The Social Signal Interpretation Framework (SSI) for Real Time Signal Processing and Recognition

Johannes Wagner [1]*, Florian Lingenfelser* [1] *and Elisabeth André* [1]

[1]Lab for Human Centered Multimedia, Augsburg University

johannes.wagner@informatik.uni-augsburg.de

Abstract

The construction of systems for recording, processing and recognising a human's social and affective signals is a challenging effort that includes numerous but necessary sub-tasks to be dealt with. In this article, we introduce our Social Signal Interpretation (SSI) tool, a framework dedicated to support the development of such systems. It provides a flexible architecture to construct pipelines to handle multiple modalities like audio or video and establishing on- and offline recognition tasks. The plug-in system of SSI encourages developers to integrate external code, while a XML interface allows anyone to write own applications with a simple text editor. Furthermore, data recording, annotation and classification can be done using a straightforward graphical user interface, allowing simple access to inexperienced users.

Index Terms: speech processing, social signal processing, multimodal fusion, real-time recognition, machine learning

1. Introduction

The correct interpretation of a spoken message often requires incorporation of additional non-verbal signs. For example one might accompany a sentence with a subtle grin to indicate an ironic meaning, or by lowering the head give additional hints that he or she is in a sad mood. In fact, automatic recognition of human's social signals has become an elaborated field of research over the last years. In order to provide the capabilities for sensing, processing and interpretation of speech, facial expressions, gestures, etc. we need to overcome a great number of challenges. A universal framework – equipped with appropriate technical and algorithmical solutions for given requirements – is desirable for the assembly of such applications. Therefore we will first introduce important tasks to be handled by such a framework in general and follow up with the presentation of a practical signal interpretation framework.

Depending on the type of signal we would like to observe, we need to employ sensors, such as cameras, microphones, or haptic devices. Thus *data recording* defines the first task, which combines connection of adequate hardware as well as streaming and synchronization of captured signals. Since recorded signals are just raw data, we need a strategy to separate interesting parts of the signals from periods carrying irrelevant information. This effort is called *data segmentation* and involves the detection of onsets and offsets of meaningful actions. Finally, a detected action has to be classified into one out of a set of predefined categories, or – in case it does not fit to any of the available categories – labelled as unclassified. Sometimes a mapping to continuous values is used instead of discrete states. In any case, two steps are required for adequate categorisation, namely the mapping of the raw signal values onto a set of compact features keeping only the essential information of the segment (*feature extraction*) and the actual *classification* according to a classification scheme with one or more pre-trained classification models.

Obtaining a well fitting classification model is a crucial part of signal processing and recognition and involves several procedures. A *collection* of representative samples is needed in first place. This requires separate recording sessions during which users are either asked to show certain actions or interact with systems that have been manipulated to induce desired behaviours. Afterwards, the collected data is observed by experts who describe the user interaction (*annotation*). Based on categories assigned by the raters, statistical methods can be applied to separate the feature space into sub-regions, with each region representing another category. Several linear (e. g. LDA or Naive Bayes) as well as non-linear (e. g. SVM or ANN) classification techniques exist for this purpose. While the extraction of discriminative features in combination with suited classification is important, one can easily neglected that the success of a model in the first place depends on the quality of the training samples. Here, two general approaches are possible: collecting data from several (ideally representative) subjects to build a universal *user-independent* model; or running new recordings for each user to build personalized *user-dependent* models. While the first approach allows a straightforward use of the system it involves the danger that the model does not adequately render the situation in which it is used. The second approach, however, requires tools that allow a user to record own data and extract a model out of it.

In order to assemble all mentioned parts, we present in this paper our Social Signal Interpretation (SSI) toolbox, a framework for building online recognition systems [1]. In particular SSI supports the following tasks:

- Parallel and synchronized streaming from several sensor devices.
- On-the-fly signal filtering, feature extraction, and classification.
- Database recording, offline sampling and model learning, including tools to evaluate learned models.
- A graphical user interface (GUI) to obtain and evaluate personalized models.

