Diorama - An Open Source Digital Radio Mondiale (DRM) Receiver using MATLAB

Torsten Schorr, Andreas Dittrich, Wolfgang Sauer-Greff, Ralph Urbansky
University of Kaiserslautern, Institute of Telecommunications
P.O.B. 3049, D-67653 Kaiserslautern, Germany ({schorr|dittrich|sauer|urbansky}@eit.uni-kl.de)

Abstract: *Since the terrestrial digital sound and data broadcasting system Digital Radio Mondiale (DRM) utilizes a channel bandwidth up to 10 kHz only, a DRM software can be implemented on a conventional personal computer (PC) with a sound card. This paper presents a real-time MATLAB based open source software radio called Diorama. In addition to offering many features for educational and research purposes, Diorama includes synchronization for sampling rate adjustment, frequency offset and channel estimation based on all pilots and 2D-Wiener filtering as well as soft-input multi-stage decoding.*

1 Introduction

A digital sound and data broadcasting system for long-, medium- and short-wave radio, complying with the frequeny and bandwidth allocations of the existing analog AM radio, was proposed by the DRM consortium and recently published by the European Telecommunications Standards Institute ETSI [1]. In long terms, it intends to replace the current analog broadcast transmission in frequency band below 30 MHz.

Because of the high efficiency of coded Orthogonal Frequency Division Multiplexing (OFDM)-based transmission schemes and modern source compression techniques an adequate audio quality is achieved utilizing a bandwidth of only about 10 kHz. Considering that a common multimedia Personal Computer (PC) is able to capture audio signals with sampling frequencies of 48 kHz, it is possible to implement a complete DRM receiver in software. Besides the existing PC based software radios of Fraunhofer IIS [11] and TU Darmstadt [2] the open source software radio *Diorama* from the Institute of Communications at the University of Kaiserslautern is available since March 2005 [3].

As the origin of the word "Diorama" indicates, (Greek "dia" for "through" and "horan" for "look") it was primarily developed for educational purposes and allows to "look through" the surface of a software DRM receiver at each time and makes it possible to discover how it is working in detail. Therefor it is written for MATLAB, the popular mathematical high-level language and interactive environment [10]. Only a small portion of algorithms is implemented in the programming language C in order to speed up computationally expensive parts, e.g. bit level operations in the source decoder. Most signal variables which will be of interest are still accessible in the workspace at every time. With this approach at PCs with a CPU speed greater than 800 MHz a real-time and continuous radio operation is possible.

The paper is organized as follows. In Section II, an overview of the DRM system is given, and in Section III the concepts and characteristics of a PC based software receiver are discussed. Then, in Section IV implementation aspects of *Diorama* are described starting with the demodulation of the signals from the intermediate frequency position to the baseband, followed by the processing steps synchronization, equalization, channel decoding, source decoding and multimedia output. The paper ends with concluding remarks.

2 DRM System Overview

2.1 Overview

On the one hand, DRM intends to meet the bandwidth requirements of the existing AM broadcasting system, i.e., 9 kHz and 10 kHz for long- / medium-wave and short-wave, respectively. On the other hand, fading due to multipath propagation and doppler have to be mitigated and single frequency networks have to be enabled. Therefore, DRM applies coded OFDM with scattered pilots, unequal error protection, Multi Level Coding (MLC) and Hierarchical Mapping (HM) to admit an audio data rate up to 24 kbit/s [1]. Moreover, 4 Robustness Modes and 4 protection levels allow for different rates. Extended data rates can be achieved by twice the AM bandwidth, i.e. 18 kHz and 20 kHz, respectively.

To deliver an acceptable audio quality at these low data rates, the DRM system contains Advanced Audio Coding (AAC) for general audio signals and Code Excited Linear Prediction (CELP) and Harmonic Vector eXcitation Coding (HVXC) for speech signals [1]. Additionally, data for Spectral Band Replication (SBR) can be transmitted to reconstruct high frequency parts of the source signal and enhance the perceptual audio quality. DRM also includes a Parametric Stereo (PS) coder.

In DRM three data streams are multiplexed: the Main Service Channel (MSC), the Fast Access Channel (FAC) and the Service Description Channel (SDC). The MSC contains the data of up to 4 services, either audio or data services. Whereas the FAC provides service selection information for fast scanning and necessary parameters to start multiplex decoding, the SDC delivers information for complete MSC decoding, attributes of the services and alternative sources for the same data.

