Chapter 4
Real-time Incremental Processing

In Chap. 2 a large set of acoustic features based on Low-level Descriptors (LLDs) and statistical functionals has been introduced and the most common modelling methods were summarised. The features and the modelling methods used in this thesis have been selected with the goal of on-line processing in mind, however, most of them are all general methods that are suitable both for on-line and off-line processing. This section deals specifically with the issues encountered in on-line processing. Instead of on-line processing, the terms real-time processing or incremental processing are sometimes used synonymously, although they are not the same. Thus, in order to avoid confusions, these terms are defined here to clarify the differences between them:

Real-time processing refers to processing of data in real-time. This implies two things: (a) the processing of the data requires less time than the duration of the data, i.e., the time it will take to record or play-back the data at normal speed, and (b) the processing lag is as low as possible, ideally zero. In other fields, such as computer operating systems, real-time also refers to scheduling requirements, e.g., that events are processed exactly at the time they occur or within a certain deadline (Ben-Ari 1990, p. 164). Here, the definition of real-time will be restricted to points (a) and (b), where (a) *must* be enforced, and for (b) the lag must have a fixed upper bound and should be as low as possible, which is roughly between 100 ms and 5–10 s for speech and music analysis tasks.

On-line processing refers to processing of a continuous live stream of data while data are recorded or transmitted, i.e., computing and returning results continuously before the end of the data stream has been seen. This is in contrast to off-line processing, where a whole segment is recorded (e.g., a speech utterance or a song) before it is processed. In on-line processing only single pass processing algorithms can be used, while off-line processing allows for multiple processing passes. Algorithms with multiple processing steps or algorithms which use context (past or future data) are not suitable for on-line processing without major modifications and, even then,

F. Eyben, *Real-time Speech and Music Classification by Large Audio Feature Space Extraction*, Springer Theses, DOI 10.1007/978-3-319-27299-3_4

they are (in most cases) not able to give the same results as in off-line processing. For on-line processing, real-time processing is implicitly required. If the processing of the incoming data is too slow, the lag between data input and output will constantly increase, soon making the system unusable. To implement on-line processing, generally data are aggregated in some (rather short) buffer and a result is generated every time the buffer is full. Depending on the application only the data in the buffer is used to produce the result, or potentially also more past data or the previous results can be used. The lag between inputs and outputs is determined by the buffer size. For affect recognition, for example, a buffer size of ≈1–4 s is suitable (e.g., Eyben et al. 2012). I.e., in the most naive on-line processing approach, audio data would be recorded for, e.g., 2 s, then features would be extracted and a classification would be performed, returning the result. The total lag of the system then is given by the buffer size (2 s) and the time required for feature extraction and classification.

Incremental processing is a more optimised implementation of on-line processing, which tries to minimise the lag between data input and analysis result output. The trick is to process incoming data as soon as possible, while waiting for new data to arrive. In the above example of affect processing, the LLDs can be computed from the signal already before the 2 second buffer of audio data is filled because for computing a LLD vector only 20–60 ms (depending on the frame size) of audio data are required. Thus, all LLDs are already computed when the 2 second audio segment has been recorded and the lag is reduced by the time required to compute the LLDs. If, for instance, the computation of the LLDs requires 0.2 s, the lag is reduced by this amount. Moreover, a preliminary result can be computed from a 1 second buffer, for example, and another, final result would be then computed from the full 2 second buffer. This reduces the lag further, but possibly at the cost of accuracy of the preliminary result.

The remaining parts of this section show important issues of on-line processing in speech and music analysis in detail, such as input segmentation issues (Sect. 4.1) and feature extraction issues (Sect. 4.2). Next, the implementation and architecture of the incremental on-line processing which has been implemented for this thesis in the open-source feature extraction toolkit openSMILE (Sect. 4.3) is described. Finally, as an example, a fully continuous and incremental approach to automatic speech emotion recognition is described in Sect. 4.4.

4.1 Segmentation Issues

The specific issues encountered with incremental processing in various speech and music analysis tasks depend on the task set-up and its requirements, especially the temporal level the analysed phenomenon occurs on. In most applications multiple hierarchical (temporal) levels of processing are used.

For speech, the most important first step is to identify segments in a continuous audio stream, which actually contain speech. This is well-known as Voice Activity

Detection (VAD) or Speech Activity Detection (SAD). A robust VAD method for neutral and emotional speech, which was developed for this thesis, is described in Sect. 5.1. It shows excellent results in high noise and reverberation levels at low latency. The low lag required is a challenge for the VAD algorithm in an incremental on-line processing framework. Some algorithms, such as simple energy based VAD, require thresholds to be tuned over one or more utterances. They cannot be used for on-line processing, for example. For more details see Sect. 5.1.

Once the speech segments and the speech pauses have been identified, segments need to be either combined (if too short) or split into sub-segments (if too long) in order to match the optimal analysis unit length. The choice of this length is a highly debated topic (cf. Schuller and Rigoll 2006; Batliner et al. 2010; Eyben et al. 2012), and the answer depends strongly on the type of data and labels. For instance, some pathological vocal chord disorders may be evident from only single vowels (cf. Parsa and Jamieson 2001), while speaker traits like age, or depression are only evident from multiple utterances and sometimes are also reflected in the linguistic content rather than only in the acoustic content. Emotion lies somewhat in between—very strong and prototypical emotions can be expressed in single words, while others are only evident in utterances which contain a handful of words (cf. Batliner et al. 2010). Finding the optimal unit of analysis is still an active area of research, especially for affect recognition (Schuller and Rigoll 2006; Schuller et al. 2007a; Busso et al. 2007; Mower and Narayanan 2011). As stated by Zeng et al. (2009), the audio segmentation is one of the most important issues for real-life use-cases, but it has been "largely unexplored so far". This thesis does not claim to solve this issue, but shall highlight various constraints that arise in the context of real-time and incremental processing, and propose a suitable segmentation algorithm for these scenarios.

4.1.1 On-Line Segmentation

For fast on-line processing, linguistic analysis units such as words, phrases, etc. are unsuitable as long as an Automatic Speech Recognition (ASR) system has to be used to perform the necessary segmentation. The delay introduced is too high for most on-line applications and the computing resources required by the ASR reduce the resources available for the paralinguistic analysis of speech, for example. Thus, for on-line systems, a low-resource, probabilistic pause based segmentation method is suggested in this thesis: A range for the length of speech units is defined, ranging from 1–10 s for affect recognition, for example. The algorithm starts in the 'pause' state. If the VAD function (a generic speech activity function returning a probability value is assumed here) crosses the voicing threshold for at least N_{pre} consecutive frames, the start of a speech segment is assumed at the first frame above the threshold. The end of the speech segment is detected when the VAD function falls below the threshold for N_{post} consecutive frames. An end is not detected if the current segment length is smaller than the minimum allowed segment length L_{min}. If the current segment length exceeds the maximum allowed segment length L_{max}, N_{post} is linearly reduced down

to 0 up to a final maximum length $L_{max,grace}$. Thus, between L_{max} and $L_{max,grace}$ splitting at short pauses becomes more and more likely, and at $L_{max,grace}$, an end of the current speech segment is always defined, even if the VAD function is above the threshold. In that case, the next segment will start at exactly the next frame, if the current segment is followed by at least N_{pre} consecutive frames above the threshold. The length of the ith speech segment detected with this method is denoted by $N_{seg}^{(i)}$ in the following.[1]

4.1.2 Incremental Segmentation

To avoid problems which occur when long utterances are force cut to a maximum length as described above and to decrease the maximum lag (which is the maximum segment size in the above method), the above on-line segmentation has been extended by an incremental segmentation algorithm (published by the author of this thesis in Eyben et al. 2012): A window length N_{win} and a window step $N_{step} \leq N_{win}$ (both in frames) are defined in addition to the minimum and maximum length parameters above. The corresponding variables, which express these lengths in seconds are denoted as L_{step} and L_{win}, respectively. Speech segments are detected with the same on-line method as above, but split into analysis sub-segments. Each sub-segment is labelled with an index pair (i, j), where i is the index of the master segment, and j is the index of the sub-segment within the master segment. The first sub-segment ($j = 0$) starts at the beginning of the ith master segment and is of maximum length N_{win} or $N_{seg}^{(i)}$, whichever is lower. The following sub-segments start at $j \cdot N_{step}$ as long as $j \cdot N_{step} < N_{seg}^{(i)}$ and have a corresponding length of N_{win} or $N_{seg}^{(i)} - j \cdot N_{step}$, whichever is lower. The proposed method was used in the SEMAINE system (Schröder et al. 2012) and for the first two Audio-visual Emotion Challenge (AVEC) baseline evaluations (Schuller et al. 2011a, 2012a). A typical choice for L_{step} is one second; for L_{win} typical values are between 2 and 5 s.