2. Signal Processing Framework

There is a number of free and commercial software related to machine learning. Some of them are specialized in a certain task, such as Weka[1], which offers a large collection of ma-

[1]http://www.cs.waikato.ac.nz/ml/weka/

chine learning algorithms for data mining, or tailored to a certain modality, such as Praat[2] for audio processing. Others, like Matlab[3] or its free counterpart Octave[4], offer more general toolboxes and a simplified scripting language for applying a large body of signal processing algorithms. The variety of tools with support for live sensor input, on the other hand, is considerably smaller. Examples of architectures for multimodal processing, are Pure Data [2], EyesWeb [3] or OpenInterface [4]. Here, developers can choose from a considerable set of input, processing and output components, and patch them using a graphical programming environment to construct complex processing pipelines. However, they are not specially tuned for building machine learning pipelines and collecting training corpora. A toolkit developed for real-time affect recognition from speech is the openEAR toolkit [5] with its feature extracting backend openSMILE [6]. It is, however, limited to audio processing.

The Social Signal Processing framework (SSI) complements existing tools by offering special support for the development of online recognition systems from multiple sensors [1]. SSI covers tasks to assemble the machine learning pipeline, ranging from live sensor input and real-time signal processing to high-level feature extraction and online classification. Different strategies for fusing information from different modalities at data, feature and decision level are available, as well as a generic GUI for data acquisition and model training. Even though SSI is not limited to the task of emotion recognition it has been developed with a focus on sensing of affective and social signals.

SSI is implemented in C/C++ and optimized to run on computer systems with multiple CPUs. Even though SSI is developed under Microsoft Visual Studio, its XML interface allows anyone to write and edit applications with a simple text editor. Like other patch-based tools, a recognition pipeline in SSI is set up from autonomic components. The run-time engine takes care of the execution and synchronization of involved modules and to handle the communication between connected components. This architecture of decoupled elements greatly benefits the exchange and extension of pipelines. For instance, a new filter algorithm can be scheduled ahead of the feature extraction or a different classification scheme can be plugged without touching the remaining parts of the pipeline. To support the interchangeability of components, the framework defines abstract interfaces for four different component types:

- *Sensors* deliver one or more signal streams to the pipeline, either by connecting a physical sensor device or through simulation (e. g. from a file on disk).
- *Transformers* manipulate one or more data streams and feeds it as a new stream back to the pipeline.
- *Triggers* signal the beginning and ending of detected actions.
- *Consumers* receive one or more signal streams without writing back to the pipeline. Here, streams can be stored to disk, streamed to an external application (e. g. via sockets) or visualized in a graph. In combination with a trigger this can also be the seat of a classifier.

The general architecture of SSI and a simple plug-in system suits the integration of external code. External tools, such as OpenCV, ARToolKitPlus, SHORE, Torch, Speex, Watson as well as several tools developed at our lab (e. g. AuBT, EmoVoice, WiiGLE) have been successfully integrated to the system. Currently, SSI supports recording and streaming from multi-channel audio interfaces, web/dv cameras, Nintendo's Wii remote control, Microsoft's Kinect and various physiological sensors.

[2] http://www.praat.org

[3] http://www.mathworks.com/products/matlab/

[4] http://www.gnu.org/software/octave/

3. Online Recognition Pipeline

Let us illustrate the functions of the SSI framework by means of a concrete example. In order to build an emotional speech recognizer we start from an audio source, e. g. a microphone integrated in a headset. A fundamental property of human voice called pitch is the number of vibrations of the vocal cords per second. In order to detect voice activity, we connect the audio source with a pitch transformer and plug it to an activity-detector (AD). The AD will raise an event to all connected components as soon as speech activity is detected. As a handy side effect, pitch is also a well-known indicator of emotional speech. Since the buffer mechanism of the run-time engine allows several components to share the same signal source, we can directly forward the pitch signal to a (at this stage pre-trained) classifier, e. g. a Support Vector Machine (SVM). To bring the detected pitch segments into the compact fixed-length format needed by the SVM, we place another transformer (StatFeat) to apply a number of statistical functions, such as mean, standard deviation, range, etc. Figure 1 shows a scheme of this basic recognition engine.

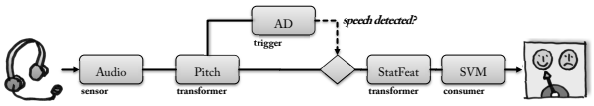

Figure 1: Basic Pipeline with Pre-Trained Classification

As mentioned earlier, recognition needs training. A small extension allows us to enhance the previous pipeline with the possibility to collect data. We simply plug in a component that stores the processed streams to disk. Several alternatives are possible. We could store the features of each detected speech segment right before they are passed to the classifier, but this approach has two obvious disadvantages: It is neither possible to replay the collected samples (e. g. for annotation purpose), nor to change the applied feature extraction at a later stage. Hence, it is obviously preferable to store raw audio instead. SSI offers tailored tools to apply a pipeline (or parts of a pipeline) to offline data. In this way we can simulate the transformations in the pipeline during the training of the classifier. Furthermore several feature selection algorithms and over-sampling techniques (for under represented classes) are part of the SSI framework. The modified pipeline is illustrated in Figure 2.