2.2 OFDM

OFDM distributes the information to be transmitted on K subcarriers equally spaced at $\Delta f = 1/T_u$, where each subcarrier of an OFDM symbol $x(t)$ is subject to 16- or 64-Quadrature Amplitude Modulation (QAM) by a complex valued symbol c_k. Any OFDM symbol with an OFDM symbol duration of $T_s = T_u + T_g$, where T_u is the usefull symbol duration and T_g defines the length of the guard interval, can be expressed as

$$x(t) = \sum_{k=K_{min}}^{K_{max}} c_k \cdot \exp(j2\pi kt/T_u), \qquad (1)$$

for $0 \le t < T_s$, which finally has to be transposed to the desired RF transmission frequency by a conventional mixer. Note that subcarrier $k = 0$ will not be used. The guard interval can be considered as a cyclic extention of the useful symbol interval and helps to avoid intercarrier and intersymbol interference due to transmission channels with memory, as long as the length of the overall channel impulse response is below T_g.

A DRM transmission frame consists of N_s OFDM symbols, where some cells are modulated with known fixed phases and amplitudes. These pilot cells will be used for frequency and time synchronisation as well as for channel estimation to equalize the channel transfer characteristics.

With respect to different wave propagation scenarios, 4 Robustness Modes are defined, see Tab. 1:

Mode A for long- and medium-wave local transmission mainly by ground waves without fading,

Mode B for short-wave intracontinental transmission mainly with fading due to interference of the ground wave and a single hop ionospheric wave.

Mode C for short-wave intercontinental transmission mainly with fading due to interference of multipath propagation by multi hop ionospheric waves.

Mode D especially for Near Vertical Incidence Skywave transmission, frequently used in tropic regions and resulting in fading and doppler.

Mode	A	B	C	D
OFDM Symbol Duration T_s /ms	26.66	26.66	20	16.66
Guard Interval Duration T_G /ms	2.66	5.33	5.33	5.33
Carrier Spacing Δf /Hz	41.66	46.88	68.18	107.14
Number of Carriers @ 9kHz K	204	182	---	---
Number of Carriers @ 10kHz K	228	206	138	88
Number of Carriers @ 18kHz K	412	366	---	---
Number of Carriers @ 20kHz K	460	410	280	178
OFDM Symbols per Frame N_s	15	15	20	24
MSC Net Data Rate in kbit/s	6.2 -71.9	4.8 - 56.1	9.1 - 45.4	6.0 - 30.6

Tab. 1 Robustness Modes of DRM.

2.3 Channel Coding

DRM is designed to be flexibly adaptable to different channel constraints and propagation conditions. Additionally, several logical channels are provided with different error protection needs. To achieve the 4 different protection levels specified in DRM, a code rate flexible MLC scheme with different mapping strategies is used [1]. Service providers can choose between Equal Error Protection (EEP) and Unequal Error Protection (UEP), combined with Standard Mapping (SM) or Hierarchical Mapping (HM) for data needing very strong protection.

To simplify the receiver structure all necessary code rates from 1/4 to 8/9 are achieved by puncturing of the same convolutional mother code with rate 1/4. The encoder is depicted in Fig. 1. An energy dispersal scrambling is applied before encoding to avoid periodic bit patterns.

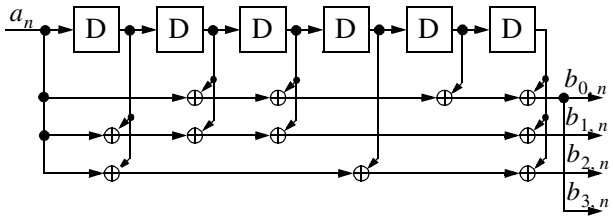

Fig. 1. Convolutional Encoder [1]

The MLC scheme divides the QAM mapping into real and imaginary parts, for which different partitioning strategies can be applied. This allows to use standard partitioning (Ungerboeck set partitioning), hierarchical partitioning (block partitioning) or a mixture of both. The different bit levels of the QAM mapping are considered as single binary channels with certain individual channel capacities related to the error probabilities, for which the code rates of the constituent codes have to be chosen appropriately.