Two *variations* of this incremental approach are possible, which can be both combined arbitrarily:

1. to reduce the initial response lag, the window length N'_{win} grows incrementally up to the maximum length N_{win} with the step size N_{step}; thus, the sub-segment at $j = 0$ has the length N_{step}, at $j = 1 \ldots \lfloor \frac{N_{win}}{N_{step}} \rfloor$ the length $(j + 1)\, N_{step}$ and at $j > \frac{N_{win}}{N_{step}}$ the length N_{win},
2. and/or the maximum window length N_{win} can be set to infinity, i.e., the fixed maximum length constraint removed—the maximum sub-segment length will then correspond to the current segment length $N_{seg}^{(i)}$, i.e., the last sub-segment contains the complete ith segment.

[1] In openSMILE this behaviour is implemented in the `cTurnDetector` component.

With the above approach the segment length can be controlled, however, it remains variable within the specified bounds and excludes long speech pause segments. For some applications a constant segment length might be required, or for music, for example, a segmentation based on beats or bars should be chosen.[2]

4.2 Feature Issues

In an incremental processing framework, certain constraints are put on feature extraction algorithms. Most important, the processing delay (lag) must be as low as possible, ideally only one audio frame. Consequently, multi-pass processing is not possible, i.e., all audio features used in an incremental system must be extractable in a single processing pass. For instance, features, which are centred to zero mean for one or more utterances cannot be used because these would require the mean value of the feature to be computed in a first pass, and the mean subtraction to be performed in a second pass. Such a normalisation would be applied to pitch, for example, to adapt to speaker specific pitch ranges. For incremental processing a different strategy has to be applied for such normalisations, i.e., the mean values have to be computed on-the-fly and updated with every new incoming value, and then immediately applied. These normalisation issues are discussed in more detail in Sect. 5.2.

Another issue is post-processing of formant and pitch contours (cf. Sects. 2.2.8 and 2.2.11.5, respectively). Thereby typically the best path over the full input is found – given some side constraints such as smoothness or the most likely range—and chosen as the final contour. In incremental processing, it is not possible to wait for all input to be complete. Neither is it feasible in this case of Viterbi based post-smoothing, for example, to update the best path decision with every frame, as in the case of an incrementally updated mean value. This would on the one side involve a memory overhead, as N best paths, which are constantly growing, need to be stored, and on the other side would also introduce long delays, as it cannot be predicted at which point in the past the paths will merge to a single path (and thus a decision on the best path can be made for a frame at time $(t - \tau)$—cf. Sect. 2.2.11.5 for details).

The features used and described in this thesis (Sect. 2.2) have all been designed to be extractable in incremental processing, or suitable modifications have been made to limit the processing lag to an acceptably small, constant maximum lag. As LLDs are extracted frame by frame anyhow, virtually all LLDs can be extracted incrementally—except for global post-processing and smoothing applied to the raw LLDs as discussed above. Supra-segmental features (cf. Sect. 2.4), however, are computed from LLDs on a higher temporal level. For on-line processing, this level is the most important factor which determines the lag of the overall processing. Segments

[2] Incremental segmentation of music into beats and bars is not part of this thesis. An on-line segmentation approach for music has not been investigated in this thesis, however, an off-line segmentation method basing on the beat tracker presented by Schuller et al. (2007b) and Eyben et al. (2007) has been used.

must be of finite length and should ideally be small or at least matching the processing delay requirements of the task at hand. The previous section (Sect. 4.1) has dealt with these issues of segmentation.

Some feature extraction algorithms might require certain higher level knowledge to change their extraction behaviour based on it. For instance, features could only be extracted from vowels, when a vowel/non-vowel classifier is employed, or extracted for certain word classes, if a full blown ASR system combined with a part-of-speech tagger is present. In the case of music analysis, features such as Pitch Class Profiles (Sect. 2.2.14.2) should be extracted from beat or bar segments (cf. Schuller et al. 2008), for example, to obtain a better spectral accuracy. In any case, the limiting factor for incremental and real-time processing is the computational complexity and delay introduced by the secondary processing (e.g., ASR or beat tracker). Generally such features are suitable for being used in on-line processing, however, they are not considered in this thesis because it would be outside of the scope of this thesis to properly deal with the overhead introduced by combining such algorithms with the idea of incremental processing.

4.3 Architecture of the openSMILE Framework

For this thesis, extraction algorithms for all of the acoustic features and functionals described in Sects. 2.2 and 2.4 have been developed from scratch in C++. The main goals thereby were the development of an efficient, incremental framework and the extensive coverage of a wide range of descriptors. The code has been released by the author as an open-source toolkit called openSMILE[3] (Eyben et al. 2010a, 2013a). The toolkit has been used extensively to provide high quality baseline features for the series of international evaluation challenges at INTERSPEECH (see Chap. 3 and Schuller et al. 2009b, 2010, 2011b, 2012b, 2013). Alone Eyben et al. (2010a) has been cited over 250 times at the time of writing.[4] From these citations, more than 160 are publications from groups around the world which are not affiliated with the author's team. openSMILE has won an honourable mention award twice at the ACM Multimedia open-source software competition[5] (Eyben et al. 2010a, 2013a).

This section describes the software architecture and the incremental data-processing framework of openSMILE (Eyben et al. 2010b, 2013b) in detail as published in the *openSMILE book*[6] (Eyben et al. 2010c, 2013c) by the author of this thesis. openSMILE's core architecture is designed in a modular way: Feature extractors are composed of one or more components and each component has a

[3] Available at: http://opensmile.audeering.com/.

[4] According to Google scholar citations.

[5] http://sigmm.org/Resources/software/ossc.

[6] Available for the latest version at http://www.audeering.com/research-and-open-source/files/openSMILE-book-latest.pdf.

clearly defined interface, both for data and for parameter settings. A core framework, which connects the components, is built of three parts:

1. a *commandline parser* and a *configuration manager* parse commandline options and configuration files and provides the initial configuration parameters to the respective components,
2. a *component manager* takes the role of a "scheduler" and manages the component execution and asynchronous communication between components (messages) as well as pausing and resuming of the processing, and
3. a *data-memory* efficiently manages the incremental data-flow between the components.

The component manager is the most important component, as it controls the order of execution of the data processing modules (components). It is responsible of instantiating, configuring, and executing all components. The following gives a software architectural description of the interplay between component manager, the component classes, and the data-memory:

The components instantiated by the component manager (data processing components) are all descendants of the `cSmileComponent` class. They have two basic means of standardised communication: (a) directly and asynchronously, via smile messages, and (b) indirectly and synchronously via the data memory.

Method (a) is used to send out-of-line data, such as events and configuration changes directly from one smile component to another. Classifier components, for example, send a 'classificationResult' message, which can be received by other components (esp. custom plug-ins), to change their behaviour or send the message to external sources. More details on the smile messaging architecture are found in Sect. 4.3.2.