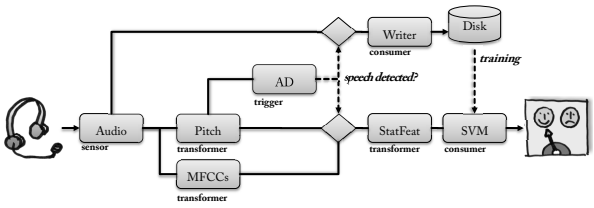

Figure 2: Extended Pipeline with Data Storage

Note that apart from the *writer* component, another transformation block has been added to the example. It calculates the Mel-frequency cepstral coefficients (MFCCs) and therefore makes more features available for classification. In this manner,

a pipeline can be stepwise improved with new filter and feature algorithms[5].

It is also possible to extend the system with further modalities. In this example we add a camera to analyse facial expressions of the user in addition to the vocal observations. In the optimal case, a new modality will carry complementary information and improve the robustness of the recognition. To do so, we set up a second pipeline by placing a camera sensor to which we connect a face detector (FaceDetec). For each image frame in which a face is found, we extract position features related to a person's mimic, such as position of eye browns or corners of the mouth[6] (see Figure 3).

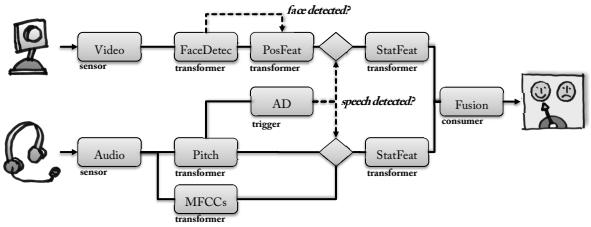

Figure 3: Multimodal Pipeline

In order to combine audio and visual information to a single decision SSI offers several alternatives. A straightforward way would be to merge the features calculated from each modality to a high dimensional, universal feature set and use it to train a single classifier (feature level fusion). Alternatively, separate classifiers can be trained, which outputs are combined by means of decision level fusion. SSI implements various combination rules, including weighted and unweighted voting schemes, algebraic combiners of continuous classifier outputs, specialist selection algorithms, constructed lookup tables, etc. Furthermore, schemes for meta level fusion, where model results are declared as meta data and used for the training of meta classifiers, are available as well.

4. Model Training

So far, no general purpose databases for emotion recognition exist. This is due to a number of reasons: in contrast to speech recognition, which is based on a fixed set of phonemes, it is difficult if not impossible to find a common sense on emotional states to be included in such a database. Dimensional models like Mehrabian's PAD (Pleasure, Arousal, Dominance) model [8] could be an alternate approach, e. g. by training discrete regions in the model and afterwards interpolating over the remaining area. But even then a number of other factors such as recording quality, background noise, user groups (gender, age, etc.), and culture differences have to be considered and make it hard to find a general solution. Available databases with emotional speech, for instance, have been recorded under rather specific conditions and cover only a limited set of emotions. Especially non-prototypical emotions are usually under-represented. As a result, databases like the Berlin Emotional Speech Database [9] or FAU Aibo Emotion Corpus [10] are well suited for testing the performance of an emotional speech recognizer, but will not necessarily produce a model that fits specific requirements in terms of emotional states and environmental factors.

These reasons often force developers to collect their own databases. Since most recognition systems are based on machine learning methods this usually requires a considerable amount of emotional data. Furthermore, to ensure satisfying recognition rates, it is decisive that the conditions during training are similar to the conditions that can be expected for the final system. In the best case the training data will be recorded from users interacting with the actual application. However, our experiences have shown that this is often too time consuming, especially since it requires thorough annotation by experts [7]. Instead, application developers should be provided with a tool that allows them to create databases on their own, preferably in a simple and fast fashion. As mentioned earlier two scenarios are possible: creating a user-independent model from recordings of a group of representative users, or building personalized user-depend model for each user who intends to use the application. To allow for both options we have developed a graphical user interface (GUI) that is simple enough to enable even novice users to complete all necessary steps to obtain their own model, but at the same time has the option to conduct and manage recordings of multiple users to train user-independent models.