To avoid the influence of burst errors generated during the decoding process, which are distributed between the levels, different pseudo random bit interleavers are provided for the individual levels. Fig. 2 shows the structure of an MLC encoder for 64-QAM in the case of equal partitioning for the real and imaginary part, where equal levels of both parts are multiplexed.

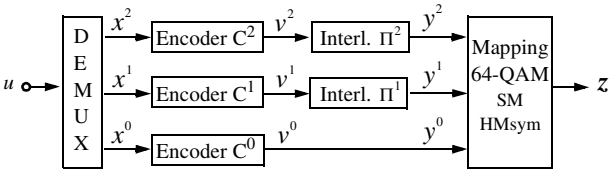

Fig. 2. MLC scheme for the MSC

For the Main Service Channel (MSC), an additional interleaving of the QAM-symbols is provided before channel transmission, consisting of a 400 ms block interleaver, optionally combined with a 2 s convolutional interleaver.

3 PC Based Software Receiver

3.1 Related Work

To our best knowledge, today there exist three fully functional real-time software radios for DRM besides the one presented here. Closely connected with the development of DRM are the "Fraunhofer Software Radio" and the "DRM Software Radio" both from Fraunhofer IIS [11]. The third software radio named "Dream" was developed at the University of Darmstadt [2] and is published as open source. It is programmed in C++ and is available for Windows and Linux operation systems.

3.2 Hardware Aspects

In order to sucessfully receive and decode DRM signals on a PC some hardware requirements have to be fulfilled.

First, a suitable frontend is necessary, which receives and downmixes the signals from the antenna to a suitable intermediate frequency, typically at 12 kHz, which then can be digitized using the soundcard of the PC. For many commercial shortwave radios there exist detailed instructions to modify the mixing circuits for a proper signal in the required frequency range. Also there are a few tuners on the market, which are able to receive and downmix DRM signals ready to use with a PC or laptop and often bundled with the Fraunhofer IIS software receiver. As required for OFDM reception, low phase noise oscillators are mandatory for the mixing operations.

Second, a proper soundcard is required, which performs sampling and analog to digital conversion (ADC) and digital to analog conversion (DAC). In general, these devices are capable to sample at rates of 44,1 kHz or 48 kHz. The latter is preferable because the sampling time then is exactly a quarter of the elementary time period T as specified by the DRM standard. This leads to a simplification of the following signal processing, since for all robustness modes the processing is done in the same way by only varying the number of audio samples per symbol and adjusting the DFT size, while maintaining the orthogonality of the subcarriers. Different source sampling rates can be used by applying sample rate conversion afterwards, which can efficiently be done using polyphase filters [8]. However, some soundcards resample internally at certain sampling rates, which is optimized for natural audio signals and degrades the DRM signal in terms of jitter or phase noise and so makes it unsuitable for further OFDM decoding.

Another aspect, which should be mentioned, is that we perform signal processing on multitasking operation systems (Windows 98/NT/XP, Linux), which are not designed for real-time applications. So, continuous operation can not be guaranteed in all situations, however in practice it works accurately on modern PCs.

4 Diorama

4.1 Matlab Environment

MATLAB is a high-level language and interactive environment that offers the capability to efficiently perform computationally intensive operations while providing easy debugging, online calculations and visualization tools. It allows to integrate compiled dynamic libraries, written in low level programming languages e.g. C or C++. We have used this mechanism, e.g., to access the soundcard by starting separate recording and playing threads, which are collecting or buffering the audio signals. Inside these program parts we have placed dynamic sample rate conversion algorithms, which are necessary for the OFDM processing. The global loop for the software radio in MATLAB is performed blockwise, where one block corresponds to one DRM frame, that is a cycle time of 400 ms.

4.2 Demodulation, Synchronization, and Equalization

The real valued audio samples from the recording thread are filtered, subsampled in order to reduce the data and demodulated to baseband for further OFDM processing. These three operations are done in a single step using the polyphase decimating structure depicted in Fig. 3.