Method (b) is the standard method for handling data-flow (audio input and features) in openSMILE. Thereby a *data source* component produces a frame (vector) of data and writes it to the *data memory* to a named location (referred to as level or slot). A *data processor* reads this frame, applies some processing algorithm to it, and writes a modified frame back to a different location in the data memory. This step is usually repeated for multiple data processors. Finally, a data sink reads the frame and passes it to an external handler (file, network stream) or interprets (classifies) it in some way and sends the result as a message (Method (a)) to another component or displays it on the console, for example. The advantages of passing data in this way via the data memory are that multiple components can read the same data, and data from past frames can be stored efficiently in a central location for later use. This eliminates the need to compute the Fast Fourier Transformation (FFT) twice if both Mel-Frequency Cepstral Coefficients (MFCCs) and spectral roll-off points, for example, shall be extracted. This data-flow architecture is described in more detail in Sect. 4.3.1.

A set of source, processing, and sink components as well as their interconnections can be selected and instantiated via a text-based configuration file. Parameters of each component can also be set in this same file. To enable efficient on-line processing, openSMILE requires the parameters chosen for all algorithms, and especially

the frame sizes and data-flow connections between components to remain constant throughout the run-time. This constraint enables all settings to be loaded and verified once, buffers of the right size to be allocated, and models files, parameter files, and input/output files to be loaded or opened at start-up. This is implemented by three distinct execution states, the first two involving setup of the system and the data-processing channels, and the third involving the actual processing loop:

Pre-config phase Command-line options and the configuration file(s) is/are parsed,

Configuration phase The component manager is created and instantiates all components referenced in the configuration file. Components do their individual initialisation (loading of files, creating and initialising parameters and models, optimisations to speed up computations, etc.) and allocate their I/O requirements in the data-memory. At the end of the configuration phase the ideal minimal memory requirements for the system can be allocated in the data-memory.

Execution phase When all components have been initialised successfully, the component manager starts the main execution loop.[7] In one iteration of the execution loop, the component manager runs every component once. For this purpose, every component has a single `tick()` method, which implements the main incremental processing functionality and reports on the status of the processing via its return value. The component checks for the availability of new input data (which is fully flexible and can be samples, frames, segments, etc.) and processes the new chunk of data, if available. The loop is continued as long as at least one component's tick method returns a status which indicates that data has been processed by this component.

If all components have indicated that they do not have any data to process, it can be safely assumed that no more data will arrive and the end of the input has been reached—which could be the end of an input file in the case of off-line processing, or stopping/pausing of the recording in on-line processing. In that case, the component manager signals the End-of-input (EOI) condition to the components by running one final iteration of the execution loop with an end-of-input flag set. After that, the component manager runs the execution loop again, until all components again report that no data have been processed. This second phase is referred to as end-of-input processing. It is only important for off-line processing, e.g., to compute features from the last (but incomplete) frames, to mean normalise a complete sequence, or to compute functionals from a complete sequence. Since version 2.0, openSMILE supports multiple iterations of this overall loop, i.e., the normal processing can be run again for an arbitrary number of times in a non EOI state after the execution loop has been run in a EOI state. This obviously is unsuitable for on-line processing, but brings openSMILE's algorithms and efficiency also to multi-pass batch and off-line processing.

[7] Also referred to as tick-loop in the code.

4.3.1 Incremental Processing

Figure 4.1 shows the overall incremental data-flow architecture of openSMILE, where the data memory is the central link between all components that process data. Three types of data processing components exist: the *data source* components which write data to the memory from external sources, such as devices, files, or streams; the *data processor* components which retrieve data from the data memory, modify them, and copy them back to the memory to a different location; and the *data sink* components which can only read data from the memory (and send it to devices, files, classifiers, etc.).

The incremental data-flow in openSMILE is handled by the `cDataMemory` component. This component manages multiple data memory 'levels' internally. These levels are independent storage locations, to which data can be written to by exactly one component and data can be read from by an arbitrary number of components. From the outside (the component side) the levels appear to be a $N \times \infty$ matrix, with N rows, whereby N is the frame size. Components can access frames (=columns) at any location in this virtual matrix (virtual index). Internally this matrix is either organised as a ring buffer (for on-line processing) with a predefined length or a buffer which grows dynamically to the needed length. If the matrix is internally represented by a ring-buffer, a write operation only succeeds if there are empty frames

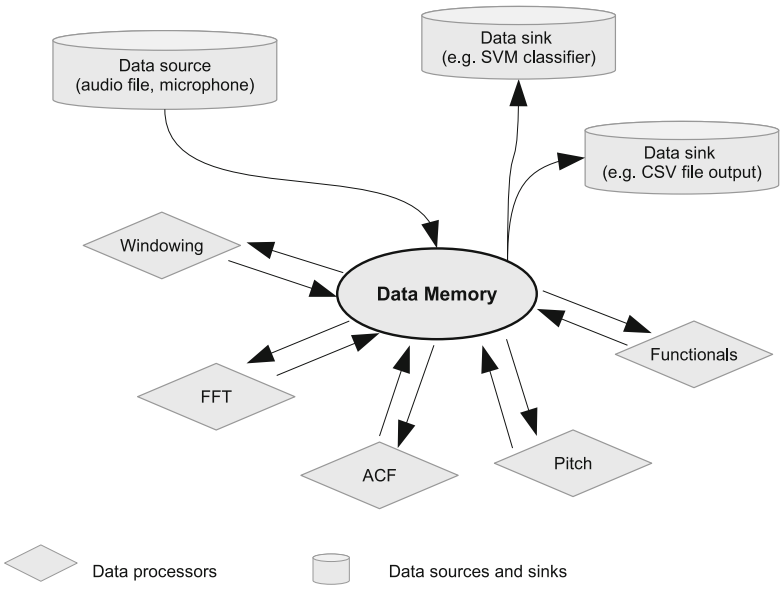

Fig. 4.1 openSMILE's basic data-flow architecture with openSMILE's generic component types. A central data memory to which data source components feed data, data processors modify the data, and data sink components pass the data on to classifiers, outputs, or external processing (e.g., a user interface); cf. Eyben et al. (2010a)

in the buffer, and a read operation only succeeds if the referred frame index lies no more than the ring buffer size in the past and has not yet been overwritten by new data.

In the case of a fixed length (non-ring) buffer, a write will always succeed as long as the referenced index is smaller than the buffer size. A read will succeed if the read position index is greater equal 0 and smaller than or equal to the highest index where data has already been written to. In case of a variable size buffer (i.e., growing dynamically) a write will succeed always, except when there is no memory left. A read will succeed under the same conditions than for the fixed buffer case.

In order to allow for a robust, synchronised, and memory efficient data-flow with ring-buffers, the data memory must track to where data have already been written to and which data have already been read. To achieve this, the data memory performs frame accounting on the internal buffer matrix: A writer pointer p_w, that stores the location to which the next sample/frame is to be written, and a set of read pointers $p_{r,i}$ (with $i = 0 \ldots N_r - 1$, and N_r representing the number of components which read from the current level) is defined for this purpose. The pointers always store a virtual index, i.e., the index in an infinite[8] buffer starting at element 0 regardless of the actual internal organisation of the buffer (cyclic/ring, fixed-length, or dynamic length).

The write pointer is updated after every successful write operation to the index n with a `max` operation: $p'_w = max(p_w, n + 1)$. The read pointers for each reader are updated similarly using the same rule: $p'_{r,i} = max(p_{r_i}, n + 1)$. For calculation of the free space in the buffer, the lowest read index $p_r^{(min)} = min(p_{r,i})$ is considered. With the buffer size N_b, the free space in a non ring buffer is given by $N_{free} = N_b - p_w$, and the number of elements available in the buffer is defined logically as $N_{avail} = p_w$. For a ring buffer, the following equations apply (ring buffer is denoted in the symbols by $\overline{N}$):

$$\overline{N}_{free} = N_b - \left(p_w - p_r^{(min)}\right), \tag{4.1}$$

$$\overline{N}_{avail} = max\left(N_b, p_w\right), \tag{4.2}$$

$$\overline{N}_{avail}^{(i)} = p_w - p_{r,i}, \tag{4.3}$$

where $\overline{N}_{avail}^{(i)}$ is the number of *unread/new* frames that are available to read for the reader i, while $\overline{N}_{avail}$ is the maximum number of frames available to read.

