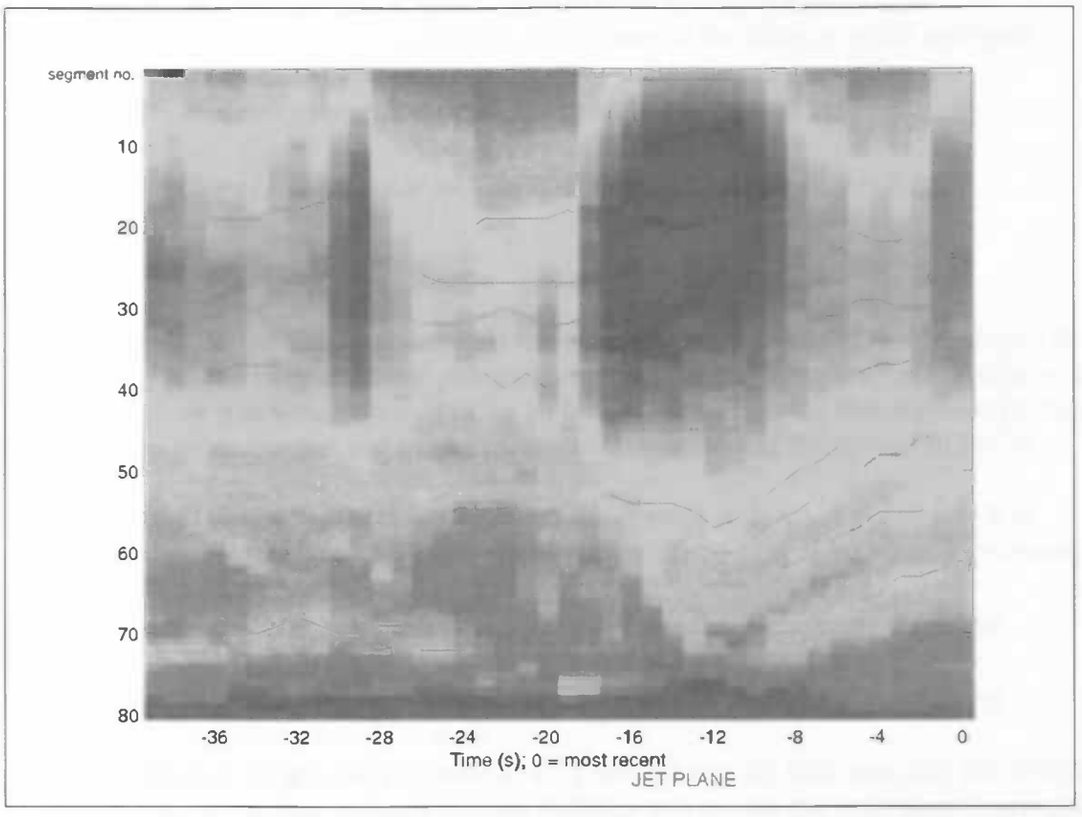

Figure 23: Jet plane taking off from the Kaagbaan

Other high energy parts (the first 20 seconds and the last 2) are caused by the nearby highway. Note the turbine ridge around segment 20 from -24 to -10 s and the 'rising ridges' from -18 s until the end.

If this does not succeed, we look for a descending plane:

- There should be a blot at least 3 frames wide and 18 segments high in the area above segment 40. This blot must occur during the presence of the turbine ridge.

Detecting propeller planes

The system first looks for a blot with the following characteristics:

- The blot bottom lies above segment 70.
- The blot is wider than 5 frames.
- The blot is higher than 20 segments.
- The center segment is 40 or lower.
- The mean blot energy is no more than 10 dB lower than 'enLevel'.

Such a blot is caused by the turbulent sound from the engines. Having found one, the system then looks for a ridge originating from the propellers. The search area is restricted to 125 Hz (segment 70) through 50 Hz (segment 78).
After this, the systems performs a check for false alarms. At least three of the following five criteria need to be met for the detection not to be discarded:

- The mean ridge energy must be higher than 'enLevel'.
- The blot energy must also be higher than 'enLevel'.

- The standard deviation of the ridge(s) must be above 4 dB.
- There should be a ridge present at least two of the three possible moments (start, center and end of blot).
- The blot energy standard deviation must be above 1 dB.

APPLICATION III: VEHICLE DETECTOR FOR SECURITY PURPOSES

System design process

This application (the 'VCS system') is the most elaborate that has been developed in the course of this project. In this paragraph we will therefore not only examine how it works, but we will simultaneously take an in-depth look into the development process itself. As a guideline, we use the *iterative waterfall model* as mentioned in Dix et al. (1998).
The waterfall model is a model of software design that consists of six stages. It is iterative because, at each stage, the process can return to a previous stage if necessary. The six stages are:

- Requirements specification: a description of what the system should be capable of, what input it must handle and what output it must give.
- Architectural design: practical considerations such as the programming language that should be used.
- Detailed design: the program flow of control and the functions that are needed.
- Implementation and unit testing: building and testing the individual functions.
- Integration and testing: building the whole application and testing it.
- Operation and maintenance: keeping the system 'in shape' during its active life.

The first three steps of the waterfall are described in this paragraph. The system we developed is not a perfect example of a waterfall process (such examples are in fact rarely seen): because the VCS system has been derived from the general system, most of the code had already been developed. In this chapter, we therefore only review relevant additions.
The code of the individual functions (stage 4) can be found in Appendix IV. The next chapter contains the test results for the VCS system (stage 5). Unfortunately, as of the time this is written, the system has not yet been installed at VCS. Therefore we are not able to report on stage 6.

Requirements specification

In the VCS system, the vehicle detector is used to direct a camera system. If a vehicle approaches the VCS building, it is noticed by our detector, which then sends a signal to the camera containing information about the kind of vehicle and the direction it is headed. The direction part has not yet been implemented, but it will be in future versions. For the process to work well, the system must function real-time. Long delays render it useless, because then the vehicle is out of camera reach and the license plate cannot be scanned anymore.

In practice, this means the system must be able to signal a detection approximately 0.5 seconds after the vehicle has passed the microphone. During that time, a car traveling at 70 km/h (50 is the speed limit on the road in front of VCS) has covered approximately 20 meters. The camera maximally needs another 0.5 seconds, to be able to rotate 180 degrees, the maximum angle. Therefore, the microphone must be placed 40 meters (or more) before the camera pole.

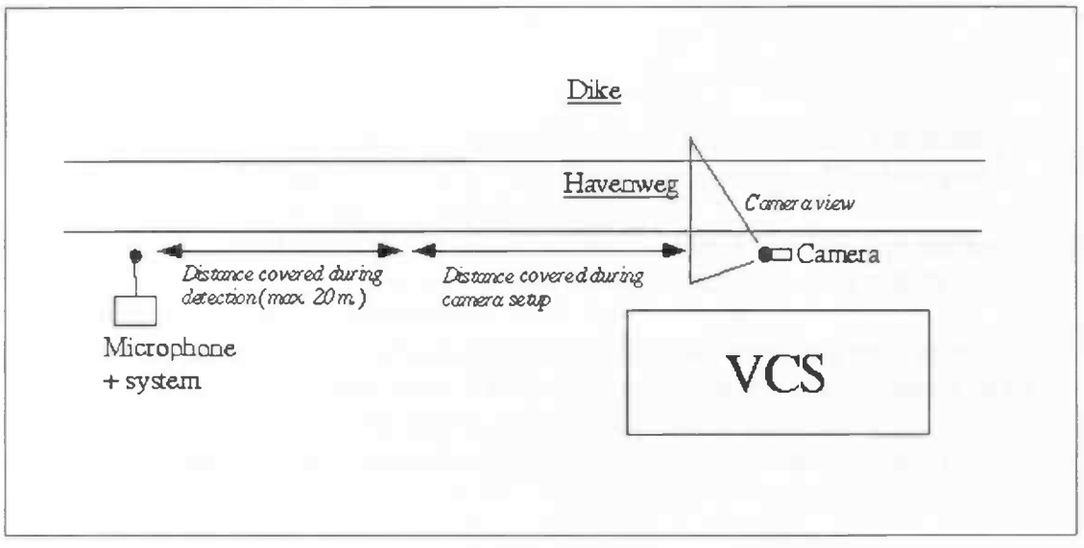

Figure 24: Placement of the microphone at the VCS site

Architectural design

To create a working system from the Matlab code, the code was converted to C code using the built-in Matlab compiler. Then it was integrated into the already existing Sound Intelligence C system. This system consists of a range of *knowledge sources*, modules performing a specific task (for instance, processing cochleograms), a *blackboard* which is used for data exchange between the knowledge sources and a *controller* that decides which knowledge source may be activated, based on the data available on the blackboard. Our compiled vehicle detection code was added to this as another knowledge source.