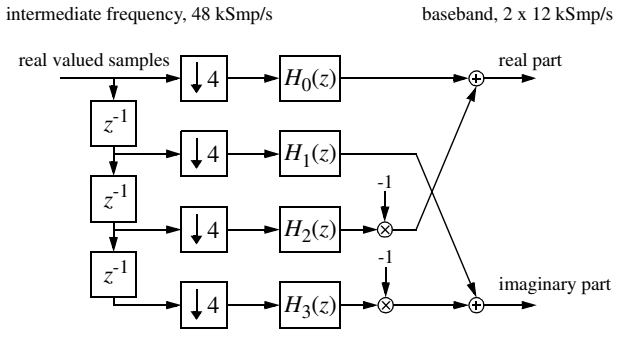

Fig. 3. Combined demodulation and subsampling

This is possible due to the fact, that we apply a frequency shift of exactly -12 kHz (or +12 kHz for a flipped spectrum), which is a quarter of the sampling rate. The finite impulse response (FIR) polyphase filter is designed to avoid aliasing and to suppress the DC part of the signal due to a non-ideal ADC, which could interfere with the useful signal

in combination with subsampling. We have found that a subfilter order of 5, that is in total 4x6 coefficients, is sufficient for an image rejection of 45 dB, considering that a filter ripple in the useful frequency range is not critical because of the succeeding equalization. At its output we have signal samples in the complex baseband taken at the elementary time period T, i.e. a data reduction by a factor of two.Using this method, it is easy to flip the spectrum (lower side band of IF signal at 12 kHz), which could be necessary for some tuners, by just changing the sign of the multiplicators in the lower two paths of the polyphase structure.

In future receiver versions, it is planed to convert the input sample rate also within this step by using polyphase resampling [8]. This will lead to an increase in the number of subfilters but not in computational expense.

The signal processing steps which follow, acquisition, time/frequency synchronization and equalization, are depicted in Fig. 4 (tracking mode only). In principle all used methods are well known for OFDM receivers [6], [7], but some aspects worth to be emphasized are described in the following.

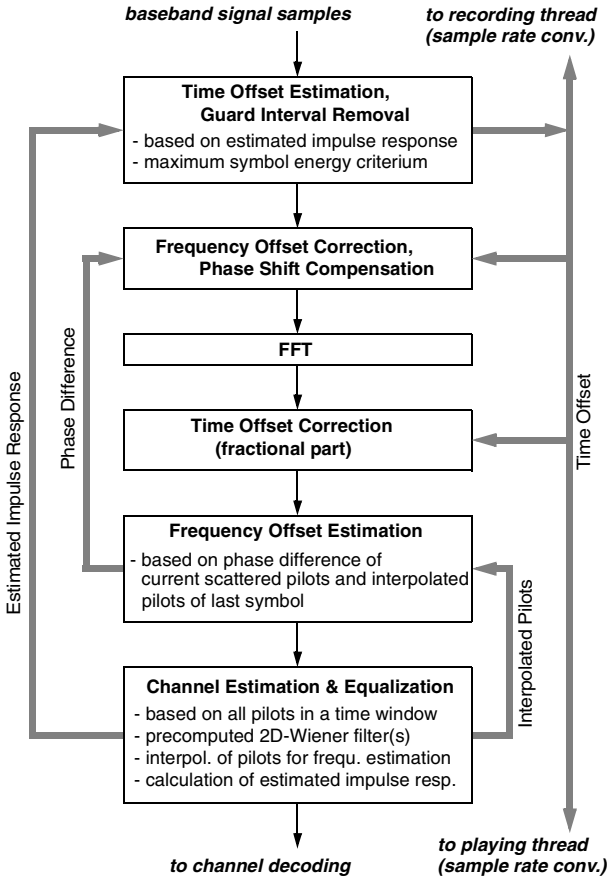

Fig. 4. Processing flow of synchronization and equalization

The software is designed to output the decoded audio stream as fast as possible. For this aim the input signal samples used for acquisition are reused in the normal tracking processing path for data decoding. Of course, we benefit only if the CPU performance is high enough to do this double job in the same time. In principle, after starting the software, every complete sampled OFDM symbol is used for data decoding and the program switching delay is only restricted by the system specifications, e.g. interleaver depth, logical frame size etc. The switching delay can be further reduced by utilizing the forward error correction capabilities in the channel decoder. This mechanism is explained in detail in the next subsection.