The ring-buffer based incremental processing is illustrated by a simplified example—which does not always use real components nor realistic frame sizes—in Fig. 4.2. Three exemplary levels are contained in this setup: wave (top), frames (middle), and energy (bottom). A `cWaveSource` component saves data samples (scalar values, frame size equal to 1) to the first level (wave). The write indices in the levels are indicated by a bold arrow. A `cFramer` produces frames of size 3 from the wave samples (non-overlapping) and saves these frames to the next level (frames). A `cEnergy` component extracts Root Mean Square (RMS) and logarithmic energy

[8] In practice limited by the range of the 'long' data-type (32-bit or 64-bit).

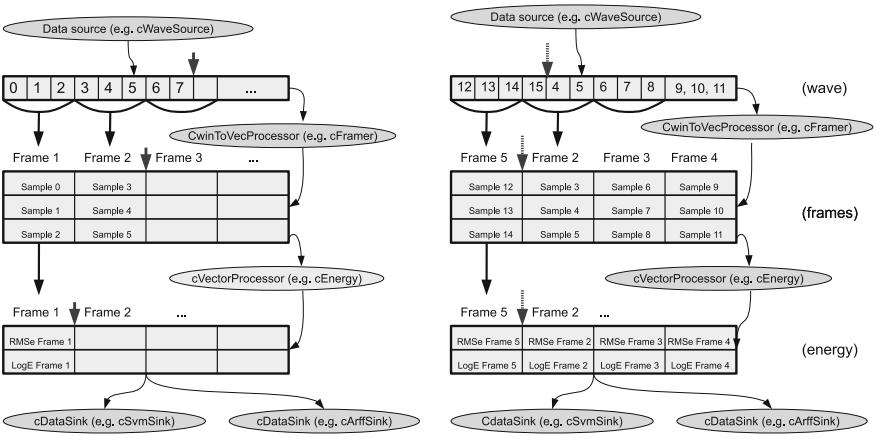

Fig. 4.2 Exemplary (simplified) incremental processing with ring-buffers as implemented in openSMILE. The plot shows partially filled ring-buffers (*left*) and filled ring-buffers with warped write pointers (*right*). Write pointers are shown by *red arrows* on the *top left corner* of the frame column; read pointers are *not* shown in order to keep the diagrams simple; cf. Eyben et al. (2010a)

features from the frames and saves them to the following level (energy). In Fig. 4.2 (right) the buffers have been filled once, and the write pointers have been warped. Data which are more than $N_b = 4$ frames in the past have been overwritten.

The *core concept* of incremental processing in openSMILE is that each component constantly checks, i.e., in each execution loop iteration, if enough new data which can be processed, is available, and—if so—immediately processes this data and makes it available for the next component(s) in the data-flow pipeline. This way, a new data sample induced into the pipeline by a data source will immediately propagate through the processing pipeline, as deep and as quickly as it is possible. In the example in Fig. 4.2 the `cFramer` component requires three input frames to compute one output frame, while the `cEnergy` component gives an output for every input frame. If it is now assumed that the `cWaveSource` component writes only a single sample to the 'wave' level during each iteration, the `cFramer` component as well as the components reading the output of the `cFramer` component are able to process data only in every third iteration.

This concept can be extended to higher levels of features without limitation. This makes the incremental processing architecture proposed for this thesis very flexible and powerful as it allows to build arbitrary high hierarchies of feature summaries. Figure 4.3 shows a simplified example for incremental computation of higher order features. Functionals (mean and variance) computed over two frames (overlapping) of the energy features (from the previous example) are extracted and saved to a functionals level. From the data in this level, for example, again functionals could be computed for the next higher level of feature summaries. In the given example, the data is read by two data sink components, where one writes the data to a Attribute Relation Feature Format (ARFF) file and the other classifies the data with a Support Vector Machine (SVM).

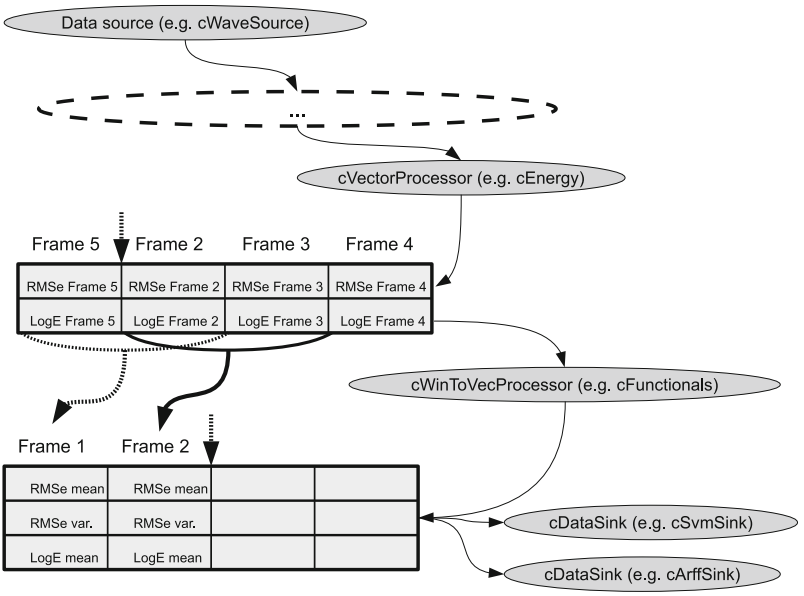

Fig. 4.3 Simplified example of incremental computation of higher-level features and feature summaries (e.g., statistical functionals). The *upper block* is left out in this figure (*dash-lined circle*), it is shown in Fig. 4.2; cf. Eyben et al. (2010a)

The size of the ring buffers of subsequent data memory levels in the processing pipeline must be set correctly according to the processing block size in order to ensure smooth processing without deadlocks. The *block size* thereby is the size of the block a reader or writer reads/writes from/to the data memory at once, i.e., in one iteration. In the above example the read block size of the functionals component is 2 because it reads two pitch frames to produce one output frame. Similarly, in the previous example the read block size of the `cFramer` component was 3. The ring buffer of the level with energy features thus must hold at least 2 frames, otherwise the functionals component will never be able to read its required input from that level and it will block the processing. This will cause the component manager to stop the processing loop because the pitch component will not be able to write new data to the pitch level, thus it will not read data from the frames level, which causes the framer component to be blocked. This process continues until all components are in a state where they do not process any data.

In order to avoid this kind of problems in complex data-flow setups, the adjustment of the buffer size is handled automatically: Reader and writer components must register with the data memory during the configuration phase and must publish their required maximum read and write block sizes, i.e., the size of the largest block the component will ever attempt to read, or to write, respectively. This might not be possible in all cases. In such cases the buffer sizes of the input and output levels have to be constrained manually in the configuration. The minimal buffer size bs_{min} is computed based on these values of minimum and maximum read/write block sizes:

$$bs_{min} = \begin{cases} 2n_w + 1 & \text{if } n_r \leq n_w \\ n_r + 2n_w & \text{if } n_r > n_w \end{cases}, \tag{4.4}$$

where n_r is the maximum read block size (from the current level) and n_w is the maximum write block size (to the current level).

4.3.2 Smile Messages

The data-flow framework described in the previous section enables the exchange of sequential, synchronous data between components. However, sometimes it might be required to exchange small amounts of asynchronous data between components, e.g., to change parameters at run-time, or to send events from one component to another.

To fulfil these needs, a messaging system has been implemented in openSMILE. As the messages are intended only for fast, simple exchange of a handful of parameters or small text string, a simple fixed size message structure has been defined. Compared to more flexible text-markup based messages such as XML messages this structure has very low overhead because no parsing is required. Also, no third-party dependencies are added to the source-code tree, which keeps the software simple and easy to compile and maintain.

The message structure holds eight integer values, eight 32-bit floating point values, a fixed length text field, a message name, and a message type. This is sufficient for most cases to send single values, classification results (class names or regression values), or control commands across components. In case there is need for more data to be sent, two pointers to custom extension data blocks have been reserved.

A message is sent out of the context of the sender component by calling a `sendMessage` method in the component manager. The component manager then calls the `processMessage` method in the receiving component instance, which is responsible of copying the message parameters of interest to local variables in the context of the receiving component.