Detailed design

Cochlea

As was mentioned in chapter 7 and in the Application II section, it is perfectly possible to work with a cochleogram in which every 100 (or 200) frames have been averaged into 1. A variant of this approach has been taken in the VCS system. Here a different cochlea is used, a so-called *low-pass cochlea*.

First of all, take notice of the fact that 'low-pass' is not used in its usual sense (blocking signal frequencies higher than a certain value). Here it refers to blocking signal components that have a local frequency that is higher than a certain value and causes them to change faster than the time constant.

The low-pass cochlea is therefore insensitive to changes that occur on a time scale

shorter than the time constant (0.5 seconds). This is accomplished by dividing the original spectrum by a low-pass filtered one. The upper bound of this *reference spectrum* is the time constant. Only signal components that change on a scale faster than the time constant, are let through. Therefore, it only contains events with time constants between 0 and the system time constant.

$$LP_\tau(t+\Delta t) = LP_\tau(t) \times e^{\frac{-\Delta t}{\tau}} + K \times E(t+\Delta t)$$

Equation 10

It is computed in the same fashion as the cochleogram energy values. The scaling factor K has been introduced to keep $LP_s(t)$ (the output -energy- of the low-pass cochlea at segment s and time t) in the same range as E.
E is divided by LP and the resulting spectrum is scaled back (see the next section). Division had been preferred over subtraction because the latter could lead to negative energy values, a physical impossibility.
With the smoothing already performed by the cochlea, we now only need to extract a column from the cochleogram every 100 frames and we have a representation of all sounds that change more slowly than in 0.5 seconds, like vehicles.
Every 0.5 seconds, our knowledge source is called with the latest 4 seconds (8 columns) of signal.

Parameters
Almost all parameters in our knowledge source have been made adaptable by putting them in a configuration file. When the C system is started, the parameter values are read from file and stored in memory.

- There are two parameters we have already seen in the airplane detector: 'enLevel' and 'Emax'.
- The percentage of the maximum energy level below which energy values are considered background during the initialization of the background model is put into 'percBG', whose default value is 85%.
- The number of dB's between background and foreground is stored as 'FGlimit', with default value 5 dB.
- Because of the division by the reference spectrum the low-pass cochlea has a response that is different from the unfiltered one. This can be compensated for by dividing all input values by a 'scalingF'. Default value: 3000.
- The last parameter is 'dMode' (stands for: diagnostic mode), a boolean that determines whether the cochleograms that the Matlab code receives are written to a file or not. Setting 'dMode' to 1 is useful when debugging the vehicle detector.

The vehicle detection and classification function, which is activated when at least one blot has been found, has a set of parameters of its own. They are described briefly here, along with their default values (the actual values used in the VCS demo system). Their use is made more clear in the next paragraph, where their roles in vehicle classification are described.

- cSegHigh: center segment higher limit (segment 50).
- cSegLow: center segment lower limit (segment 15).
- rLow: engine ridges lower segment limit (segment 73).
- rHigh: engine ridges higher segment limit (segment 79).
- falseLimit: number of pieces of evidence of false alarm needed to discard detection (3).
- falseH: suspicious height below this value (40 segments).
- falseH2: even more suspicious height below this value (20 segments).
- falsePness: suspicious peakedness above this value (3 dB).
- scMaxAboveEn: maximum scooter energy must be so much higher than the expected level (25 dB).
- scRness: scooter ridginess must be at least this fraction (0.25).
- scCseg: scooter center segment must be below this value (segment 30).
- scRenBelowEn: mean scooter ridge energy must be at least so much lower than expected level (2 dB).
- tWidth: truck blot must have at least this width (3 frames).
- tAboveEn: mean truck energy must be so much higher than expected level (10 dB).
- sepSec: minimum nr of seconds assumed between events (3 s).
- tRenAboveEn: truck ridge energy should be at least so much higher than the expected energy level (12 dB).
- minRness: ridginess should be at least this fraction (0.10).
- enRange: allowed energy range (25 dB).

In this version of the system the background model is not adaptive. It does not, contrary to the initial system, remember histogram values between calls. This problem was caused by the Matlab compiler, and attempts to solve it by making the histogram a static variable (that is initialized once and is maintained between function calls) did not succeed. This will be one of the first issues that will be addressed in future versions of the detection system.
Instead of being adaptive, the model is -during each call- initialized on the whole 8 columns times 80 segments data points. It makes, as described in chapter 6, a histogram of all values that fall below 'percBG' times 'Emax', values that are certainly background. The distribution of the energy values corresponding to the bins that are 0.01 or higher is analyzed per segment and their mean values are calculated. These are considered the means of the Gaussian background curves. Then 'FGlimit' is added and the resulting values are the foreground limits per segment. These are applied as a mask to separate the foreground from the rest of the signal.

False alarms
Only blots are taken into account whose center segment is between 'cSegLow' and 'cSegHigh' and whose mean energy level is between 'enLevel' and 'enLevel' + 'enRange'.
The definition of a *false alarm* has been changed. Also, as we have seen, the number of conditions that have to be met for a passage to be classified as false, has been made adaptable (3 is standard).

- Is the blot height lower than 'falseH' segments?
- Is the blot height lower than 'falseH2' segments?
- Is the peakedness higher than 'falsePness' dB?
- Or is it lower than 0 dB?
- Is the maximum blot energy lower than 'enLevel' +3 dB?
- Is the mean blot energy lower than 'enLevel' +3 dB?
- Can a ridge be determined only once, out of a possible three times during passage?
- Can no ridge be determined?
- Is the blot ridginess less than 'minRness'?
- Is the mean engine ridge energy greater than 'enLevel' +'enRange'?

The segment shift and energy-width fraction are no longer used, since these are not reliable signs of a false alarm.

Classifying vehicles

Truck detection has been somewhat elaborated in this version of the system. A blot-ridge combination belongs to a truck when either the blot width is greater than 'tWidth' or the mean ridge energy is greater than 'enLevel' +'tRenAboveEn'. The rationale behind this idea is that trucks can be long vehicles, or vehicles that are not necessarily very long but have a powerful (diesel) engine. Furthermore, in all cases the mean blot energy must be 'tAboveEn' dB above the expected energy level.

Scooters are detected as follows:

- Their maximum blot energy must be 'scMaxAboveEn' above the expected energy level.
- Scooter blot ridginess must be higher than 'scRness'.
- The blot center segment must lie lower than 'scCseg'.
- The energy of the lower engine ridges (between segments 'rLow' and 'rHigh') must be below the expected energy level, but above 2/3 of it.

Three or more of these conditions must be met.

Output

The output of this system is a 16 byte ascii string which is sent to the PC com port at 2400 baud, 8N1 (8 data bits, no parity bit, 1 stop bit). An example is the following: #04220103112100$ means a car (04) has been detected on the 22th of January, 2003, at 11:21:00 am.

In addition to this, the system also maintains a log file, to which it writes its detections, their time stamps and the most important blot and ridge parameters. The log file can be used for lookup and troubleshooting purposes. In the previous chapter an example of a log file entry can be found.

9. Field evaluation

During development, the systems have regularly been tested. Improvements to the systems have subsequently been made, based on these tests. The test results are summarized in this chapter.
There are a few important terms that are frequently used:

- A *detection* (sometimes called a *hit*) is made when the system notices a vehicle passing. During detection, it is not yet known to which class the vehicle belongs.
- After detection, the vehicle is *classified*. This classification can be either *correct* or *incorrect*. The annotations made by the person who did the recording, are used as controls.
- Vehicles are *missed* when they really pass by but are not detected by the system.
- *False alarms* are detections when there are no vehicles present.

The number of hits and the number of misses always add up to 100 %. The number of false alarms is divided by the total number of passages to place it in perspective. In theory, the resulting percentage could exceed 100 %.