For channel estimation and equalization, we have implemented a real two dimensional Wiener filter as described in [5]. The coefficients are precomputed and the filter operation is applied using matrix multiplication in a polyphase way. I.e., each scattered pilot constellation relative in time position to the OFDM symbol, which has to be equalized, corresponds to a separate precomputed Wiener filter matrix. Using this approach, non separable equalization filters can be investigated.

In Fig. 5 the applications of Wiener filtering is illustrated. Each line represents an OFDM symbol, where the filled circles are pilots cells.

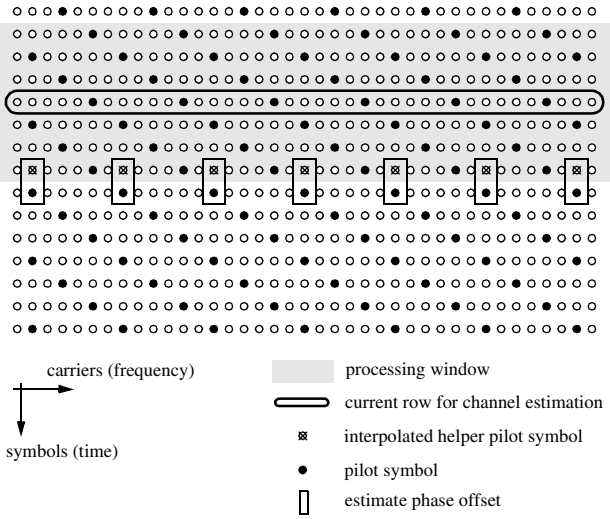

Fig. 5. Two dimensional Wiener filtering for channel estimation and frequency synchronization

All pilot cells inside a certain window (grey rectangle) are phase and gain corrected according to its nominal value as specified in the DRM standard and arranged in the vector $\mathbf{p}_l$, which is used as input to the Wiener filter matrix $[\mathbf{W}_{\text{eq}}]_{l \bmod F}$. The output vector $\hat{\mathbf{H}}_l$ is the channel estimate for each carrier position at time instance l:

$$\hat{\mathbf{H}}_l = [\mathbf{W}_{\text{eq}}]_{l \bmod F} \cdot \mathbf{p}_l \,. \tag{2}$$

The matrix $[\mathbf{W}_{\text{eq}}]_{l \bmod F}$ is precalculated for all different pilot constellations, utilizing robust assumptions about the channel and the noise behavior. Because the pilot positions are cyclic, the matrix can be reused every F symbols. For frequency synchronization also 2-D-Wiener filtering is utilized. For each OFDM symbol, the weighted accumulated phase differences,

$$\Delta\hat{\varphi} = \angle(\mathbf{q}_l^T \cdot \mathbf{q}_{l-1}^*) \,, \tag{3}$$

between the current scattered pilots at the time position l, $\mathbf{q}_l$ and the Wiener interpolated cells at the same carrier positions of the last symbol $\mathbf{q}_{l-1}$ are calculated,

$$\mathbf{q}_{l-1} = [\mathbf{W}_{\text{sync}}]_{l \bmod F} \cdot \mathbf{p}_l \,, \tag{4}$$

and used as input for the frequency synchronization control loop (see also Fig. 5).

Time synchronization for guard interval removal and sample rate adjustment is based on the estimated channel impulse response. Because we are not able to control the clock timing of the ADC and DAC, we have to synchronize not only the incoming signal due to OFDM orthogonality reasons but also the decoded digital audio signal. In DRM, the broadcast audio stream is locked to the OFDM symbol timing. So, in order to avoid an over- or underrun of the audio playing buffer, which is clocked by the ADC/DAC hardware, the decoded audio samples also have to pass a resampling stage before analog audio is generated. Since we assume a common clock source for the ADC and DAC of the soundcard, we can use the information of the OFDM timing estimation block to adjust output resampling.

4.3 Channel Decoding

To recover the transmitted data bit streams, the punctured convolutional codes (Fig. 1) with MLC (Fig. 2) have to be decoded. Since optimal decoding of an MLC scheme with an overall Maximum Likelihood (ML) or Maximum-A-Posteriori (MAP) decoder is infeasible because of the huge number of states, suboptimum Multi Stage Decoding (MSD) is applied at the receiver.