4.4 Fully Continuous Speech Emotion Recognition

In this section, a novel time and value continuous modelling approach with Long Short- Term Memory Recurrent Neural Network (LSTM-RNN) is introduced (cf. Eyben et al. 2012) for a speech emotion recognition task. Thereby, modelling of dimensional affect ratings is performed based on the incremental segmentation method shown in Sect. 4.1.2 as well as directly from LLDs. Additionally, multi-task learning is applied to estimate the confidence of the automatic predictions and exploit dependencies between multiple affective dimensions.

The next section (Sect. 4.4.1) gives an overview of the state-of-the art methods in emotion recognition with continuous, dimensional labels under realistic conditions. Next, the proposed method for robust, low-latency, continuous multi-task, dimensional affect recognition is introduced and described in Sect. 4.4.2, and a suggestion for reduction of the number of acoustic parameters is given in Sect. 4.4.3. The evaluation protocol and the obtained results are presented and discussed in the next chapter in Sect. 6.4.

4.4.1 Related Work

Recent work has addressed the challenge of automatically identifying affect in natural conversations. Devillers et al. (2005) associate speech turns with multiple targets (mixtures of discrete emotion categories) based on a realistic, affective speech data set which contains non acted speech. Schuller et al. (2007c) also investigate emotion recognition on realistic and non prototypical speech. The recent INTERSPEECH challenges have attracted a lot of attention and have nourished advances in the field, but also demonstrated how challenging the topic is (Schuller et al. 2009b, 2010, 2011c, b, 2012b, 2013).

In contrast to a discrete, categorical representation of affect, such as the basic six emotion categories by Ekman and Friesen (1975), a continuous dimensional representation (Fontaine et al. 2007) could be better suited to capture a plethora of affective states and subtle differences among these (Fontaine et al. 2007; Douglas-Cowie et al. 2007; McKeown et al. 2012). Yet, automatic affect recognition in a dimensional label space is an active, but still young area of research (Grimm et al. 2007b; Wöllmer et al. 2008; Schuller et al. 2009a; Gunes and Pantic 2010b; Eyben et al. 2011; Gunes and Pantic 2010a; Schuller et al. 2012a). Currently, a common strategy is to reduce the dimensional emotion classification problem to a two-class task: e.g., positive versus negative valence or high versus low activation classification, for example (cf. e.g., Nicolaou et al. 2010; Schuller et al. 2009c; Wöllmer et al. 2010), a four-class task (classification into the quadrants of the two-dimensional valence-activation space (cf. e.g., Caridakis et al. 2006; Fragopanagos and Taylor 2005; Glowinski et al. 2008; Ioannou et al. 2005), or to a dynamic number of categories by automatic identification of clusters in the emotional space (Wöllmer et al. 2009; Lee et al. 2009; Wöllmer et al. 2010). Fixed clusters or categories, however, introduce the problem of ambiguity again: Instances with an actual (continuous) label on or near to the category/cluster boundary are likely to be misclassified. A possible solution to this problem is a regression model which directly predicts the continuous values.

Only very few publications using regression models on speech exist so far: e.g., Grimm et al. (2007a) use Support Vector Regression (SVR) to predict affect in three dimensions (activation, valence, power/dominance); Wu et al. (2010a) fuse three methods: robust regression, SVR, and locally linear re-construction; Wöllmer et al. (2008) use LSTM-RNNs and SVR; the work presented by Wöllmer et al. (2010) utilises a Bidirectional Long Short-Term Memory Recurrent Neural Network

(BLSTM-RNN) to perform regression for affect dimensions and then quantises the resulting predictions into four quadrants. The author's previous work in Eyben et al. (2010d) investigates a regression technique for continuous dimensional affect recognition which extends the work by Wöllmer et al. (2008). Alternatives to SVR include linear regression (Cohen et al. 2003), radial base function networks (Yee and Haykin 2001), and standard feed-forward neural networks. In the context of affect-sensitive virtual agents, Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) (see Sect. 2.5.2.2) have been applied (Peters and O'Sullivan 2002; Eyben et al. 2010d). One of the first attempts to get research teams globally to work on a comparable task of fully continuous emotion recognition was the second AVEC (Schuller et al. 2012a).

For regression analysis of emotion, the use of confidence measures has so far been—to the best of the author's knowledge—neglected. In closely related literature no experiments on confidence estimation of regression predictions have been reported. While the importance of confidence measures has risen with methods of unsupervised, semi-supervised (Deng and Schuller 2012; Zhang and Schuller 2012), and active learning (Han et al. 2013) gaining attention, this has only been explored for categorical labelling, so far (e.g., Deng et al. 2012). A method for dimensional music mood prediction which includes a confidence measure has been introduced by Schmidt and Kim (2010), who use linear regression to estimate the mood coordinates of a song excerpt in a two-dimensional valence-activation space. The uncertainty of the regressor is thereby modelled as a second regression target in a multi-target regression framework. Schmidt and Kim (2010) collected a continuously annotated database for training and evaluating the system with an on-line game application. There, participants had to label the current mood of a song by moving their mouse in a valence-activation plane while competing against another rater.

In this thesis a similar attempt to estimate the confidence for speech affect by multi-task learning is presented in Sect. 4.4.2.3: LSTM-RNNs are trained on the inter-rater standard deviation and the mean label for each dimension. Both, modelling each dimension individually and modelling all dimensions in one network is investigated. This multi-task approach is partially inspired by the work of Steidl et al. (2009), who employ a similar technique to estimate class confidences.

4.4.2 Proposed Continuous Modelling Approach

The approach towards incremental affect recognition in real-time chosen in this thesis is based on (bidirectional) LSTM-RNN regressors (see Sect. 2.5.2.2). Such networks can successfully model long-term dependencies between input data, which makes them suitable for supra-segmental-modelling of time connected segments as well as for modelling of LLDs directly. As any neural network, they are able to handle multi-dimensional target patterns, which enables multi-task learning as used here for the estimation of a confidence measure and true multi-dimensional affect prediction.

4.4.2.1 Segmentation Issues (LLD vs. Supra-Segmental)

As opposed to speech recognition, emotion recognition from isolated short-time audio frames is virtually impossible: while single phonemes are highly correlated to a specific spectral representation in short signal windows, speech emotion is a phenomenon observed over a longer time window (typically more than 1–2 s). Thus, supra-segmental modelling is typically applied (Sects. 2.4 and 2.5.1) with units of analysis that are complete sentences, sentence fragments (i.e., chunks, e.g., by grammatical rules) or words (Steidl 2009).

In order to enable output of emotion predictions at constant time intervals, independent of word or phrase level segmentations, the incremental segmentation scheme described in Sect. 4.1.2 is applied. Thereby $L_{step} = 1\,\text{s}$ and $L_{win} = 5\,\text{s}$ are chosen, and the variant (1.) of the algorithm is used. Temporal dependencies among the segments remain due to the overlap of the segments and the slow changing affect labels. These dependencies can be exploited by LSTM-RNNs, as is shown by a comparison to standard RNNs, Feed-Forward Neural Networks (FFNNs), and SVR in Sect. 6.4.2. While theoretically there is no explicit mention that long range temporal context is necessary for successful identification of emotional states, inspection of the data shows that in conversations as contained in the SEMAINE database (Sect. 6.1.9), for example, emotion does not change rapidly nor often and is dependent on the context of the discussion (cf. also Lee et al. 2009).

A major disadvantage of supra-segmental approaches is that one complete input segment is required for analysis and only a single output is produced at the end of every segment. This disadvantage is partially mitigated by the incremental segmentation approach used. However, for fully continuous affect recognition the requirement of having to define analysis segments must be abandoned. Under ideal circumstances (frame-wise) LLDs should be used as input to a classifier/regressor which can connect these inputs over time in a meaningful way and produce a prediction for every short-time frame. The regressor employed here—LSTM-RNN—is capable of this as was shown, e.g., by Eyben et al. (2010d). Thus, in this thesis it is evaluated to which extent the modelling of LLDs without any higher level feature summarisation is feasible. Thereby, each speech utterance is viewed as one sequence when training the LSTM-RNN.

