A Composite System for Real-Time Robust Whistle Recognition

Andrea Bonarini, Daniele Lavatelli, and Matteo Matteucci

Politecnico di Milano, Department of Electronics and Information,
Artificial Intelligence and Robotics Lab
{bonarini, matteucci}@elet.polimi.it

Abstract. In Robocup Middle-Size League (MSL) the challenge to recognize signals given by the referee by whistling has been introduced from this year as a way to reduce the interaction via radio-link. We present Whistle Recognizer (WR), a system able to recognize different whistling patterns, after a relatively short training done in advance. This composite system encompasses neural networks and more traditional information processing techniques. It demonstrated to be quite effective and can be easily integrated in a multi-thread control architecture, as the vast majority of those used in the league; thus, it candidates itself as a potential off-the-shelf module to be used by MSL teams not interested in research about signal processing and analysis.

1 Introduction

In Robocup Middle-Size League (MSL) the challenge to recognize signals given by the referee by whistling has been introduced from this year as a way to reduce the interaction via radio-link. Whistling has been introduced as a signal to be detected at the end of a half: detection is not compulsory, but the teams able to recognize this event will gain points for the challenge competition. Since many Robocup teams are not directly interested in signal processing and whistle analysis, an off-the-shelf package for this task would be clearly useful.

Whistle recognition has been faced in different domains to automatically detect interesting events or summarize multimedia data in sport events [1].

In this paper, we present Whistle Recognizer (WR), a composite system able to recognize different whistling patterns, after a relatively short training done in advance. This composite system (i.e., a system encompassing neural networks and more traditional information processing techniques [2]) is quite effective and can be easily integrated in any thread-based control architecture, as an off-the-shelf whistle sensor.

The tool we describe in the following sections recognizes whistle events and patterns from a raw audio data stream. With the term "pattern" we refer to a sequence of a predefined number of whistles at the rate of 1-2 whistles per second; we consider this as a possible future extension to the simple start/stop bit of information provided by a single whistle. A typical use of WR is to gather the raw audio stream using a microphone placed on the robot body and a cheap

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 130–141, 2006.

sound-card; output of the system is a message sent to the controlling process to communicate the whistle type: short single, long single, or multiple, with multiplicity.

Since whistles and referees are different from game to game, a key issue for a whistle recognition system is fast tuning and adaptation in order to reduce set-up time and increase robustness. This is obtained by implementing the recognition system using a composite approach to first extract the characteristic features from the signal, and then applying a neural classifier to detect whistling events. The system is integrated with a learning tool used to set recognition parameters basing on a short recorded sample of whistle and background noise. Section 2 gives a detailed description of the system architecture and algorithms; all design choices are explained there, while the following section gives a summary of the learning tool.

2 System Architecture

As introduced in the previous section, the system is based on a classical digital signal processing algorithm to extract signal features followed by a neural stage and an event counter. Figure 1 shows the schema of this architecture. The signal is acquired by a commercial sound-card (section 2.1) connected to a microphone; the periodogram is computed from the raw signal by a fast algorithm to extract features related to the spectral power of the whistle. To improve classification capabilities, a frequency mask has been introduced to give only the interesting samples in input to the neural stage (section 2.3). Finally, a non-linear perceptron (section 2.4) recognizes the presence of the whistle signal and passes this information to an output event counter, which produces the recognition message.