The constituent levels are decoded successively starting at level 0 and passing the estimated data $\hat{y}$ to the demappers of the following levels. In our case only binary information is passed between the levels. The usage of soft information is also possible but requires more complex demappers and decoders like the Soft Output Viterbi Algorithm (SOVA).

Since the constituent codes are not approaching the channel capacity we can achieve a further decoding gain by iterative decoding of the levels. Therefor we use the retrieved information of all other levels for the demapping of a certain level. Fig. 6 shows the decoder structure.

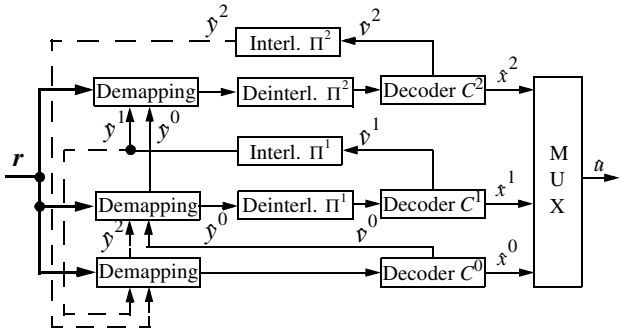

Fig. 6. Multi-Stage Decoder for the MSC

To reduce the processing load, iterating can be aborted if there is no further change in the retrieved binary information.

Because DRM is an OFDM transmission system we assume that the transmission of a single symbol can be described by a flat fading channel model with Additive White Gaussian Noise (AWGN), whereas the noise considered for the whole spectrum is not necessarily white. For the received symbol r_n we can write:

$$r_n = z_n h_n + n_n \,, \tag{5}$$

where z_n is the transmitted symbol, h_n is the complex value of the channel transfer function for the considered subcarrier and n_n is an additive noise value. Using log-likelihood ratios

$$L(\hat{y}_n^{\,j}) = \ln\left(\frac{P(\hat{y}_n^j = 1)}{P(\hat{y}_n^j = 0)}\right) \tag{6}$$

as input of the Viterbi decoder, where $\hat{y}_n^j$ is the estimated binary symbol at level j, we obtain

$$L(\hat{y}_n^{\,j}) = \frac{\hat{h}_n^2}{\sigma_n^2}\left|\frac{r_n}{\hat{h}_n} - z_n^0\right|^2 - \frac{\hat{h}_n^2}{\sigma_n^2}\left|\frac{r_n}{\hat{h}_n} - z_n^1\right|^2 , \tag{7}$$

where $\hat{h}_n$ is the estimated value of the channel transfer function, σ_n^2 is the estimated noise variance, $r_n/\hat{h}_n$ is the equalized received value and $z_n^{0,1}$ is the closest possible signal value with $\hat{y}_n^j = 0$ or $\hat{y}_n^j = 1$, respectively.

As already observed in [9], the actual probability density function of the noise signal at the individual levels differs from the Gaussian distribution because of the error propagation from one level to another. A satisfactory approach is the usage of a linear metric resulting in a better error performance:

$$L(\hat{y}_n^{\,j}) = \frac{\hat{h}_n}{\sigma_n} \cdot \left|\frac{r_n}{\hat{h}_n} - z_n^0\right| - \frac{\hat{h}_n}{\sigma_n} \cdot \left|\frac{r_n}{\hat{h}_n} - z_n^1\right| . \tag{8}$$

It is necessary to consider the noise of the channel as colored, especially because of noise introduced by certain PC audio interfaces. Therefor we have also to estimate the

noise variance σ_n^2 of the individual carriers. To calculate the noise values, an estimate $\hat{z}_n$ is determined using the output $\hat{y}_n$ of the channel decoders. The noise variance is given by the expectation value

$$\sigma_n^2 = \mathrm{E}\{\left|r_n - \hat{z}_n \hat{h}_n\right|^2\}. \quad (9)$$

In a similar way, the overall Signal to Noise Ratio (SNR) can be determined in terms of Modulation Error Ratio (MER) and Weighted Modulation Error Ratio (WMER).