4.4.2.2 Modelling with LSTM-RNNs

LSTM-RNNs with two fully connected hidden layers with 10 – 140 LSTM cells are used to model the five dimensional affective labels (both individually and jointly). Various uni- and bidirectional network configurations are evaluated in Sect. 6.4.2. Two cases are distinguished: modelling of frame-wise LLDs (rate of 10 ms) and supra-segmental features (rate of 1 s.). When modelling supra-segmental features, a full recording session, i.e., the series of all speech utterances in the order they occur in the original, continuous recording, represents one sequence. This means, the network has access to temporal context across the full progression of the recording session,

e.g., a full conversation. In the case of LLD modelling, sequences are limited to a speech utterance or a turn (for comparison: typically 1 – 10 supra segmental feature vectors), in order to avoid numerical problems when training a LSTM-RNN with Backpropagation Through Time (BPTT) on very long sequences.

The networks are trained with gradient descent BPTT (Werbos 1990) or gradient descent Resilient Propagation (rProp) (Riedmiller and Braun 1993)—for details, see Sect. 2.5.2.2. When doing so, an initial set of non-zero network weights must be chosen as starting point for the gradient descent weight updates, which is usually done with a random number generator. However, this makes the training likely to converge in local minima of the target error function, dependent of the initialisation. A possible way to reduce the influence of the initial weights is to train $N > 1$ networks with varying initialisations, and at the end take the average of all N output activations. This method is applied here, with $N = 5$ runs and with fixed random seeds 0, 1, 2, 3, 4 for the 5 runs, respectively.

During training of the networks Gaussian white noise with $\sigma = 0.3$ is added to the features. This technique is commonly used to improve the generalisation capabilities of neural networks (cf. e.g., Fernandez et al. 2008; Graves and Schmidhuber 2005). Overall, this leads to a longer training time (more iterations). However, it helps to avoid over-fitting and therefore potentially will improve the performance on the evaluation and development sets, especially for smaller data-sets.

4.4.2.3 Multi-target Learning

A novel concept investigated by the author of this thesis for the first time is multi-target learning of dimensional affect and rater-agreement in a single model (Eyben et al. 2012). Previous work of the author on dimensional affect recognition has dealt with single target learning for the affective dimensions activation and valence only (Eyben et al. 2010d). With neural networks, multi-target learning is no different from single target learning, except for the topology of the network's output layer: when predicting one continuous dimension the output layer consists of a single linear summation unit; for multiple targets the number of linear summation units in the output layer is equal to the number of targets.

Two aspects are investigated in this thesis:

1. instead of training individual networks for every affective dimension, a single network with multiple outputs is applied to predict multiple affective dimensions simultaneously,
2. the variance of multiple individual raters' labels is learnt as a second target for each dimension.

It is assumed, that—based on this variance measure—the network might learn to predict a confidence measure for its output based on human agreement levels.

4.4.3 Acoustic Features

The INTERSPEECH 2010 Paralinguistics Challenge (IS10) baseline set (Sect. 3.2) is chosen as acoustic parameter set for the continuous modelling. It contains 1,582 features.

Because training of the LSTM-RNN with a high number of inputs (1.5 k for the supra-segmental features) suffers more from data sparsity than related work based on SVR (see Sect. 6.4.2), a Correlation-based Feature-subset Selection (CFS) is applied to the training set to determine relevant supra-segmental features for each of the five dimensions. The development and evaluation data are not used in the feature selection process to ensure fair evaluation conditions. The CFS algorithm (Hall 1998) evaluates the worth of each feature subset by considering the ability of each individual feature to predict the class or dimensional numeric label, along with the degree of redundancy between the features. Feature subsets that are highly correlated with the target while having low intra-correlation are preferred.

Out of 1,582 features, after CFS, 38 features remain for activation, 39 features for expectation, 30 features for intensity, 32 features for power, and 28 features for valence. Instead of including a full list of selected features, the most frequently occurring LLDs for each dimension sorted by frequency of occurrence (given in parentheses) are given in the following (Eyben et al. 2012):

- **Activation**: MFCCs (16), log. Mel-Frequency Bands (MFBs) (9), Line Spectral Frequencies (LSFs) (5), loudness (4), Jitter (2).
- **Expectation**: MFCCs (18), F_0 (7), LSFs (7), loudness (3), log. MFBs (2)
- **Intensity**: MFCCs (11), loudness (7), LSFs (6), log. MFBs (5)
- **Power**: MFCCs (24), log. MFBs (3), LSFs (3), F_0 (2)
- **Valence**: MFCCs (14), LSFs (7), log. MFBs (4)

It can be seen that functionals applied to MFCCs are constantly the most frequently selected features (also supported by Wu et al. 2010b, for example), which must partially be accounted to the fact that MFCCs make up a large portion of the original set. Besides MFCCs, however, there are some variations in the selected features observable across the five dimensions, which can be summarised as follows: for *activation*, spectral band energies, format frequencies (related to LSFs), and loudness seem to be most important, while for *expectation* F_0 is a key factor, which appears logical when considering that surprise is the primary related emotion category with a low value of expectation; for *intensity*, a similar result as for activation is found with a minor tendency towards loudness being more important here; for power, mostly MFCC based features are selected, which could indicate that the subject's dominance is reflected primarily by the way of articulation and least by prosody; *valence* seems to be described best by a mixture of MFCCs and LSFs, again being a hint on the possible importance of the way of phonation / vocal timbre, or even linguistic content.

For multi-target learning, the reduction of the feature set to a single set of relevant features is more complex because all targets have to be considered when evaluating the worth of the features. Instead of extending CFS to multiple targets, it was decided

to use the joint feature set, i.e., the union set of the five CFS reduced feature sets (for each dimension individually). In this way, 138 features are selected. This number is below the sum of the number of features selected for each dimension individually (167), which indicates the presence of a small overlap of the reduced features sets, i.e., features which are relevant for more than one dimension.

Features which were found to be relevant for at least three dimensions are the skewness of the third MFCC (for all four dimensions), the percentage (temporal) where the MFCC 7 is above 75 % of its range (up-level time), and the Inter-Quartile Range (IQR) 2–1 of the 6th Line Spectral Pair (LSP) frequency, as well as the second quartile of the 7th LSP frequency. There are 24 features which were selected for at least two dimensions. The most frequent LLDs among these are loudness, the 0th and 6th LSP frequency, MFCC 10, and the probability of voicing.

For the direct modelling of the LLDs with the LSTM-RNN no feature reduction is performed. Due to the smaller size of the set the reduction was not necessary: The IS10 set contains 76 LLDs (including delta regression coefficients—cf. Sect. 3.2).

References

A. Batliner, D. Seppi, S. Steidl, B. Schuller, Segmenting into adequate units for automatic recognition of emotion-related episodes: a speech-based approach. Adv. Human Comput. Interact., Special Issue on Emotion-Aware Natural Interaction **2010**, 1–15 (2010). Article ID 782802 (on-line)

M. Ben-Ari, *Principles of Concurrent and Distributed Programming* (Prentice Hall, Englewood Cliffs, 1990). ISBN 0-13-711821-X

C. Busso, S. Lee, S. Narayanan, Using neutral speech models for emotional speech analysis, in *Proceedings of the INTERSPEECH 2007*, Antwerp, Belgium, August 2007. ISCA, pp. 2225–2228

G. Caridakis, L. Malatesta, L. Kessous, N. Amir, A. Raouzaiou, K. Karpouzis, Modeling naturalistic affective states via facial and vocal expressions recognition, in *Proceedings of the 8th International Conference on Multimodal Interfaces (ICMI) 2006*, Banff, Canada, 2006. ACM, pp. 146–154

J. Cohen, P. Cohen, S.G. West, L.S. Aiken, *Applied multiple regression/correlation analysis for the behavioral sciences*, 2nd edn. (Lawrence Erlbaum Associates, Hillsdale, 2003)