2.1 Data Acquisition and Feature Extraction

Raw data is acquired from the computer sound-card and signal level is adjusted by acting on the pre-amplifier through the IGAIN feature. Considering a whistle power spectrum concentrated below the 4 Khz we used a sampling frequency of 8 KHz and a quantization of 8 bit/sample proved to be adequate for the job. The

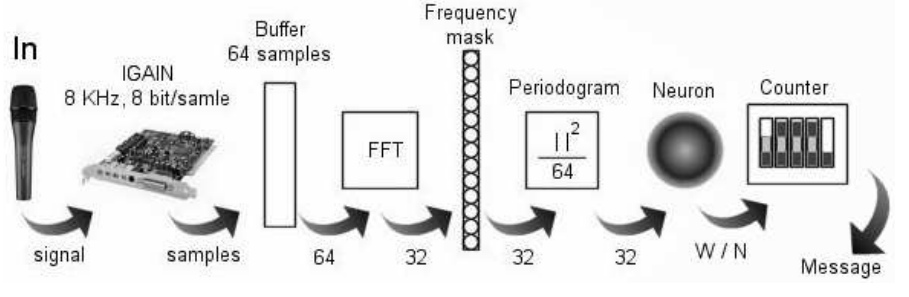

Fig. 1. System architecture

elemental block on which the system operates to detect a whistle is a 64-sample sequence, yielding a time resolution of 8 ms.

2.2 The FFT and the Periodogram

The features used by the neural classifier are taken from the signal periodogram and, as it can be expected, this computation is the system bottleneck; so, we have designer it with special attention to to make it as fast as possible. Let us consider the Discrete Fourier Transform (DFT) formula [3, 4]:

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{-nk} \tag{1}$$

where N is the number of samples, n is the time index (from 0 to $N-1$), k is the frequency index (from 0 to $N-1$) ,$W_N = e^{-j\frac{2\pi}{N}}$ and X_k is an N-sample sequence representing the original sequence x_k in the frequency domain. In general, the so-called Fast Fourier Transform (FFT) algorithm is used instead of the basic formula, to lower the N^2 complexity to $N \log N$. This is done by subdividing the original sequence in two sub-sequences, x_{2n} and x_{2n+1}, and then applying the algorithm recursively on them. In the particular case of N power of 4, the original sequence can be directly subdivided in 4 sub-sequences, so obtaining another 25% improvement ($0.75N \log N$). In this case the situation is the following:

$$\begin{bmatrix} X_k \\ X_{k+\frac{N}{4}} \\ X_{k+2\frac{N}{4}} \\ X_{k+3\frac{N}{4}} \end{bmatrix} = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -j & -1 & +j \\ +1 & -1 & +1 & -1 \\ +1 & +j & -1 & -j \end{bmatrix} \begin{bmatrix} G_k \\ W_N^{-k} H_k \\ W_N^{-2k} L_k \\ W_N^{-3k} M_k \end{bmatrix} = \underline{W} \begin{bmatrix} G_k \\ W_N^{-k} H_k \\ W_N^{-2k} L_k \\ W_N^{-3k} M_k \end{bmatrix} \tag{2}$$

where G_k,H_k,L_k and M_k are the following sequences:

$$G_k = \sum_{n=0}^{\frac{N}{4}-1} x_{4n} W_{\frac{N}{4}}^{-nk} \qquad H_k = \sum_{n=0}^{\frac{N}{4}-1} x_{4n+1} W_{\frac{N}{4}}^{-nk}$$

$$L_k = \sum_{n=0}^{\frac{N}{4}-1} x_{4n+2} W_{\frac{N}{4}}^{-nk} \qquad M_k = \sum_{n=0}^{\frac{N}{4}-1} x_{4n+3} W_{\frac{N}{4}}^{-nk}$$

Let us point out that the computation of the $\underline{W}$ matrix does not need any multiplication, so, given G_k,H_k,L_k and M_k, the X_k sequence can be computed performing only 3 complex multiplications ($W_N^{-k}*$ and so on), while the standard 2-subsequences FFT algorithm will have needed 4 complex multiplications. In doing so, we obtain a 25% gain on the total number of multiplications and this is the most important reason to select a 64-samples sequence.

The FFT of the 64-samples sequence can be easily computed with 3 stages of recursion. In fact, the 64-samples sequence is subdivided in 4 16-samples subsequences; each 16-samples sequence is subdivided in 4 4-samples subsequences, whose FFTs are then atomically calculated.