INITIAL TESTING

To test the initial system, a sound file was made containing two cars, all 'Ranum' scooters (3) and all 'Maarhuizen' trucks (2). Every vehicle was detected and correctly classified and no false alarms occurred. However, this data set is of course too small to draw far-reaching conclusions about the detector quality.

SCORE COMPUTATION

The scores are computed by a script that compares the system output with the annotation file. A detection must be close enough in time to a corresponding annotation entry (a maximum difference of 2 seconds is allowed because, when annotating, it is hard to ascertain the exact time of passage). If the classification and the entry correspond, the number of correct hits is incremented. If they do not, 1 is added to the number of incorrect hits. At the end of the matching, all remaining (unmatched) entries are counted as passages that were missed and all remaining detections as false alarms. Then all numbers are converted to percentages and written to a detection information file.

APPLICATION I

The Utrecht data were analyzed once by the original system and on the basis of the outcome some modifications were made, most notably the addition of the bus detector described in the previous chapter.
Testing is done in three stages, all on the same data set:

- First we test the detection and classification of all vehicle classes of interest (cars, scooters, trucks and buses) in a representative part of the file of 10 minutes long. Cars, which had not been annotated in Utrecht, had to be

annotated afterwards by a human listener. This test is very important to ascertain the reliability of the system.

- Then the detection and classification of *special* vehicles (all vehicles but cars) over the whole file are tested.
- The last test involves the detections from the RIVM measuring post. This post registers all continuous exceedings of a certain threshold value that last at least 2 seconds. For this project, the threshold has been set to 70 dB(A). We test, again for the 10 minute part, which percentage of the 'RIVM events' is explained by our system.

The results of the first test:

- Detected: 84 %
  - Subsequently classified correctly: 83 %
  - Subsequently classified incorrectly: 17 %
- Missed: 16 %

Total number of vehicles: 117
Of which...
cars: 107
scooters: 5
trucks: 3
buses: 2

- False alarms (compared to total number): 3 %

Anno. \ System	*Car*	*Scooter*	*Truck*	*Bus*	*Missed*	*Total*
Car	76 (84%, ± 6.3%)	8 (9%, ± 4.9%)	1 (1%, ± 1.8%)	5 (6%, ± 4.0%)	17	107
Scooter	0 (0%, ± 0%)	4 (80%, ± 29%)	1 (20%, ± 29%)	0 (0%, ± 0%)	0	5
Truck	1 (33%, ± 45%)	0 (0%, ± 0%)	0 (0%, ± 0%)	2 (66%, ± 45%)	0	3
Bus	0 (0%, ± 0%)	0 (0%, ± 0%)	0 (0%, ± 0%)	2 (100%, ± 0%)	0	2

Table 3: Confusion matrix for all vehicle classes in a 10 minute sample from the Utrecht data set

- The missed passages are almost all partially masked, for instance by crossing traffic, or overlapped. We did a test to see which percentage of the misses is actually caused by overlap or crossing. A Matlab script was made that placed 117 passages of average length 2.5 seconds randomly in a 10 minute time span. The mean overlap after 10,000 runs is 11%. This means that a good 2/3 (11 out of 16%) of the misses can be explained by overlap or crossing.
- Assuming that we can correct for this effect using statistical techniques, the total detection percentage equals 69% +15% +11% +3% =**98%** and thus gives a very good estimate of the total number of vehicle passages.
- The table above is a *confusion matrix*. The leftmost column contains the vehicle classes as annotated, the top row the system classifications. The cells on the diagonal (in italic) represent correct classifications, the off-diagonal cells misclassifications. Because here we are interested in the classification performance only, we compute the correct/incorrect percentages (in brackets) without taking the number of misses into account. Of course, the rows must still add up to the total number of vehicles for their corresponding classes.
- Between the brackets, after the percentages, are 90% (α =0.1) confidence

intervals (Hogg and Tanis 1997). They give a good measure of how reliable our percentages are, given the number of vehicles in a class.

- One can see from the confusion matrix that both the bus and the scooter detector are not specific enough and claim too many cars. However, this observation cannot be proven statistically, as the percentages are virtually meaningless due to the magnitude of their confidence intervals.
- False alarms in this part were caused by a strong gust of wind in combination with a series of cars passing, which led to an extra detection, and a container that was rolled over the sidewalk, causing a sound a bit like a heavy vehicle.

The results of the second test:

- Detected: 97 %
  - Subsequently classified correctly: 55 %
  - Subsequently classified incorrectly: 45 %
- Missed: 3 %

Total number of special vehicles: 70
Of which...
scooters: 16
trucks: 38
buses: 16

The corresponding confusion matrix:

Anno. \ System	***Scooter***	***Truck***	***Bus***	***Missed***	***Total***
Scooter	12 (86%, ± 18%)	0 (0%, ± 0%)	2 (14%, ± 14%)	2	16
Truck	6 (16%, ± 9.7%)	18 (47%, ± 13%)	14 (37%, ± 13%)	0	38
Bus	1 (6%, ± 10%)	1 (6%, ±1 0%)	14 (88%, ± 14%)	0	16

Table 4: Confusion matrix for special vehicles in the Utrecht data set

- Incorrect classifications mainly occur with trucks, since they form a class that is hard to define. The definition of a truck that we have used, viz. every transportation vehicle that is larger than a van, is rather arbitrary. It is especially hard to distinguish trucks from buses. A better class definition, more grounded in physics, is needed to reach better results, as well as more data.
- In total two special passages have been missed; both are scooters that pass relatively slowly (and therefore quietly) on the cycle lane on the other side of the street.

The results of the third test:

In this last test, we compare our detections with RIVM's detections (with a 70 dB(A) threshold applied). We are interested in ascertaining the percentage of RIVM events that coincides with our system's events.

- Events explained: 89 %
- Missed: 11 %

Total number of RIVM events: 19

- Passages within events explained: 91 %
- Missed: 9 %

Total number of passages within RIVM events: 54

Events are considered to have been explained when one or more passages detected by our system fall within their time interval. As can be seen, almost all detections (17 out of 19) of the RIVM measuring post in the 10 minute representative sample are explained by our system. The 2 misses are indeed misses, as there are passages in the annotations at those times. When looking at the individual passages, the score is even slightly better. However, we have to keep in mind the system's reliability as ascertained by tests 1 and 2.
This third test enables us to compare our method with a 'traditional' method (which is very useful as was explained in chapter 2). 69 per cent of the RIVM events actually consists of several passages. This entails that the RIVM method is not a very good way of detecting individual passages and can therefore not be used for counting purposes etc. Our method, on the other hand, is better suited for this.

APPLICATION II

The Lochard system has been trained on the individual plane sound samples provided by Lochard (see the data overview in Appendix I). As test sets, the Australia and three of the four Schiphol recordings have been used. One of the Schiphol files (the Amsterdamse Bos recording) has been excluded because it is qualitatively very different from the training data. In the discarded recording, planes fly by at a distance of several kilometers, instead of 50 to 200 meters as in the other sets. It is possible to adapt the system to this new situation, but then more data are needed. Unfortunately, this is the only Schiphol file that includes propellers.

- Of the Australia set, all (11) jets and (5) propellers were detected and recognized correctly.
- The Kaagbaan set (ascending planes at about 200 meters, highway closeby):
    - 11 jets detected (85 %)
    - 2 jets missed (15 %)
    - 13 jets total
    - 3 false alarms (23 %, compared to total number of jets)
- The Zwanenburgbaan set (descending planes at about 50 meters, highway closeby):
    - 11 jets detected (85 %)
    - 2 jets missed (15 %)
    - 13 jets total
    - 3 false alarms (23 %)
- The Buitenveldert set (ascending planes at about 200 meters):
    - 11 jets detected (85 %)
    - 2 jets missed (15 %)

- 13 jets total
- 1 false alarm (8 %)

APPLICATION III

The VCS system has been trained on the first 15 minutes of the file and on some specific passages (scooters and trucks mainly) from elsewehere in the file. After finetuning, it has been tested using the whole recording (75 minutes long). The results are as follows:

• Detected:	96 %
• Subsequently classified correctly:	97 %
• Subsequently classified incorrectly:	3 %
• Missed:	4 %
Total number of vehicles:	121
Of which...	
cars:	103
scooters:	11
trucks:	7
• False alarms (compared to total number):	5 %

The classifications that go wrong are mainly scooters that are classified as cars. The misses consist of very slow moving cars, whose blots contain relatively little energy. They tend to be considered background by the system, as their energy levels are almost equal to those of vehicles on the elevated road which should be in the background. False alarms are caused by high energy passages on the dike (e.g. accelerating trucks).