J. Deng, B. Schuller, Confidence measures in speech emotion recognition based on semi-supervised learning, in *Proceedings of INTERSPEECH 2012*, Portland, September 2012. ISCA

J. Deng, W. Han, B. Schuller, Confidence measures for speech emotion recognition: a start, in *Proceedings of the 10-th ITG Symposium on Speech Communication*, ed. by T. Fingscheidt, W. Kellermann (Braunschweig, Germany, September 2012). IEEE, pp. 1–4

L. Devillers, L. Vidrascu, L. Lamel, Challenges in real-life emotion annotation and machine learning based detection. Neural Netw. **18**(4), 407–422 (2005). doi:10.1016/j.neunet.2005.03.007. ISSN 0893-6080

E. Douglas-Cowie, R. Cowie, I. Sneddon, C. Cox, O. Lowry, M. McRorie, J.C. Martin, L. Devillers, S. Abrilian, A. Batliner, N. Amir, K. Karpouzis, *The HUMAINE Database*, vol. 4738, Lecture Notes in Computer Science (Springer, Berlin, 2007), pp. 488–500

P. Ekman, W.V. Friesen, *Unmasking the Face: A Guide to Recognizing Emotions from Facial Expressions* (Prentice Hall, Englewood Cliffs, 1975)

F. Eyben, B. Schuller, S. Reiter, G. Rigoll, Wearable assistance for the ballroom-dance hobbyist – holistic rhythm analysis and dance-style classification, in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME) 2007*, Bejing, China, July 2007. IEEE, pp. 92–95

F. Eyben, M. Wöllmer, B. Schuller, openSMILE – the munich versatile and fast open-source audio feature extractor, in *Proceedings of the ACM Multimedia 2010*, Florence, Italy, 2010a. ACM, pp. 1459–1462

F. Eyben, M. Woellmer, B. Schuller, openSMILE version 1.0.1 – source code, GPL. http://opensmile.sourceforge.net, 2010b

F. Eyben, M. Woellmer, B. Schuller, The openSMILE documentation v. 1.0.1. http://sourceforge.net/projects/opensmile/files/openSMILE_book_1.0.1.pdf/download, 2010c. Documentation of the the openSMILE toolkit, referred to as openSMILE book, online version 1.0.1

F. Eyben, M. Wöllmer, A. Graves, B. Schuller, E. Douglas-Cowie, R. Cowie, On-line emotion recognition in a 3-D activation-valence-time continuum using acoustic and linguistic cues. J. Multimodal User Interfaces (JMUI) **3**(1–2), 7–19 (2010d). doi:10.1007/s12193-009-0032-6

F. Eyben, M. Wöllmer, M. Valstar, H. Gunes, B. Schuller, M. Pantic, String-based audiovisual fusion of behavioural events for the assessment of dimensional affect, in *Proceedings of the International Workshop on Emotion Synthesis, Representation, and Analysis in Continuous space (EmoSPACE) 2011, held in conjunction with FG 2011*, Santa Barbara, March 2011. IEEE, pp. 322–329

F. Eyben, M. Wöllmer, B. Schuller, A multi-task approach to continuous five-dimensional affect sensing in natural speech. ACM Trans. Interact. Intell. Syst., Special Issue on Affective Interaction in Natural Environments **2**(1), 29 (2012). Article No. 6

F. Eyben, F. Weninger, F. Gross, B. Schuller, Recent developments in openSMILE, the munich open-source multimedia feature extractor, in *Proceedings of ACM Multimedia 2013*, Barcelona, Spain, 2013a. ACM, pp. 835–838

F. Eyben, F. Weninger, M. Woellmer, and B. Schuller, openSMILE version 2.0rc1 – source code, open-source research only license. http://opensmile.sourceforge.net, 2013b

F. Eyben, F. Weninger, M. Woellmer, B. Schuller, The openSMILE documentation v. 2.0 rc1. http://sourceforge.net/projects/opensmile/files/openSMILE_book_2.0-rc1.pdf/download, 2013c. Documentation of the the openSMILE toolkit, referred to as the openSMILE book, online version 2.0 rc1

S. Fernandez, A. Graves, J. Schmidhuber, *Phoneme recognition in TIMIT with BLSTM-CTC*, Technical report, IDSIA, Switzerland, 2008

J.R.J. Fontaine, K.R. Scherer, E.B. Roesch, P.C. Ellsworth, The world of emotions is not two-dimensional. Psychol. Sci. **18**(2), 1050–1057 (2007)

N. Fragopanagos, J.G. Taylor, Emotion recognition in human-computer interaction. Neural Netw., 2005 Special Issue on Emotion and Brain **18**(4), 389–405 (2005)

D. Glowinski, A. Camurri, G. Volpe, N. Dael, K. Scherer, Technique for automatic emotion recognition by body gesture analysis, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops 2008 (CVPRW'08)*, Anchorage, June 2008. IEEE, pp. 1–6

A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. **18**(5–6), 602–610 (2005)

M. Grimm, K. Kroschel, S. Narayanan, Support vector regression for automatic recognition of spontaneous emotions in speech, in *Proceedings of the ICASSP 2007*, vol. 4, Honolulu, April 2007a. IEEE, pp. 1085–1088

M. Grimm, E. Mower, K. Kroschel, S. Narayanan, Primitives based estimation and evaluation of emotions in speech. Speech Commun. **49**, 787–800 (2007b)

H. Gunes, M. Pantic, Dimensional emotion prediction from spontaneous head gestures for interaction with sensitive artificial listeners, in *Proceedings of the International Conference on Intelligent Virtual Agents (IVA)*, . (Springer, Berlin, 2010a), pp. 371–377. ISBN 978-3-642-15891-9

H. Gunes, M. Pantic, Automatic, dimensional and continuous emotion recognition. Int. J. Synth. Emot. (IJSE) **1**(1), 68–99 (2010b)

M.A. Hall, *Correlation-based Feature Subset Selection for Machine Learning*, Doctoral thesis, University of Waikato, Hamilton, New Zealand, 1998

W. Han, H. Li, H. Ruan, L. Ma, J. Sun, B. Schuller. Active learning for dimensional speech emotion recognition, in *Proceedings of INTERSPEECH 2013*, Lyon, France, August 2013. ISCA, pp. 2856–2859

S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

S. Ioannou, A. Raouzaiou, V. Tzouvaras, T. Mailis, K. Karpouzis, S. Kollias, Emotion recognition through facial expression analysis based on a neurofuzzy method. Neural Netw., 2005 Special Issue on Emotion and Brain **18**(4), 423–435 (2005)

C.-C. Lee, C. Busso, S. Lee, S.S. Narayanan, Modeling mutual influence of interlocutor emotion states in dyadic spoken interactions, in *Proceedings of INTERSPEECH 2009*, Brighton, UK, September 2009. ISCA, pp. 1983–1986

G. McKeown, M. Valstar, R. Cowie, M. Pantic, M. Schroder, The SEMAINE database: annotated multimodal records of emotionally colored conversations between a person and a limited agent. IEEE Trans. Affect. Comput. **3**(1), 5–17 (2012). doi:10.1109/T-AFFC.2011.20. ISSN 1949-3045

E. Mower, S.S. Narayanan, A hierarchical static-dynamic framework for emotion classification, in *Proceedings of the ICASSP 2011*, Prague, Czech Republic, May 2011. IEEE, pp. 2372–2375

M. Nicolaou, H. Gunes, M. Pantic, Audio-visual classification and fusion of spontaneous affective data in likelihood space, in *Proceedings of the 2010 20th International Conference on Pattern Recognition (ICPR)*, Istanbul, Turkey, August 2010. IEEE, pp. 3695–3699

V. Parsa, D. Jamieson, Acoustic discrimination of pathological voice: sustained vowels versus continuous speech. J. Speech, Lang. Hear. Res. **44**, 327–339 (2001)

C. Peters, C. O'Sullivan, Synthetic vision and memory for autonomous virtual humans. Comput. Graph. Forum **21**(4), 743–753 (2002)