To drastically reduce the average computation time, another property of the DFT can be used: symmetry. As x_n (the elemental 64-samples sequence) is a sequence of real numbers, we can build a 64-samples sequence of complex numbers in the following way:

$$s_n = x_n + jy_n \tag{3}$$

After the calculation of the FFT of s_n, S_k, the X_k and Y_k sequences can be rebuilt using only sums and subtractions, by the following formulas:

$$\Re(X_k) = \frac{1}{2}[\Re(S_k) + \Re(S_{-k})], \quad \Im(X_k) = \frac{1}{2}[\Im(S_k) - \Im(S_{-k})] \tag{4}$$

$$\Re(Y_k) = \frac{1}{2}[\Im(S_k) + \Im(S_{-k})], \quad \Im(X_k) = \frac{1}{2}[\Re(S_{-k}) - \Re(S_k)] \tag{5}$$

This reduces by 50% the average number of multiplications, as we can get two DFTs by applying a single FFT, and, once the FFT has been calculated, the periodogram is given by:

$$P_k = \frac{|X_k|^2}{64}, \quad with \;\; k = 1, \ldots, 32. \tag{6}$$

2.3 Frequency Mask for Feature Selection

When the periodogram is computed, many of its samples represent frequencies which should not be used for recognition, as they are far from the whistle power spectrum and do not contain useful information. All these frequencies act as noise for the classifier and should be reduced to improve classification performances. So, the problem is: how many and which samples should be considered?

Since the whistle signal has a bandwidth of about 30÷40 Hz centered around its average frequency and the periodogram has a resolution of $\frac{f_{Nyquist}}{32} = \frac{4KHz}{32} = 125Hz$, a window of 3 samples can be adequate. A pure whistle signal can, in fact, cause no more than 2 samples in the periodogram to raise, in absence of background noise.

However, listener and source can move, and we need to face also the Doppler effect. Suppose a stationary source is generating sound waves with frequency $\overline{f}$ and wavelength $\overline{l} = v/\overline{f}$, being v the speed of sound. A stationary observer at a certain distance from the source will hear a sound with pitch $\overline{f}$. $\overline{f}$ times each second the observer sensor will be pushed in and pulled out as pressure crest and pressure trough reach it. The time period between two consecutive crests is $T = 1/\overline{f}$. Assume the observer in this case is a robot and starts driving away from the source. Assume that at time t_1 a pressure crest reaches the "robot ear" at position x. The next crest will be at position x at time $t_1 + T$, but the "ear" will no longer be there. In this case the crest has to travel an extra distance before it reaches the observer and this takes an extra time interval Δt. The time interval between subsequent crests reaching the ear of the observer is now $T' = T + \Delta t$.

While the observer has traveled a distance $\Delta x = v_o \cdot (T + \Delta t)$, at speed v_o, the wave has traveled a distance $\Delta x + \overline{l} = v \cdot (T + \Delta t)$. Therefore, using $\overline{l} = v/\overline{f} = v \cdot T$, we have $v_o \cdot T + v_o \cdot \Delta t + v \cdot T = v \cdot T + v \cdot \Delta t$, or $\Delta t = v_o \cdot T/(v - v_o)$. Thus, we obtain:

$$T' = T + v_o \cdot T/(v - v_o) = v \cdot T/(v - v_o), \tag{7}$$

$$f' = \overline{f}(v - v_o)/v. \tag{8}$$

The period has increased, the apparent frequency of the wave has decreased, the pitch has decreased. If the observer is driving towards the source, then the time interval between successive crests reaching the sensor will be shorter than T. The apparent frequency of the sound wave reaching the observer is thus

$$f' = \overline{f}(v + v_o)/v. \tag{9}$$

The perceived pitch of a sound wave also changes if the observer is stationary and the source is moving. Then the apparent frequency of the sound wave reaching the observer when the source is moving towards him with speed $v_s$is:

$$f'' = \overline{f}v/(v - v_s) \tag{10}$$

Whenever the source and the observer move with respect to each other, the wavelength of the sound reaching the ear will be Doppler shifted:

$$f = f''(v \pm v_o)/v = \overline{f}v/(v \mp v_s) \cdot (v \pm v_o)/v = \overline{f}(v \pm v_o)/(v \mp v_s). \tag{11}$$