10. Conclusions and recommendations

RESEARCH QUESTION

Having seen the results of our systems, time has come to draw some conclusions. The first question that needs to be answered is of course the research question, posed in chapter 4. For the sake of completeness, it is repeated here:

Is it possible to build a CPSP based system that can detect and classify vehicle passages?

First, we will consider the question as a whole. This project has resulted in three working applications that involve both vehicle detection and classification. All these systems have three important properties:

- They can function without human intervention and therefore they are truly *autonomous systems*. We have succesfully automated a part of human cognition and applied it usefully. Therefore, this project is a good example of Artificial Intelligence.
- Due to the use of the background model, the systems are also very *adaptive* and *environment independent.* They can operate in almost any environment, as long as the relevant signal parts are above background level. The background itself, whatever its characteristics, is modeled and this model is constantly updated.
- The applications have been derived from a very *generic* vehicle detection and classification system. The same techniques (background model, smoothing) and structures (ridges, blots) can be used to detect vehicles as different as cars and airplanes. In this context it is interesting to note that the initial system was turned into an airplane detector in just one day of work.

To look more closely into the different aspects of the research question we split it into two parts:

- Is it possible to build a CPSP based system that can detect vehicle passages?
- Is it possible to subsequently classify these sounds?

The first question can be answered affirmatively. In our most elaborated system, application III (the 'VCS system'), the miss rate is at a commercially acceptable low (4 per cent, where VCS would have agreed with 5 per cent). Application II, the airplane detection system (15 per cent) is no match for this but still has a good score when we take into account the much less optimized state of this system and the fact that there was constant highway noise on the recording with the same sound level as the airplanes. The miss rate for application I, the city street system (16 per cent), may seem high, but as we have seen, this can be partially explained by overlap and can be corrected for.
The second question is more difficult to answer. The correct recognition rate (97 per cent) for the VCS system is high enough to be acceptable for camera surveillance purposes, but we have to keep in mind that there is no great number of special vehicles in that data set. The city street application's results are more relevant in this case. A 83 per cent correct rate is not acceptable. When not taking cars into account, it

even drops to 55 per cent. This is better than chance (there is 33 per cent chance of predicting a special vehicle class correctly), but not good enough for commercial purposes. However, we should not forget that the definition of trucks that was used was too arbitrary and likely caused a large number of incorrect hits. Furthermore, this is the first application of its kind, so it is not optimized in any way. Viewed this way, 83 per cent is a good starting point.
Based on these considerations our conclusion must be that vehicle detection using CPSP output is a viable method, certainly at close distance from the sources. There are some promising classification results, but the systems will have to be improved to get these into acceptable regions. Further research is needed to ascertain whether the system, with appropriate modifications, can be used in quiet areas.

COMPARISON WITH OTHER APPROACHES

There are two other important issues, both concerning the 'surplus value' of the methods used in this project. Why did we use CPSP instead of more traditional methods and what distinguishes our system from, for example, the RIVM methods described in chapter 2?

To start with the first question, CPSP has been used because it is the only sound analysis tool that maintains the continuity of the signal. However, we do not make use of this property on the higher levels of the systems we developed, because we apply smoothing with a time constant of 0.5 seconds or higher. After such an operation, little is left of the original continuity. Seen from this perspective, we might just as well have used a Fast Fourier Transform (FFT) or a similar discontinuous technique. However, there might be advantages in using CPSP in the stages before smoothing. Some additional research is needed to see whether this is the case. Remember that using CPSP and examining if it was suitable for vehicle detection was a basic assumption of this project. It may not have had a particular advantage in this research, but perhaps in future systems, for other signals with a smaller time constant, it will be indispensable. Then it will be very useful that we have already found out how to deploy CPSP techniques in this context.

In what ways is our system different from -or maybe even better than- traditional sound measuring or calculating techniques? In this project, three great advantages have come to the fore:

- When measuring sound levels, our system only takes into account the contributions from the relevant sources. Other sources and the background are discarded, as opposed to traditional measurements where only the *overall* sound level counts. Therefore, our system gives a more truthful measure of the sound level (or disturbance). It seems that indeed, as speculated in chapter 2, our system can be used to improve the quality of, for instance, RIVM's measurements by only selecting vehicle sources and correcting for other, local sources. It enables the verification of calculation practice (models) by measurement.
- Due to the use of a background model that is adaptive, the system is able to function in varying environments. Differing weather conditions, the appearance and disappearance of agricultural machines etc. do not hamper the

functioning of the system.

- The use of blots and ridges makes it possible to give an extensive qualitative description of the vehicle passage. This information can be used in the classification process but also to give more psycho-acoustical measures of the passage. Besides the mean and maximum energy level the system could, for instance, give a 'nuisance factor' as output. This idea has not yet been implemented, but there is interest from RIVM in such a system. It will have to be found out what features are relevant for such a measure. Energy (loudness) will undoubtedly be an important one, but also the tonality (the extent to which the signal is dominated by periodic components from the same source, in our systems measured by the ridginess) seems to play an important part. For instance, scooter sound is for most people more annoying than car sound, even at equivalent loudness levels.

SHORTCOMINGS

Throughout this project, the emphasis has been on *developing* a system, from the first design to a working application. Due to the limited amount of time available, this focus has sometimes been at the expense of other important aspects. We will mention some of these 'gaps' here and express the hope they will be addressed in future research.

- In chapter 5, we spoke of the bottom-up approach. Due to lack of time, we decided not to use a Kohonen map to derive classes for us, but to use pre-defined, visually based, classes (car, truck etc.) instead. As we have seen in the city street application, this did not always work out well. The distinction between trucks and buses was not made well by the system. At this stage, applying a Kohonen map would have been very useful. Especially since even RIVM itself indicates that the classes it uses (light, heavy vehicles etc.) are sometimes hard to distinguish, it is highly recommended that a Kohonen network be applied to vehicle data to derive a new set of boundaries. After training, this network will have to be analyzed to discover what features it uses to discern the different vehicle classes. There was no time left to implement this idea, but it will have to be before the research continues.
- In addition to this, one could study the physics of vehicle sounds much more closely and use the knowledge that approach yields to further enhance the applications.
- Many of the parameter values in the applications seem to come 'out of the blue'. They have been derived empirically, by experimenting and testing for improved performance. This has led to a situation where the applications function more or less satisfactorily but we do not know exactly why this is so. Why must a certain parameter be set to value X and what is its physical meaning? These kinds of questions will have to be answered if we want to gain more insight into (vehicle) sound recognition. Recall that in chapter 3, it was said that the knowledge of the physical properties of sound sources is very important in this context. We have had a great amount of experience with vehicle sounds in the course of this research, and now the time has come to translate this back into knowledge that can be used in the future.

- The top left part of figure 7 shows how the system can detect and classify sounds in the background the same way as in the foreground. For instance, a distant highway may generate a considerable amount of noise but soon becomes part of the background in our system. We could build a classifier that searches for highway evidence in the background and maybe even competes with other classifiers. This idea has not been implemented, but it will probably be very useful in quiet areas, where distant, constant sources such as highways and airplanes flying over at great height are responsible for much of the noise.

11. Bibliography

Chapter 1: Introduction

Berends, W. (2002), "Waar geniet je nog in stilte? - resultaten van een enquete onder recreanten", Stichting Natuur en Milieu, Utrecht.

Van den Berg, G.P. (2002a), "Op zoek naar stilte - meting van indicatoren voor stilte in recreatieve (natuur)gebieden in de Randstad", Natuurkundewinkel, Groningen.

Chapter 2: Current practice

Bekebrede, G. (ed.) (1994), "Luchtvaartlawaai - voordrachten gehouden op de bijeenkomst van 13 september 1994 te Utrecht", NAG-Journaal, 124.

Brouwer, H. et al. (1999), "Vliegtuiggeluid en verstoring: voorspellen, bewaken en beheersen", NAG-Journaal, 147.

Jabben, J., Potma, C. & Swart, W. (2001), "Monitor resultaten geluid 2000 - overzicht monitormetingen omgevingsgeluid 2000", RIVM, Bilthoven.