Because of MSC cell interleaving a decoding delay of 400 ms or 2 s respectively is introduced. To speed up audio output, decoding is started even if not all cells r_n of a coded frame are received. The missing cells are marked as erasures by setting $\hat{h}_n = 0$ at these particular positions. At sufficiently high SNR the decoder is capable to determine the actual values of a subset of the erased positions. Since the audio source coding frame length is a fraction of the block length of the channel code, there is a quite high probability that some audio frames are retrieved without errors and audio playback can be started.

4.4 Source Decoding

In a DRM broadcast system several data and audio services can be provided. Because of the low data rates, source coding in terms of lossy and lossless compression for audio and data streams, respectively, is applied.

For audio broadcasting, DRM offers AAC for audio signals and for speech signals CELP and HVXC. SBR information can be transmitted to reconstruct high frequency parts of the source signal and PS allows to improve the performance of stereo coding. Optionally, UEP can be used to achieve graceful degradation of the audio signal at bad reception conditions.

Diorama contains an interface to the third party open source AAC decoder FAAD2 [12], including SBR and PS decoding and supporting UEP. For text decompression, e.g. in the case of web site broadcasting, *Diorama* uses the zlib library [13]. Decoding of the NewsService Journaline(R) [14] is supported, too.

4.5 Multimedia Output and User Interface

Besides the playback of decoded AAC audio streams via the PC soundcard, *Diorama* saves the files of the data services into an arbitrary directory. This way, it is possible to view the received files of a web site, slide show or NewsService Journaline(R) in background.

Diorama's control panel allows to enable or disable several decoding functions of the receiver and to open the visualization windows. It also offers the possibility to pause the execution of *Diorama* to have a look into the running decoding process. The most important parameters of the DRM signal and receiver status are shown in the signal information window. Online displaying of input spectrum, synchronization variables, channel estimation, constellations and SNR at each carrier is also available.

5 Conclusions

Diorama represents a complete MATLAB based DRM receiver capable to run in real-time on a conventional PC, with a system load comparable to other software radios. The capability of pausing the program offers the user access to MATLAB tools for debugging and visualization during the decoding process. This makes *Diorama* valuable for the development of new algorithms and for teaching.

6 References

[1] European Telecommunication Standard Institute (ETSI), Sophia Antipolis Cedex, France, *Digital Radio Mondiale (DRM); System Specification*, ETSI ES 201 980 V2.1.1, April 2004.

[2] A. Kurpiers and V. Fischer, "Open-source implementation of a digital radio mondiale (DRM) receiver", in *9th International IEE Conference on HF Radio Systems and Techniques*, Bath, United Kingdom, June 2003

[3] A. Dittrich, T. Schorr, "Diorama - Real Time DRM Receiver for Matlab", open source project homepage, http://nt.eit.uni-kl.de/forschung/diorama/, March 2005

[4] A. Dittrich, T. Schorr, R. Urbansky, "Diorama - A MATLAB Based Open Source Software Radio for Digital Radio Mondiale (DRM)", to be published at International OFDM Workshop, Hamburg, Germany, August/September 2005

[5] P. Höher, S. Kaiser, P. Robertson, "Two-Dimensional Pilot-Symbol-Aided Channel Estimation By Wiener Filtering", in *Proc. Int. Conf. Acoust., Speech and Signal Processing*, Munich, Germany, Apr. 1997, pp. 1845--1848

[6] M. Speth, S. A. Fechtel, G. Fock and H. Meyr, "Optimum Receiver Design for Wireless Broad-Band Systems Using OFDM - Part I", in *IEEE Trans. on Communications*, vol. 47, no. 11, November 1999

[7] M. Speth, S. A. Fechtel, G. Fock and H. Meyr, "Optimum Receiver Design for OFDM-Based Broadband Transmission - Part II: A Case Study", in *IEEE Trans. on Communications*, vol. 49, no. 4, April 2001

[8] J. G. Proakis and D. G. Manolakis, "Digital Signal Processing", Prentice-Hall, New Jersey, 1996

[9] V. Fischer, A. Kurpiers and F. Kulla, "Improved Multistage Decoding of Multilevel Codes for Digital Radio Mondiale (DRM)", *8th IEEE International Symposium on Consumer Electronics 2004*

[10] http://www.mathworks.com/

[11] http://www.drmrx.org/

[12] http://www.audiocoding.com/

[13] http://www.zlib.net/

[14] http://www.iis.fhg.de/dab