The Doppler effect, computed with reasonable parameters (i.e., robot speed = 5 m/s, whistle speed = 2 m/s, as worst case hypothesis), gives a frequency shift of about 60 Hz at whistle average frequency (about 3 KHz). For this reason, the observation window can be optionally extended from 3 to 5 samples to compensate the Doppler effect.

The next problem to face is now: which frequencies should be considered? Different whistles (or different environments) will lead to different choices of frequencies and this calls for an adaptive system to reduce set-up time. To face this issue, the concept of data separation has been introduced. Given a few examples of whistle signal and background noise, these can be used to determine which are the frequencies (i.e., the periodogram samples) that better characterize the whistle sound and make it possible to distinguish it from the background noise. Data separation gives a measure of how many samples can be correctly recognized with a fixed threshold by observing a periodogram sample.

Figure 2, on the left, shows the situation for samples 21, 22, 23, and 24 of the periodogram. Whistle examples (light gray) are obtained by transforming sequences marked as "whistle" by a supervisor; the same applies to noise (dark gray) examples. For each sample, all the examples are classified in two sets: W(histle) and N(oise); so, for a generic sample, we have the situation shown in Figure 2, on the right. Data separation (DS) for the sample k is defined as:

$$DS_k \triangleq 1 - \frac{W_k \cap N_k}{W_k \cup N_k} \tag{12}$$

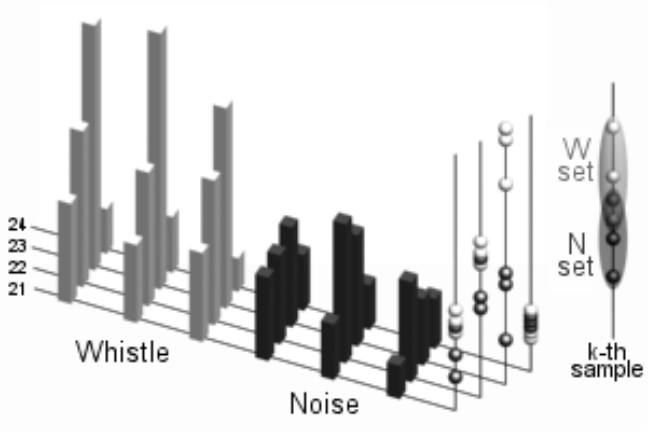

Fig. 2. Data separation

Once DS has been calculated for all 32 periodogram samples, the 3 samples with the highest DS values are selected and marked as "1" in the frequency mask. All other samples are marked as "0" in the frequency mask.

2.4 Perceptron with Hysteresis and Event Counter

The inputs used in the neural classifier are 3 (or 5, if Doppler compensation is enabled) samples of the periodogram. Being reasonable to consider low values of these samples as belonging to noise and high values to whistle signal, we can use a hyper-ellipsoid as separation surface for the perceptron. When the 3 (or 5) inputs identify a point internal to the hyper-ellipsoid, the sample will be recognized as noise, otherwise it will be recognized as whistle.