Moerkerken, ir. A. & Middendorp, ing. A.G.M. (1981), "Berekening van wegverkeersgeluid - een toelichting op de standaard rekenmethode I uit het reken- en meetvoorschrift verkeerslawaai artikel 102 Wet Geluidhinder", Staatsuitgeverij, Delft.

Nooteboom, S.G. and Cohen, A. (1995), "Spreken en verstaan – een nieuwe inleiding tot de experimentele fonetiek", 5th ed., Van Gorcum, Assen.

Thompson, dr. D.J. et al. (1997), "Trein- en wegverkeersgeluid", NAG-Journaal, 139.

VROM, Department of (2001), "Reken- en meetvoorschrift railverkeerslawaai '96", VROM, The Hague.

Chapter 3: Theoretical foundations

Andringa, T.C. et al. (1999), International Patent Application WO 01/33547, "Method and Aparatuses for Signal Processing".

Andringa, T.C. (2002a), "Continuity Preserving Signal Processing" (PhD thesis), University of Groningen, Groningen.

Andringa, T.C. (2002b), presentation on *Continuity Preserving Signal Processing*, handout provided at TNO, Delft.

Chapter 4: Practical foundations

Van den Berg, G.P. (2002b), personal communication.

CHAPTER 5: RESEARCH QUESTION

Andringa, T.C. (2002c), personal communication.

Haykin, S. (1999), "Neural networks: a comprehensive foundation", 2nd ed., Prentice Hall.

CHAPTER 7: GENERAL SYSTEM DEVELOPMENT

Hogg, R.V. and Tanis, E.A. (1997), "Probability and statistical inference", 5th ed., Prentice Hall.

CHAPTER 8: APPLICATIONS

Dix, A.J., et al. (1998), "Human-computer interaction", Prentice Hall Europe.

DANKWOORD

Het heeft even geduurd maar nu is het zover: het is volbracht! (Johannes 19:30 / Els Borst)
Dit was niet mogelijk geweest zonder de volgende mensen, die ik bij dezen dan ook hartelijk wil bedanken voor hun hulp, steun, troost, inspiratie en wat dies meer zij. Voor hen die ik vergeten ben alvast een hartgrondig excuus!

Tjeerd, voor de prima dagelijkse begeleiding

Frits, voor alle theoretische en praktische hulp en de opnameappatuur die ik steeds maar weer mocht lenen

Esther en Lambert, voor de begeleiding vanuit de studie

Peter en Maartje, voor alle cochleaire en wiskundige adviezen

Gert-Jan (GJ), voor de programmeerhulp en de saucijzenbroodjes

Jan Jabben en Charlos Potma van het RIVM

Joost, voor het goede contact met Lochard

Wendy, voor het gezelschap gedurende de eerste maanden

En natuurlijk **Ron**, voor alle voortgang

12. Appendices

APPENDIX I: RECORDINGS

Paddepoel

On July 5th, 2002, Frits van den Berg recorded for two hours just outside the Groningen city limits, near the Zernike science park, alongside the Paddepoelsterkanaal. This location has many similarities with a quiet area, because it has a rural quality and many natural sounds can be heard (sheep, rushing reeds, birds etc.) but the peace is often, as with quiet areas, disturbed by unnatural sources such as passing boats, vehicles passing by, distant freeway traffic noise and noise from a building site at Zernike.

Maarhuizen

On July 26th, 2002, recordings were made together with Maya van den Berg. We recorded road noise from the provincial road from Groningen to Lauwersoog (the N361) near Winsum. The measurements were made on several locations along the road, the distance varying from 10 to 20 meters.

Ranum

Three days later, Maya van den Berg made some additional recordings at the same site, this time at a greater distance from the road (200 meters). This was done to study the effects of distance on road noise. In quiet areas, most of the traffic noise comes from distant sources. A lot of agricultural machine noise was picked up too, a frequent source of quiet area disturbance.

VCS Waalwijk

VCS is a company specialized in camera surveillance. They had approached Sound Intelligence to work together on a camera system enhanced with sound detection. The pilot system was to be developed at the VCS building itself, at the Waalwijk industrial zone. This area is deserted at night, so good security measures must be taken. If it is known in advance that a vehicle, be it a scooter, a car or a truck, is approaching, the security camera can be pointed at it so that, for instance, the license plate can be scanned.
For our detectors to work well, sufficient training data is needed. On the 14th of October, we went to VCS to make some test recordings and study the surroundings of the building, mainly the nearby roads. Based on this visit a report was written, containing instructions for useful test data collection by VCS personnel.

Breda Central Station

Later that day, we went to the Breda central train station to gather data for another SI (pilot) project. The NS, the former Dutch state railroad company, aims to improve

their camera surveillance at stations using sound detection. Until now, camera operators had to look at a lot of monitors simultaneously, which makes it very hard to spot incidents directly. Thus, performance can be improved by presenting the images from cameras pointed at an area where sound detectors have picked up an 'interesting' sound (such as a cry for help or breaking glass) more urgently.
For the pilot project, two camera-microphone combinations will be installed at the station. At both locations, all occurring sounds were recorded for two hours. The data were used by HuQ scientists to make a station background model and models for all sounds normally occurring at the Breda station. These models are needed to make the distinction between normal sounds and sounds requiring attention.

VCS (II)

Unfortunately, the VCS people failed to make the necessary recordings so we were asked to go there again and do it ourselves. On the 29th of November, 2002, we recorded passages of cars, trucks and one scooter (thanks to a VCS employee who owned one).
The recordings were made approximately two meters from the road that lies in front of the VCS building (Havenweg). A complication is the elevated road that lies on a dike 5 meters high, 10 meters behind the Havenweg. A lot of heavy traffic (trucks) passes over this dike.
There are no buildings between the Havenweg and the dike, so there is no reflection of sound. Not far off (about 200 meters) is a highway, which can sometimes be heard, dependent on the direction of the wind and the amount of noise on the dike.
Most of the passages are from cars, vans (which were counted as cars in order not to make the classification too complicated) and trucks. The road is busiest during peak hours (around 9 a.m. and 5 p.m.) and lunch time (at noon). Approximately once per hour a cyclist passes, which holds true for pedestrians as well.

Utrecht

On Wednesday the 22th of January, 2003, one and a half hour of recording was made in the Constant Erzeij Street in Utrecht. The DAT recorder was positioned next to the RIVM measuring post, approximately 1 meter from the road.
The street consists of two sidewalks, two cycling lanes and two car lanes. Houses (three stories high) are at both sides.
The weather was cloudy and there were occasional showers and strong gusts of wind. The temperature was 6 degrees Celsius.
During recording, all special vehicles (scooters, trucks and buses) were annotated. Annotating car passages was impossible, due to the their sheer number. However, a representative 10 minute sample was fully annotated afterwards solely on the basis of hearing.

Lochard

To aid the development of the plane detector, Lochard provided us with relevant data complete with annotations. These included:

- A set of 7 jet overflights.

- A set of 4 propeller plane overflights.
- One jet mixed in with house music.
- A set of non-plane test data ranging from plane-like generators to absolutely not plane-like barking dogs.
- Two hours of DAT recording made near an airport in Australia, including both jets (11) and propellers (5). Other vehicles can be heard, but they are not very loud.
- Four half-hour recordings made near Schiphol.
    - Kaagbaan: 13 ascending jets that fly over at about 200 meters. Very close to a highway, whose sound level is approximately as high as the planes'.
    - Zwanenburgbaan: 13 descending jets that fly over at about 50 meters. There is also a highway nearby.
    - Buitenveldert: 13 ascending jets at approximately 200 meters. Recorded close to a McDonald's restaurant.
    - Amsterdamse Bos: 9 jets and 3 propellers flying over at a distance of several kilometers. Other sounds include barking dogs and playing children.

APPENDIX II: MATLAB SOURCE CODE

In this chapter all the vehicle detector source code is presented. The code was written in Matlab, but later automatically converted to C so as to be able to embed it into the SI system. However, the generated code is almost unreadable, so we do not present it here.
The sections reading parameter values from file have in the C system been removed from the functions themselves and made *static*. That way it is necessary to read the values only once.