M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the RPROP algorithm, *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1 (San Francisco, 1993). IEEE, pp. 586–591. doi:10.1109/icnn.1993.298623

E.M. Schmidt, Y.E. Kim, Prediction of time-varying musical mood distributions from audio, in *Proceedings of ISMIR 2010*, Utrecht, The Netherlands, 2010. ISMIR

M. Schröder, E. Bevacqua, R. Cowie, F. Eyben, H. Gunes, D. Heylen, M. ter Maat, G. McKeown, S. Pammi, M. Pantic, C. Pelachaud, B. Schuller, E. de Sevin, M. Valstar, M. Wöllmer, Building autonomous sensitive artificial listeners. IEEE Trans. Affect. Comput. **3**(2), 165–183 (2012)

B. Schuller, G. Rigoll, Timing levels in segment-based speech emotion recognition, in *Proceedings of the INTERSPEECH-ICSLP 2006*, Pittsburgh, September 2006. ISCA, pp. 1818–1821

B. Schuller, B. Vlasenko, R. Minguez, G. Rigoll, A. Wendemuth, Comparing one and two-stage acoustic modeling in the recognition of emotion in speech, in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU) 2007*, Kyoto, Japan, 2007a. IEEE, pp. 596–600

B. Schuller, F. Eyben, G. Rigoll, Fast and robust meter and tempo recognition for the automatic discrimination of ballroom dance styles, in *Proceedings of the ICASSP 2007*, vol. I, Honolulu, April 2007b. IEEE, pp. 217–220

B. Schuller, D. Seppi, A. Batliner, A. Maier, S. Steidl, Towards more reality in the recognition of emotional speech, in *Proceedings of the ICASSP 2007*, vol. IV, Honolulu, 2007c. IEEE, pp. 941–944

B. Schuller, F. Eyben, G. Rigoll, Beat-synchronous data-driven automatic chord labeling, in *Proceedings of the 34 Jahrestagung für Akustik (DAGA) 2008*, Dresden, Germany, March 2008. DEGA pp. 555–556

B. Schuller, R. Müller, F. Eyben, J. Gast, B. Hörnler, M. Wöllmer, G. Rigoll, A. Höthker, H. Konosu, Being bored? Recognising natural interest by extensive audiovisual integration for real-life application. Image Vis. Comput., Special Issue onVisual and Multimodal Analysis of Human Spontaneous Behavior **27**(12), 1760–1774 (2009a)

B. Schuller, S. Steidl, A. Batliner, F. Jurcicek, The INTERSPEECH 2009 emotion challenge, in *Proceedings of INTERSPEECH 2009*, Brighton, UK, September 2009b. pp. 312–315

B. Schuller, B. Vlasenko, F. Eyben, G. Rigoll, A. Wendemuth, Acoustic emotion recognition: a benchmark comparison of performances, in *Proceedings of the IEEE Workshop on Automatic*

Speech Recognition and Understanding (ASRU) 2009, Merano, Italy, December 2009c. IEEE,pp. 552–557

B. Schuller, S. Steidl, A. Batliner, F. Burkhardt, L. Devillers, C. Müller, S. Narayanan, The INTERSPEECH 2010 paralinguistic challenge, in *Proceedings of INTERSPEECH 2010*, Makuhari, Japan, September 2010. ISCA, pp. 2794–2797

B. Schuller, M. Valstar, F. Eyben, G. McKeown, R. Cowie, M. Pantic, AVEC 2011 - the first international audio/visual emotion challenge, in *Proceedings of the First International Audio/Visual Emotion Challenge and Workshop, AVEC 2011, held in conjunction with the International HUMAINE Association Conference on Affective Computing and Intelligent Interaction (ACII) 2011*, vol. II, ed. by B. Schuller, M. Valstar, R. Cowie, M. Pantic (Springer, Memphis, 2011a), pp. 415–424

B. Schuller, A. Batliner, S. Steidl, F. Schiel, J. Krajewski, The INTERSPEECH 2011 speaker state challenge, in *Proceedings of INTERSPEECH 2011*, Florence, Italy, August 2011b. ISCA, pp. 3201–3204

B. Schuller, A. Batliner, S. Steidl, D. Seppi, Recognising realistic emotions and affect in speech: state of the art and lessons learnt from the first challenge. Speech Commun., Special Issue on Sensing Emotion and Affect – Facing Realism in **53**(9/10), 1062–1087 (2011c)

B. Schuller, M. Valstar, R. Cowie, M. Pantic, AVEC 2012: the continuous audio/visual emotion challenge - an introduction, in *Proceedings of the 14th ACM International Conference on Multimodal Interaction (ICMI) 2012*, ed. by L.-P. Morency, D. Bohus, H.K. Aghajan, J. Cassell, A. Nijholt, J. Epps (ACM, Santa Monica, 2012a), pp. 361–362

B. Schuller, S. Steidl, A. Batliner, E. Nöth, A. Vinciarelli, F. Burkhardt, R. van Son, F. Weninger, F. Eyben, T. Bocklet, G. Mohammadi, B. Weiss, The INTERSPEECH 2012 speaker trait challenge, in *Proceedings of INTERSPEECH 2012*, Portland, OR, USA, September 2012b. ISCA

B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, et al, The INTERSPEECH 2013 computational paralinguistics challenge: Social Signals, Conflict, Emotion, Autism, in *Proceedings of INTERSPEECH 2013*, Lyon, France, 2013. ISCA, pp. 148–152

S. Steidl, *Automatic Classification of Emotion-Related User States in Spontaneous Children's Speech* (Logos Verlag, Berlin, 2009)

S. Steidl, B. Schuller, A. Batliner, D. Seppi, The hinterland of emotions: facing the open-microphone challenge, in *Proceedings of the 4th International HUMAINE Association Conference on Affective Computing and Intelligent Interaction (ACII)*, vol. I, Amsterdam, The Netherlands, 2009. IEEE, pp. 690–697

P. Werbos, Backpropagation through time: what it does and how to do it. Proc. IEEE **78**, 1550–1560 (1990)

M. Wöllmer, F. Eyben, S. Reiter, B. Schuller, C. Cox, E. Douglas-Cowie, R. Cowie. Abandoning emotion classes – towards continuous emotion recognition with modelling of long-range dependencies, in *Proceedings of the INTERSPEECH 2008*, Brisbane, Australia, September 2008. ISCA, pp. 597–600

M. Wöllmer, F. Eyben, B. Schuller, E. Douglas-Cowie, R. Cowie, Data-driven clustering in emotional space for affect recognition using discriminatively trained LSTM networks, in *Proceedings of INTERSPEECH 2009*, Brighton, UK, September 2009. ISCA, pp. 1595–1598

M. Wöllmer, B. Schuller, F. Eyben, G. Rigoll, Combining long short-term memory and dynamic Bayesian networks for incremental emotion-sensitive artificial listening. IEEE J. Sel. Top. Signal Process., Special Issue on "Speech Processing for Natural Interaction with Intelligent Environments" **4**(5), 867–881 (2010)

D. Wu, T. Parsons, E. Mower, S.S. Narayanan, Speech emotion estimation in 3d space, in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME) 2010*, Singapore, July 2010a. IEEE, pp. 737–742

D. Wu, T. Parsons, S.S. Narayanan, Acoustic feature analysis in speech emotion primitives estimation, in *Proceedings of the INTERSPEECH 2010*, Makuhari, Japan, September 2010b. ISCA, pp. 785–788

P.V. Yee, S. Haykin, *Regularized Radial Basis Function Networks: Theory and Applications* (Wiley, New York, 2001), 208 p. ISBN 0-471-35349-3
Z. Zeng, M. Pantic, G.I. Rosiman, T.S. Huang, A survey of affect recognition methods: audio, visual, and spontaneous expressions. IEEE Trans. Pattern Anal. Mach. Intell. **31**(1), 39–58 (2009)
Z. Zhang, B. Schuller, Semi-supervised learning helps in sound event classification, in *Proceedings of ICASSP 2012*, Kyoto, March 2012. IEEE, pp. 333–336