To avoid spurious commutations in noise-to-whistle and whistle-to-noise transients, we have introduced hysteresis, i.e., we implemented a recurrent perceptron that uses its previous state as switching condition. Figure 3 shows the non-linear perceptron with hysteresis used in the system. Calling s_{01} the noise-to-whistle

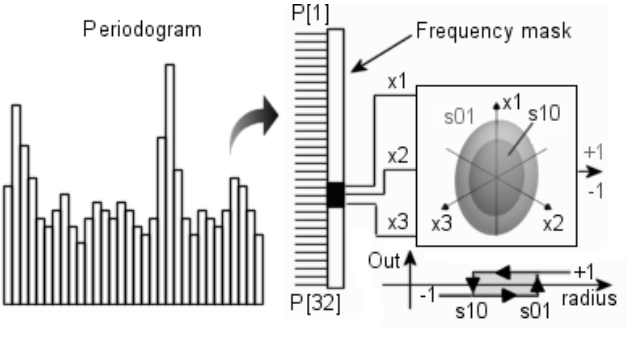

Fig. 3. Perceptron with hysteresis

(turn-on) threshold and s_{10} the whistle-to-noise (turn-off) threshold, the output of the neuron is:

$$Out = \begin{cases} if \;\; PreviousState = -1 \begin{cases} +1 & if \;\; \sum_{n=0}^{32} (\frac{x_n}{W_n})^2 > 10^{\frac{s_{01}}{10}} \\ -1 & Otherwise \end{cases} \\ if \;\; PreviousState = +1 \begin{cases} -1 & if \;\; \sum_{n=0}^{32} (\frac{x_n}{W_n})^2 \leq 10^{\frac{s_{10}}{10}} \\ +1 & Otherwise \end{cases} \end{cases} \tag{13}$$

where +1 is the output chosen to denote whistle sound detection, and -1 is the output corresponding to "noise", s_{01} and s_{10} are expressed in dB and $PreviousState$ is the previous perceptron's output.

During the learning procedure, thresholds s_{01} and s_{10} are forced to 0 and the best separation surface can be found by applying the learning rule:

$$\begin{aligned} E &= \tfrac{1}{2}(target - output) \\ W_k &= W_k - \gamma \cdot x_k \cdot E \end{aligned} \tag{14}$$

where γ is an appropriate learning rate, *target* is the desired output, as specified by the supervisor, *output* is the output of the perceptron to the current sample, and E is the error, which can be +1, 0 or -1. For instance, if the perceptron outputs +1 (whistle) while no whistles are being blown, we have:

- $outupt = +1$ (The perceptron's output)
- $target = -1$ (The correct output, provided by the supervisor).

This results in error E to be -1; this means that a noise point, which should be internal to the ellipsoid, has been actually classified as external. Since W_k is the ellipsoid radius on the k axis, it must be raised to let the ellipsoid include the example, in order to classify it as a noise sample. This is done for those axes of the ellipsoid (from W_1 to W_{32}) that are not filtered out by the frequency mask.

Doing the dual reasoning when error is +1, and combining the two cases, we obtain the learning rule 14.

The perceptron output refers to the analysis of 64 samples (8 ms) of data, and states whether this block of audio signal contains a whistle sound. Time elapsed between two consequent readings of elementary audio blocks is customizable, and it will be called "timestep" in the following. The perceptron is thus designed to produce an "event" (+1:whistle or -1:noise) every timestep; more precisely, the FFT is called once on a pair of 64-samples sequences, so we obtain a pair of events every 2·timestep.

A "counter" module of WR has to translate the sequences of events into one of the following messages:

- Short whistle
- Long whistle
- Multiple whistle, which multiplicity is n

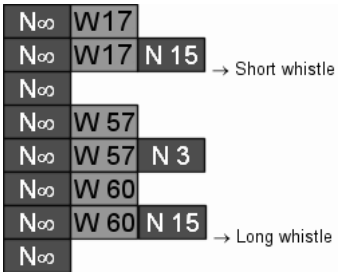

Fig. 4. A short and a long whistles

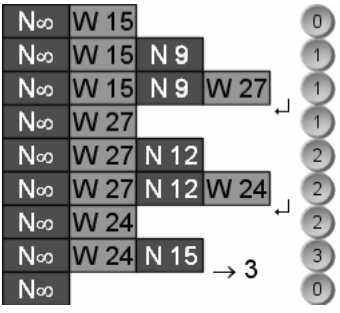

Fig. 5. A triple whistle

Table 1. Settings used in the examples

Parameter	*Value*
$ShortIfLess$	30 steps
$LongIfGreater$	50 steps
$IntervalBetween$	15 steps
$MinWhistleLength$	5 steps
$MinNoiseLength$	5 steps