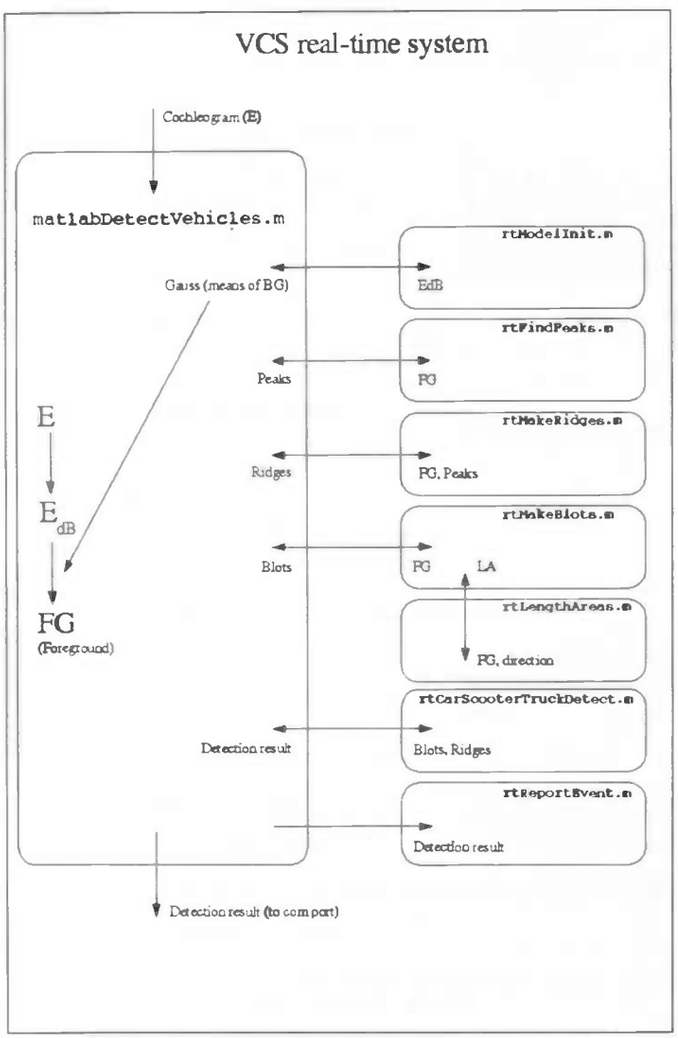

Figure 25: Overview of the VCS real-time system code

A. matlabDetectVehicles.m

This is the "main" function that is called every 0.5 seconds with the latest 8 columns from the cochlea.
The 'sendToPort' function at the end is a piece of code written completely in C. Therefore it cannot be called from Matlab, so the function calls have to be activated

(uncommentarized) in the C code resulting from the compilation process. It sends the system output to the com port.

```
function matlabDetectVehicles(E1, E2, E3, E4, E5, E6, E7, E8);

% realtime cochleogram analyzer made for the VCS demo
% ---------------------------------------------------
% Reads the current cochleogram, analyzes
% the signal and detects and classifies vehicles (cars,
% scooters and trucks) if present.
% Input: a cochleogram of 4 seconds long.
% Output: the detection result, 0 if nothing detected,
% otherwise one or more chars containing the classification
% result (c, s, t).
% Detection results are also written to 'logFile'.
% If 'dMode' is toggled, the cochleogram is written to
% 'cochs'.

% Read parameters from 'param.conf'.
fid = fopen('param.conf');
dMode = fscanf(fid, '%d', 1);          % 0: do not, 1: do write the cochleogram to file
scalingF = fscanf(fid, '%d', 1);       % the scaling factor for the cochlea energy
Emax = fscanf(fid, '%d', 1);           % the maximum energy level in dB
enLevel = fscanf(fid, '%d', 1);        % the expected (mean) energy level
FGlimit = fscanf(fid, '%d', 1);        % the foreground rises so much above background
level
percBG = fscanf(fid, '%d', 1);         % percentage of max. energy considered
background
fclose(fid);

E = [E1 E2 E3 E4 E5 E6 E7 E8];
E = E ./ scalingF;                     % scaling due to low-pass cochlea filter

if dMode == 1
  fid = fopen('cochs', 'a');
  fwrite(fid, E, 'double');
  fclose(fid);
end

% initialization of constants and matrices:
path = [pwd '/'];
nSeg = size(E, 1);                     % number of segments of cochleogram
cochLength = size(E, 2);               % cochleogram length (smoothed frames)
Cor = (linspace(5, 2, nSeg) .^ (1 ./ .15));
Cor = Cor ./ max(Cor);
E = E .* (Cor' * ones(1, cochLength));

EdB = zeros(nSeg, cochLength);         % cochleogram, logarithmic (dB) domain
EfgMask = zeros(nSeg, cochLength);     % foreground mask
Efg = zeros(nSeg, cochLength);         % foreground selection

% convert cochleogram to dB and remove values above maximum:
EdB = round(20 * log10(max(1, E)));
EdB(find(EdB > Emax)) = Emax;

% initialize the background model using the lower 'percBG' % of the EdB values:
gauss = rtModelInit(EdB, percBG, Emax);

% assign parts to foreground:
EfgMask = EdB > ((gauss + FGlimit) * ones(1, cochLength));
Efg = EdB .* EfgMask;
percFG = sum(Efg) / sum(EdB) * 100;

% find interesting structures:
Peaks = rtFindPeaks(Efg);                    % calculate peaks
Ridges = rtMakeRidges(Efg, Peaks);           % form ridges out of them
Blots = rtFindBlots(Efg, Ridges);            % construct blots

% detect and classify passing vehicles:
% (detResult is the ultimate return value)
detResult = 0;
if ~isempty(Blots) & Blots(1).w > 0
```

```
  [detResult, timeStr] = rtCarScooterTruckDetect(Ridges, Blots, path, enLevel, Emax,
percFG);
end

% Calling the send-to-com-port function: activate in .c-file
if detResult == 2
  %sendToPort('2');
end
if detResult == 4
  %sendToPort('4');
end
if detResult == 8
  %sendToPort('8');
end
if detResult == 0
  %sendToPort('0');
end
```

B. rtModelInit.m

```
function gauss = rtModelInit(Estart, PercBG, Emax);

% Purpose: computing per segment the mean of the (supposedly) Gaussian
% background noise (mu).
% Input: the cochleogram (in dB), the percentage of the maximum energy
% considered background, the maximum energy.
% Output: the mu value per segment.

nBins = Emax;                          %there are bins for all possible energy values
nSeg = size(Estart, 1);
nFrames = size(Estart, 2);
updateLength = round(nFrames / 2);
totPerSeg = 1;
reward = 0.1;
Ehist = zeros(nSeg, nFrames,2);
Hgram = ones(nSeg,nBins) ./ (nBins / totPerSeg);    % initialize histogram
gauss = zeros(nSeg, 1);
[nrElem, perc] = hist(Estart, nBins);
BGhigh = perc(round((PercBG / 100) * PercBG));      % compute what is background
Ehist = (Estart <= BGhigh) .* Estart;

for jj = nSeg : -1 : 1

  % for every segment, update the histogram:
  [Eupdate, cum] = unique(Ehist(jj,:));
  nrUpdates = diff([0 cum]);
  if Eupdate ~= 0
    Hgram(jj, Eupdate)=Hgram(jj, Eupdate) + (reward * nrUpdates);
    Hgram(jj, :)=Hgram(jj, :) ./ (((totPerSeg + reward) / totPerSeg) *
sum(nrUpdates));
  end

  % find the bins that are 0.01 or higher and compute the mu values from them:
  if jj == 80
    greaterThanZero = find(Hgram(1, :) > 0);
    gauss(80) = round(mean(greaterThanZero));
  else
    greaterThanZero = find(Hgram(jj, :) > 0.01);
    if ~isempty(greaterThanZero)
      gauss(jj) = round(mean(greaterThanZero));
    else
      gauss(jj) = gauss(jj + 1);
    end
  end

end
```

C. rtFindPeaks.m

```
function P = rtFindPeaks(E);
```

```
% Purpose: find peaks (local energy maxima) in the cochleogram.
% Input: the cochleogram.
% Output: a structure containing all peaks, information about whether they are
% part of a ridge (set to all zeros) and their energy values (in dB).

nFrames = size(E, 2);
nSeg = size(E, 1);

for fr = 1 : nFrames
  P(fr).seg = find((E(1 : end - 2, fr) < E(2 : end - 1, fr) & E(2 : end - 1, fr) > E(3
: end, fr)) | (E(1 : end - 2, fr) <= E(2 : end - 1, fr) & E(2 : end - 1, fr) > E(3 :
end, fr)) | (E(1 : end - 2, fr) < E(2 : end - 1, fr) & E(2 : end - 1, fr) >= E(3 :
end, fr))) + 1;
  P(fr).member = zeros(1, length(P(fr).seg));
  P(fr).en = E(P(fr).seg, fr);
end
```