The GUI is actually a wrapper for a pipeline implemented with the SSI framework (see Section 3). Since the pipeline overtakes all tasks related to signal processing (data recording, segmentation, feature extraction, classification), the GUI is not restricted to a certain problem, but offers a generic interface to conduct all kind of recognition tasks. For instance, an emotional speech classifier can be trained just as well as a gesture recognizer. Figure 4 includes a screenshot of the GUI showing a video and an audio track together with its annotation line.

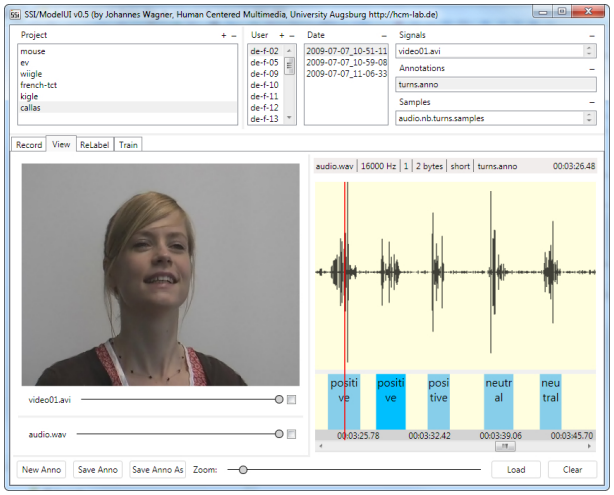

Figure 4: SSI/ModelUI offers a simple and generic way to record training data and build user-dependent or user-independent models out of it. Different recognition tasks including multiple users and sessions can be managed (top layer). In the screenshot a recorded session with a video and audio stream is reviewed. Below the audio signal an annotation track has been loaded for editing.

During a recording, stimuli are presented to the user as a series of html pages. The pages may include textual instructions, but also images or videos. Developers and users are free to change the content according to their own emotional expe-

[5] In addition to pitch and MFCCs, SSI offers components for extracting features from energy, duration, voicing and voice quality [7].

[6] A tool that can be used for this task and has already been integrated into the SSI framework is the the SHORE library developed by Fraunhofer: `http://www.iis.fraunhofer.de/en/bf/bv/ks/gpe/`.

riences. During the recording actions detected by the pipeline (e. g. speech segments) are reported back to the GUI, which may react by jumping to the next stimuli. At the same time, on- and offset of the actions are stored on disk and related to the label of the current stimulus. In this way an annotations are automatically created for each signal. To revise annotations (e. g. if the reaction of the user differs from the intended emotion or actions were falsely detected or missed) they can be visualized together with the recorded signal streams and edited. It is also possible to create new annotations by highlighting an area within a signal and convert it to a new segment in the annotation. If videos were recorded, they can be replayed to gain additional information and hints for annotation. Finally, available feature extraction and classification algorithms can be evaluated in terms of recognition performance. Depending on whether a user-dependent or user-independent model is required, it is possible to select data of a single user or of several user for training. Once a model is trained, it can be immediately tested on live input.

5. Practical Applications

SSI is being used in the EU-funded Metabo project for real-time physiological data analysis of diabetes patients in an automotive environment. Our EmoVoice component [11] as part of the SSI framework, was used in a various number of showcases in the European projects CALLAS and IRIS to analysis the expressivity in user speech. Examples are the E-Tree [12], which is an Augmented Reality art installation of a virtual tree that grows, shrinks, changes colours, etc. by interpreting affective multimodal input from video, keywords and emotional voice tone. In the virtual storytelling application EmoEmma [13], a user can influence the outcome of the story by acting as one of the characters. Furthermore, SSI was employed to construct the CALLAS expressivity corpus [14]. It consists of synchronized recordings from two high-quality cameras, an USB microphone, two Nintendos Wii remote controls and a data glove. Using the graphical interface described in Section 4 it became possible to repeat the same experiment in three countries, namely Germany, Greece and Italy. In total, about 15h of interaction from more than 50 subjects was collected. Within the AVLaughterCycle project, which aims at developing an audiovisual laughing machine, capable of recording the laughter of a user and to respond to it, SSI is used for recording and real-time processing [15].