The following parameters, expressed in numbers of timesteps, have to be provided by the user in order to define whistle timing specifics:

- Maximum length of a short whistle ($ShortIfLess$)
- Minimum length of a long whistle ($LongIfGreater$)
- Maximum interval length between two consecutive whistles ($Interval\ Between$)
- Minimum whistle length (shorter whistles are filtered)
- Minimum noise length (shorter noises are filtered)

Events output by the perceptron (Whistle or Noise) are accumulated into slots, which are groups of subsequent events of the same type. Slots are represented with colored boxes in figures 4 and 5. The generated slots are accumulated into an event stack, as shown in figures 4 and 5: each line corresponds to the status of the stack up to the occurrence of an event of different type.

The starting slot is "noise" from a sufficiently long time (N_∞). So, for instance, in figure 4, we have on the first line 17 "whistle" events (represented by the W17 block). Then, on the second line, we have the situation of the stack once the subsequent 15 "noise" events (N15) have been detected. Due to the settings for this example, the "W17" slot is not filtered out (17 is greater than 15, the minimum whistle length parameter), so the subsequent noise events are accumulated up to the maximum length of an interval between two whistles in a multiple whistle (in this example, 15), as defined by the user (see settings in Table 1). The stack situation shown on the second line is thus recognized as a single whistle, and the corresponding notification message is issued. Then we have the

recognition of a long whistle. Notice that the number of noise events detected (3) is less that the minimum required to interrupt the sequence of whistle events, so the N3 sequence is filtered out.

Figure 5 shows a triple whistle; the number on the right is the whistle multiplicity. Every time a new noise event occurs and the current whistle has not to be filtered, multiplicity is incremented; in order to save space in the stack, each pair of whistle-noise slots is eliminated when a new whistle slot is completed (see the 3rd and 4th row in figure 5: "W15" and "N9" slots are eliminated and the "W27" slot becomes the second slot); the system knows how many whistles were blown in the past, memorized as multiplicity. As with the previous example, the correct event is notified after a "N15" slot; here, the message is the multiplicity of the whistle.

3 The Learning Tool

A software tool is provided to let the user set system parameters and neuron weights. The higher part in the tool main window (Figure 6) is a sound recorder with two buffers: one for whistle recording and one for background noise