D. rtMakeRidges.m

```
function R = rtMakeRidges(E, P);

% Purpose: ridge formation.
% Input: the cochleogram and the peak structure.
% Output: a structure containing all the ridges and their properties.

ridgeId = 1;
nFrames = size(E, 2);
nSeg = size(E, 1);
R = [];

% work frame by frame:
for fr = 1 : nFrames - 1
  % check all peaks in the current fame:
  for s = 1 : size(P(fr).seg, 1)
    % find a peak that is not yet member of a ridge and try to build a ridge from
    % there:
    if P(fr).member(s) == 0
      RR.start = fr;
      ii = fr;
      currentSeg = P(fr).seg(s);
      currentEn = P(fr).en(s);
      RR.seg(1) = currentSeg;
      P(fr).member(s) = 1;
      cont = 1;
      jj = 1;
      while ii < nFrames & cont == 1
        cont = 0;
        ii = ii + 1;
        jj = jj + 1;
        if ~isempty(P(ii).seg)
          % look for a next peak on the same segment:
          next = find([P(ii).seg] == currentSeg);
          if ~isempty(next)
            RR.seg(jj) = P(ii).seg(next);
            P(ii).member(next) = 1;
            currentSeg = P(ii).seg(next);
            currentEn = P(ii).en(next);
            RR.en(jj) = 10 ^ (currentEn / 10);
            cont = 1;
          else
            % look at segments at distance 1 or 2 from the current:
            possNext = find([P(ii).seg] == currentSeg + 1 | [P(ii).seg] == currentSeg
- 1 | [P(ii).seg] == currentSeg + 2 | [P(ii).seg] == currentSeg - 2);
            if ~isempty(possNext)
              possNext2 = possNext(find([P(ii).member(possNext)] == 0));
              if ~isempty(possNext2)
                next = possNext2(find(min(abs([P(ii).en(possNext2)] - currentEn))));
                if ~isempty(next)
                  RR.seg(jj) = P(ii).seg(next);
                  P(ii).member(next) = 1;
                  currentSeg = P(ii).seg(next);
                  currentEn = P(ii).en(next);
                  RR.en(jj) = 10 ^ (currentEn / 10);
```

```
                cont = 1;
              end
            end
          end
        end
      end

      if ii == nFrames
        if cont == 1
          RR.end = ii;
        else
          RR.end = ii - 1;
        end
      else
        RR.end = ii - 1;
      end

      % if the ridge is long enough, form the structure:
      if (RR.end - RR.start) > 1
        R(ridgeId).start = RR.start;
        R(ridgeId).seg = RR.seg;
        R(ridgeId).end = RR.end;
        R(ridgeId).m = round(mean(RR.seg));
        R(ridgeId).center = round((RR.start+RR.end) / 2);
        R(ridgeId).nEn = RR.en;
        R(ridgeId).en = round(10 * log10(mean(RR.en)));
        ridgeId = ridgeId + 1;
      end
      clear RR;
    end
  end
end
```

E. rtMakeBlots.m

```
function Blots = rtMakeBlots(E, Ridges);

% Purpose: constructing blots from linked areas.
% Input: cochleogram, ridge structure (needed to calculate ridginess).
% Output: a structure containing all blots and their properties.

nSeg = size(E, 1);
upParts = (diff([zeros(1, nSeg)' E], 1, 2) >= 0) .* E;
downParts = (diff([zeros(1, nSeg)' E], 1, 2) < 0) .* E;
% compute linked areas:
[upPart] = rtLinkedAreas(upParts, 0);
[downPart] = rtLinkedAreas(downParts, 1);

B(1).id = 0;
Blots(1).w = 0;
kk = 1;
newId = 1;
maxFrShift = 3;

% pair up consecutive linked areas (rising and falling):
if ~isempty(upPart) & ~isempty(downPart)
  for jj = 1 : nSeg
    possUps=find([upPart.seg] == jj & ~isempty([upPart.limit]) & [upPart.length] > 1);
    for up = 1 : size(possUps, 2)
      if ~isempty(downPart)
        possDowns = find([downPart.seg] == jj);
        down = find(abs([downPart(possDowns).left] - [upPart(possUps(up)).right]) < 2
& ~isempty([downPart(possDowns).limit]));
        if ~isempty(down)
          B(kk).seg = upPart(possUps(up)).seg;
          B(kk).center = upPart(possUps(up)).right;
          B(kk).left = upPart(possUps(up)).limit;
          B(kk).right = downPart(possDowns(down(1))).limit;
          B(kk).en = upPart(possUps(up)).top;
          B(kk).startEn = B(kk).en;
          B(kk).id = newId;
               newId = newId + 1;
```

```
              % try to add the linked area to a blot; if this does not succeed,
              % it will be the start of a new blot:
          for ll = 1 : kk - 1
            if B(ll).seg == (B(kk).seg) - 1 & abs(B(ll).center - B(kk).center) <=
maxFrShift & B(kk).en >= B(ll).startEn - 3
              B(kk).id = B(ll).id;
              B(kk).startEn = B(ll).startEn;
              newId = newId-1;
            end
          end
          kk = kk + 1;
        end
      end
    end
  end
end

% compute properties off all the blots:
ii = 1;
jj = 1;
while jj <= max([B.id])
  bb = find([B(:).id] == jj);
  if ~isempty([B(bb(:)).left]) & ~isempty([B(bb(:)).right])
    en = round(mean([B(bb(:)).en]));
    maxEn = round(max([B(bb(:)).en]));
    left = round(mean([B(bb(:)).left]));
    top = B(bb(1)).seg;
    w = abs(round(mean([B(bb(:)).right])) - left);
    right = left + w;
    h = abs(B(bb(size(bb,2))).seg - top);
    bottom = top + h;
    center = round(mean([B(bb(:)).center]));
    peakedness = 0;
    % calculate peakedness of blot
    summits = find(E(top:bottom - 2, center) <= E(top + 1 : bottom - 1, center) &
E(top + 1 : bottom - 1, center) > E(top + 2 : bottom, center)) + 1;
    valleys = find(E(top:bottom - 2, center) >= E(top + 1 : bottom - 1, center) &
E(top + 1 : bottom - 1, center) < E(top + 2 : bottom, center)) + 1;
    if ~isempty(summits) & ~isempty(valleys)
      peakedness = mean(E(top - 1 + summits, center)) - mean(E(top - 1 + valleys,
center));
    end
    ridginess = 0;
    segShift = 0;
    % calculate ridginess of blot
    if ~isempty(Ridges) & h > 0
      rr = find([Ridges.start] < center & [Ridges.end] > center);
      ridginess = size(rr, 2) / h;
      % calculate mean segment shift during passage
      rl = find([Ridges.start] < left & [Ridges.end] > left);
      rr = find([Ridges.start] < right & [Ridges.end] > right);
      if ~isempty(rl) & ~isempty(rr)
        segShift = mean([Ridges(rr).m]) - mean([Ridges(rl).m]);
      end
    end
    if h > 10 & w > 0
      Blots(ii).en = en;
      Blots(ii).maxEn = maxEn;
      Blots(ii).left = left;
      Blots(ii).top = top;
      Blots(ii).w = w;
      Blots(ii).right = right;
      Blots(ii).h = h;
      Blots(ii).bottom = bottom;
      Blots(ii).center = center;
      Blots(ii).ridginess = ridginess;
      Blots(ii).peakedness = peakedness;
      Blots(ii).segShift = segShift;
      Blots(ii).centerSeg = round((bottom + top) / 2);
      ii = ii + 1;
    end
    jj = jj + 1;
  else
    jj = jj + 1;
```

```
  end
end
```