6. Conclusion

We have introduced our Social Signal Interpretation (SSI) toolbox, a framework for the rapid development of on- or offline signal processing and recognition systems. The functionality of SSI was illustrated by means of an affective speech recognition pipeline, which is stepwise extended to a multimodal recognition task. Most recognition systems are based on machine learning methods, which require a considerable amount of data for training. With this in view, we have also talked about the graphical front-end for SSI that lets novice users train their own model. The source code of the SSI framework and the GUI tools are freely available under LGPL[7].

7. Acknowledgements

The work described in this paper is funded by the EU under research grant CALLAS (IST-34800), CEEDS (FP7-ICT-2009-5) and the IRIS Network of Excellence (Reference: 231824).

[7] `http://hcm-lab.de/ssi.html`

8. References

[1] J. Wagner, E. André, and F. Jung, "Smart sensor integration: A framework for multimodal emotion recognition in real-time," in *Affective Computing and Intelligent Interaction (ACII 2009)*. IEEE, 2009.

[2] M. Puckette, "Pure data: another integrated computer music environment," in *Proc. of the Second Intercollege Computer Music Concerts*, 1996, pp. 37–41.

[3] A. Camurri, P. Coletta, G. Varni, and S. Ghisio, "Developing multimodal interactive systems with eyesweb xmi," in *NIME '07: Proc. of the 7th international conference on New interfaces for musical expression*. New York, NY, USA: ACM, 2007, pp. 305–308.

[4] M. Serrano, L. Nigay, J.-Y. L. Lawson, A. Ramsay, R. Murray-Smith, and S. Denef, "The openinterface framework: a tool for multimodal interaction." in *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM, 2008, pp. 3501–3506.

[5] F. Eyben, M. Wöllmer, and B. Schuller, "openear - introducing the munich open-source emotion and affect recognition toolkit," in *2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*. IEEE, 2009, pp. 1–6.

[6] ——, "Opensmile: the munich versatile and fast open-source audio feature extractor," in *Proceedings of the international conference on Multimedia*, ser. MM '10. New York, NY, USA: ACM, 2010, pp. 1459–1462.

[7] T. Vogt, E. André, J. Wagner, S. Gilroy, F. Charles, and M. Cavazza, "Real-time vocal emotion recognition in artistic installations and interactive storytelling: Experiences and lessons learnt from callas and iris," in *Proceedings of the International Conference on Affective Computing and Intelligent Interaction (ACII)*, Amsterdam, The Netherlands, September 2009.

[8] A. Mehrabian, "Framework for a comprehensive description and measurement of emotional states." *Genetic, social, and general psychology monographs*, vol. 121, no. 3, pp. 339–361, August 1995.

[9] F. Burkhardt, A. Paeschke, M. Rolfes, W. Sendlmeier, and B. Weiss, "A database of german emotional speech," in *in Proceedings of Interspeech, Lissabon*, 2005, pp. 1517–1520.

[10] S. Steidl, *Automatic Classification of Emotion-Related User States in Spontaneous Childrens Speech*. Logos Verlag, Berlin, 2009.

[11] T. Vogt, E. André, and N. Bee, "Emovoice - a framework for online recognitionof emotions from voice," in *Proceedings of Workshop on Perception and Interactive Technologies for Speech-Based Systems*. Kloster Irsee, Germany: Springer, June 2008.

[12] S. W. Gilroy, M. Cavazza, R. Chaignon, S.-M. Mäkelä, M. Niranen, E. André, T. Vogt, J. Urbain, H. Seichter, M. Billinghurst, and M. Benayoun, "An affective model of user experience for interactive art," in *ACE '08: Proc. of the 2008 International Conference on Advances in Computer Entertainment Technology*. New York, NY, USA: ACM, 2008, pp. 107–110.

[13] F. Charles, D. Pizzi, M. Cavazza, T. Vogt, and E. André, "Emotional input for character-based interactive storytelling," in *The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Budapest, Hungary, 2009.

[14] G. Caridakis, J. Wagner, A. Raouzaiou, Z. Curto, E. André, and K. Karpouzis, "A multimodal corpus for gesture expressivity analysis." Multimodal Corpora: Advances in Capturing, Coding and Analyzing Multimodality, LREC, Malta, May 17-23, 2010, 2010.

[15] J. Urbain, R. Niewiadomski, E. Bevacqua, T. Dutoit, A. Moinet, C. Pelachaud, B. Picart, J. Tilmanne, and J. Wagner, "Avlaughtercycle," *Journal on Multimodal User Interfaces*, vol. 4, pp. 47–58, 2010.