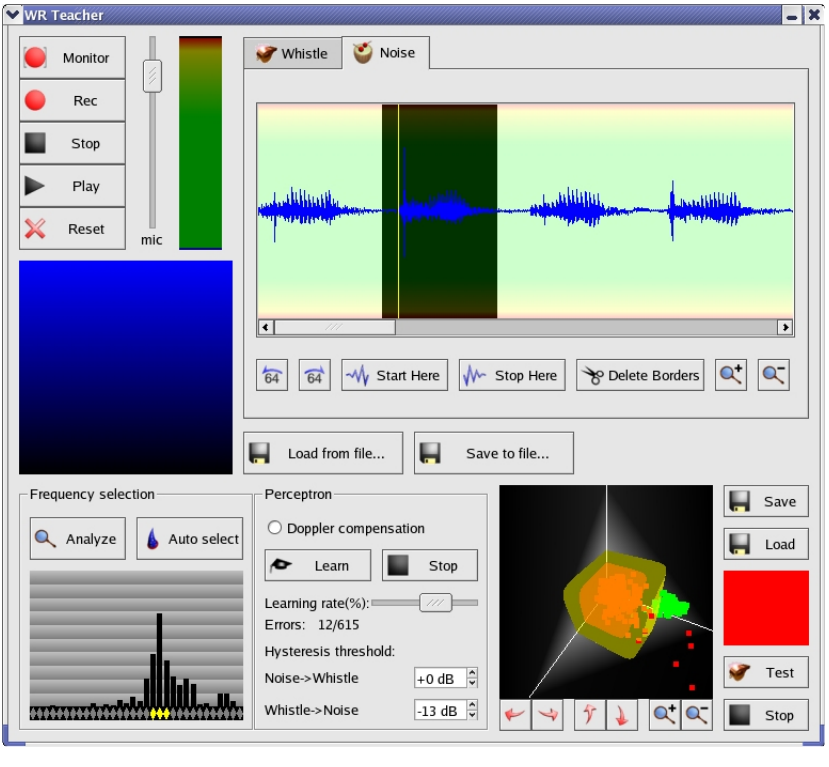

Fig. 6. The learning tool

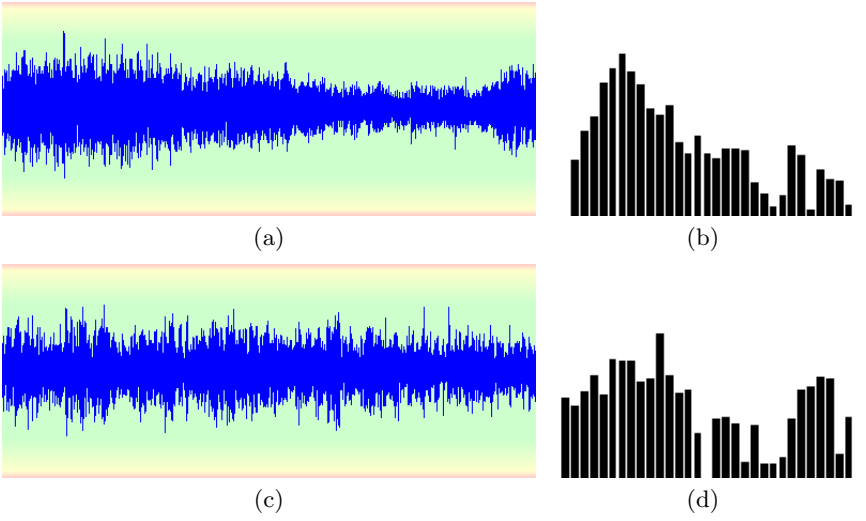

Fig. 7. Background noise (a) and whistle (c) signals captured from outside the field during a match. In (b) and (d) respective periodograms are reported.

recording. This separation is a simple way to let the user be the supervisor of the acquisition process.

The lower part of the window integrates a set of tools to control and test the perceptron. We can see a plot of data separation (on the left), which helps the user in the selection of the frequency mask, a section dedicated to the perceptron learning with a 3D visualization of the separation surfaces (in the center), and a test section (on the right).

4 Results

The system has been tested both with samples recorded from outside the field during competitions and on the robot during execution in our laboratory in Milano–Bovisa[1]. In Figure 7(a) and (c) you can notice respectively the noise and whistle signals captured during a match in Padova during the 2003 Robocup competition. In this case, we do not have a pure whistle signal so we trained the recognition system using background noise and a quite noisy whistle. Even in these conditions, the system has been able to recognize perfectly the two short whistles and the long whistle present in the sample.

Whenever we are interested on on-board whistle recognition, we should notice that the highest level of background noise is caused by the robot motors and body, which strongly vibrates while the robot is moving. This is clear in Figure 8(a) where noise has been captured on-board during robot operation.

[1] All samples used for experimental validation in this paper are available from `http://robocup.elet.polimi.it/MRT/WR.html`.