F. rtLinkedAreas.m

```
function LA = rtLinkedAreas(E, direction);

% Purpose: form linked areas in the foreground of the cochleogram. The areas can have
% either rising or falling energy.
% Input: cochleogram foreground (E), direction (-1 = down, 0 = up)
% Output: a structure containing all linked areas.

Q = (E ~= 0);
% compute the derivative:
dQ = diff([zeros(size(Q, 1), 1) Q zeros(size(Q, 1), 1)], [], 2);

LA = struct([]);
kk = 1;
for ii = 1 : size(Q, 1)
  start = find(dQ(ii, :) == 1);
  if ~isempty(start)
    for jj = 1 : size(start, 2)
      finish = find(dQ(ii, :) == -1);
      LA(kk).seg = ii;
      LA(kk).left = start(jj);
      LA(kk).right = finish(jj) - 1;
      if direction == 0
        LA(kk).top = E(ii, finish(jj) - 1);
      else
        LA(kk).top = E(ii, start(jj));
      end

      % find the end/beginning of the LA: the point 6 dB lower than the maximum:
      higherThanLimit = find(E(ii, start(jj) : finish(jj) - 1) > LA(kk).top - 6);
      if ~isempty(higherThanLimit)
        if direction == 0
          LA(kk).limit = min(LA(kk).left + higherThanLimit) - 1;
        else
          LA(kk).limit = max(LA(kk).left + higherThanLimit) - 1;
        end
      else
        LA(kk).limit = [];
      end

      LA(kk).length = [finish(jj) - start(jj)];
      kk = kk + 1;
    end
  end
end
```

G. rtCarScooterTruckDetect.m

```
function [detWhat, timeStr] = rtCarScooterTruckDetect(Ridges, Blots, path, enLevel,
maxEn, percFG);

% Purpose: detecting vehicles and classifying them as either a car, scooter or truck.
% Input: ridges, blots, path on which the log file can be found, expected energy
level,
% maximum energy level, percentage of cochleogram that is foreground.
% Output: detection result ('0', 'c', 's', 't') and a string representing the time
% of detection.

fid = fopen('param2.conf');
cSegHigh = fscanf(fid, '%d', 1);          % center segment higher limit
cSegLow = fscanf(fid, '%d', 1);           % center segment lower limit
rLow = fscanf(fid, '%d', 1);              % ridges lower segment limit
rHigh = fscanf(fid, '%d', 1);             % ridges higher segment limit
falseLimit = fscanf(fid, '%d', 1);        % nr of pieces of evidence of false alarm
needed to discard detection
falseH = fscanf(fid, '%d', 1);            % suspicious height below this value
falseH2 = fscanf(fid, '%d', 1);           % even more suspicious height below this
value
```

```
falsePness = fscanf(fid, '%d', 1);        % suspicious peakedness above this value
scMaxAboveEn = fscanf(fid, '%d', 1);      % maximum scooter energy must be so much
higher than expected level
scRness = fscanf(fid, '%f', 1);           % scooter ridginess must be at least this
value
scCseg = fscanf(fid, '%d', 1);            % scooter center segment must be below this
value
scRenBelowEn = fscanf(fid, '%d', 1);      % mean scooter ridge energy must be at least
so much lower than expected level
tWidth = fscanf(fid, '%d', 1);            % truck blot must have at least this width
tAboveEn = fscanf(fid, '%d', 1);          % mean truck energy must be so much higher
than expected level
sepSec = fscanf(fid, '%d', 1);            % minimum nr of seconds assumed between
events
tRenAboveEn = fscanf(fid, '%d', 1);       % truck ridge energy should be at least so
much higher than expected energy level
minRness = fscanf(fid, '%f', 1);          % ridginess should be at least this high
enRange = fscanf(fid, '%d', 1);           % allowed energy range in dB
fclose(fid);

% compute the current date and time based on the system clock:
tim = fix(clock);
if tim(6) < 10
  time3 = ['0' num2str(tim(6))];
else
  time3 = num2str(tim(6));
end
if tim(5) < 10
  time2 = ['0' num2str(tim(5))];
else
  time2 = num2str(tim(5));
end
time1 = num2str(tim(4));
timeStr = [time1 ':' time2 ':' time3];
date1 = num2str(tim(3));
date2 = num2str(tim(2));
dateStr = [date1 '-' date2];

detWhat = 0;
% find blots that are potentally part of a vehicle passage:
bb = find([Blots.centerSeg] < cSegHigh & [Blots.centerSeg] > cSegLow & [Blots.en] >=
enLevel & [Blots.en] < enLevel + enRange);
for bc = bb
  % look for corresponding engine ridges:
  if ~isempty(Ridges)
    rr1 = find([Ridges.m] >= rLow & [Ridges.m] <= rHigh & Blots(bc).left >
[Ridges.start] & Blots(bc).left < [Ridges.end]);
    rr2 = find([Ridges.m] >= rLow & [Ridges.m] <= rHigh & Blots(bc).right >
[Ridges.start] & Blots(bc).right < [Ridges.end]);
    rr3 = find([Ridges.m] >= rLow & [Ridges.m] <= rHigh & Blots(bc).center >
[Ridges.start] & Blots(bc).center < [Ridges.end]);
  else
    rr1 = [];
    rr2 = [];
    rr3 = [];
  end
  % compute nr. of pieces of ridge evidence and mean ridge energy:
  ridgeEv = ~isempty(rr1) + ~isempty(rr2) + ~isempty(rr3);
  ridgesMeanEn = 0;
  if ridgeEv > 0
    ridgesMeanEn = round(10 * log10(mean([Ridges(unique([rr1 rr2 rr3])).nEn])));
  end

  % check for false alarms:
  evFalse = (Blots(bc).h < falseH) + (Blots(bc).h < falseH2) + (Blots(bc).peakedness >
falsePness | Blots(bc).peakedness <= 0) + (Blots(bc).maxEn <= enLevel + 3) +
(Blots(bc).en <= enLevel + 3) + (ridgeEv == 1) + (ridgeEv == 0) + (Blots(bc).ridginess
< minRness) + (ridgesMeanEn > enLevel + enRange);
  if evFalse < falseLimit

    % matching: compute evidences for a scooter and a truck
    evScooter = ((Blots(bc).maxEn >= enLevel + scMaxAboveEn)+(Blots(bc).ridginess >
scRness) + (Blots(bc).centerSeg < scCseg) + (ridgesMeanEn < enLevel - scRenBelowEn &
ridgesMeanEn > ((2 / 3) * enLevel))) / 4;
```

```
    evTruck = ((Blots(bc).w > tWidth | ridgesMeanEn > enLevel + tRenAboveEn) &
Blots(bc).en > enLevel + tAboveEn);

    fid = fopen('lastDet', 'r');
    lastDet = fread(fid, 'int');
    fclose(fid);
    curDet = (tim(4) * 3600) + (tim(5) * 60) + tim(6);
    % if the last detection was at least 'sepSec' seconds ago (to avoid double
detections):
    if abs(curDet - lastDet) > sepSec
      fid = fopen('lastDet', 'w');
      fwrite(fid, curDet, 'int');
      fclose(fid);
      if evScooter > .5 & evScooter >= evTruck
        rtReportEvent('SCOOTER', Blots(bc), ridgesMeanEn, path, percFG, timeStr,
dateStr);
        detWhat = 2;
      else
        if evTruck > .5
          rtReportEvent('HEAVY VEHICLE', Blots(bc), ridgesMeanEn, path, percFG,
timeStr, dateStr);
          detWhat = 8;
        else
          rtReportEvent('LIGHT VEHICLE', Blots(bc), ridgesMeanEn, path, percFG,
timeStr, dateStr);
          detWhat = 4;
        end
      end
    end
  end
end
```

H. rtReportEvent.m

```
function reportEvent(eventType, Blot, ridgesMeanEn, path, percFG, tim, dat)

% Purpose: writing the system classification to file along with the most
% important characteristics.
% Input: the classification, the blot, the mean ridge energy, the system path
% where the log file can be found, the percentage of the cochleogram that is
% foreground, time and date strings.

fid = fopen([path 'logFile' dat], 'a');
fprintf(fid, '* %s %s', tim, eventType);
fprintf(fid, ' w %d h %d m.en %d max.en %d p %1.1f c.seg %d r %0.2f', Blot.w, Blot.h,
Blot.en, Blot.maxEn, Blot.peakedness, Blot.centerSeg, Blot.ridginess);
fprintf(fid, ' r.m.en %d fg.perc %d\n', ridgesMeanEn, round(percFG));
fclose(fid);
```