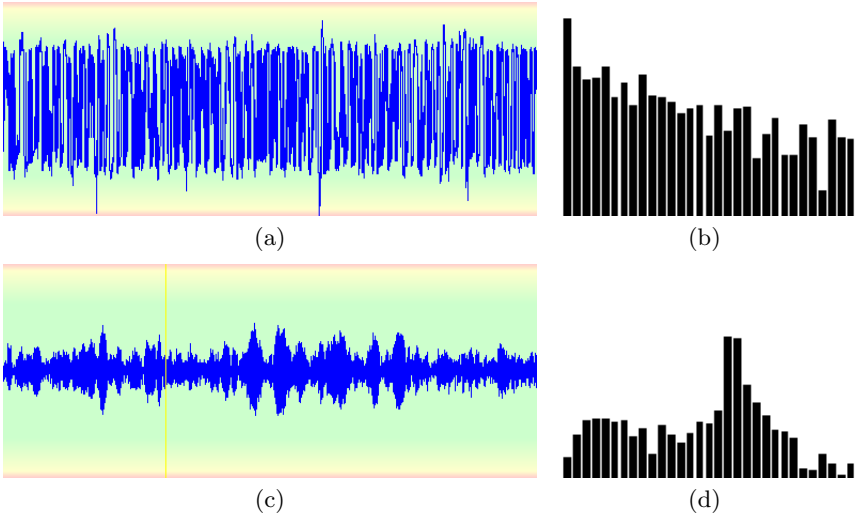

Fig. 8. Background noise (a) during execution and pure whistle (c) signals captured from the robot. In (b) and (d) respective periodograms are reported.

This results in a very strong white noise[2], which could be partially lowered by mechanically isolating the microphone from the structure. In this case, we could acquire the pure whistle signal, reported in Figure 8(c), but it has an intensity lower than noise. If the whistle-to-noise and noise-to-whistle thresholds are adequately set, the system proved to have a good behavior in rejecting noise in this especially adverse situation, resulting in a 70% correct classification rate.

Eventual shocks of the robot during the match are well filtered by adequately setting the whistle and noise minimum lengths. Human voice frequencies have been measured and they are quite distant from the whistle characteristic frequency, so this kind of background noise did not cause any problem. We also tested the system using a sample with a spurious whistle coming from a different field and it was able to reject it.

5 Conclusions

Aim of this paper was to present the design and features of WR, a system enabling robots to detect the sound of the referee whistle during a match. A composite approach, based on a spectrum analyzer followed by a neural output stage and a counter was chosen to achieve the goal providing a flexible and easily tunable system. Particular care has been taken to design a system as simple and fast as possible. We have also implemented a software tool to support the user to tune the system. During the tests, the system proved to be fast and accurate, even in presence of quite strong background noise.

[2] Actually the recorded signal resemble more to a pink noise due to the low-pass effect of the microphone.

Another important feature to be mentioned is that the system requires a very low CPU load; in fact, the whole process (FFT, perceptron and counter) takes about 150 μsec to complete on a P4 2.1 MHz, while it is called, with the default timestep, every 40 ms.

Acknowledgments

This work has been partially supported by MADSys, P.R.I.N. 2003 project "Software technologies and knowledge models for the design, implementation and experimentation of robotic multi-agent systems in real environments", co-funded by the Italian Ministry of Instruction, University, and Research.

References

1. Tjondronegoro, D., Chen, Y.P.P., Pham, B.: Integrating highlights for more complete sports video summarization. IEEE MultiMedia (2004) 22–37
2. Alippi, C., D'Angelo, G., Matteucci, M., Pasquettaz, G., Piuri, V., Scotti, F.: Composite techniques for quality analysis in automotive laser welding. In: Proceedings International Symposium on Computational Intelligence for Measurement Systems and Applications, Lugano, Switzerland (2003)
3. Oppenheim, A.V., Schafer, R.W.: Digital Signal Processing. Prentice Hall, Englewood Cliffs, NJ, USA (1975)
4. Oppenheim, A.V., Schafer, R.W.: Discrete-Time Signal Processing. Prentice Hall Signal Processing Series. Prentice Hall, Englewood Cliffs, NJ, USA (1989)