SEMAINE

THE SENSITIVE AGENT PROJECT

D2a

Improved face and voice feature extraction with speaker adaptation and learning

Date: 22 December 2009

Dissemination level: Public

ICT project contract no.	211486
Project title	**SEMAINE** **Sustained Emotionally coloured Machine-human Interaction using Nonverbal Expression**
Contractual date of delivery	*31 December 2009*
Actual date of delivery	*22 December 2009*
Deliverable number	D2a
Deliverable title	Improved face and voice feature extraction with speaker adaptation and learning
Type	Demonstrator
Number of pages	17
WP contributing to the deliverable	WP 2
Responsible for task	Björn Schuller (schuller@tum.de)
Author(s)	Florian Eyben (eyben@tum.de), Hatice Gunes (h.gunes@imperial.ac.uk), Maja Pantic (m.pantic@imperial.ac.uk), Björn Schuller (schuller@tum.de), Michel Valstar (michel.valstar@imperial.ac.uk),Martin Wöllmer (woellmer@tum.de)
EC Project Officer	Philippe Gelin

Table of Contents

1 Executive Summary

Sensitive Artificial Listeners (SAL) are virtual dialogue partners who, despite their very limited verbal understanding, intend to engage the user in a conversation by paying attention to the user's emotions and non-verbal expressions. The SAL characters have their own emotionally defined personality, and attempt to drag the user towards their dominant emotion, through a combination of verbal and non-verbal expression.

The SEMAINE system 2.0 is the first public demonstrator of the fully operational autonomous SAL system based on audiovisual analysis and synthesis. The present report is part of a group of reports describing various aspects of the SEMAINE system 2.0. The full list of reports is available from http://semaine.opendfki.de/wiki/SEMAINE-2.0.

This report describes the current state of the face and voice feature extraction integrated into the SEMAINE system 2.0, namely, incremental audio processing, voice activity detection with speaker adaptation, incremental keyword-spotting with enhanced flexibility, speaker adaptation on the feature level, audio recording and logging of features, face detection, facial point detection and tracking, and global head motion estimation.

2 System description

2.1 Voice features

Incremental feature extraction

The openSMILE feature extractor (Eyben et al., 2009) is used to extract low-level audio features for incremental on-line affect analysis. Features extracted with openSMILE have already been verified in other recently published work, e.g. (Schuller et al., 2009b; Woellmer et al. 2009b). The openSMILE component is part of the TumFeatureExtractor Semaine component. It sends the low-level features fundamental frequency, probability of voicing, and signal energy to the topic *analysis.features.voice*. All other features, such as Mel-Frequency Cepstral Coefficients, Linear Predictive Coding Coefficients, and functionals applied to low-level descriptors are used with the TumFeatureExtractor component for classification of affect and gender. See D3b for details. The next paragraphs describe the architecture of the openSMILE feature extraction core.

Figure 1 illustrates the architecture of openSMILE, which is designed to be highly modular. Components can roughly be divided into three categories: sources (which produce data, i.e. the input side), processors (which process data, i.e. do the actual computations), and sinks (which write data to external channels, i.e. the output side). All components have access to a central data memory, which offers efficient storage buffers (ring-buffer support) for data exchange between components and access to data history. This allows for great flexibility and easy addition of new or modified components without the need to change existing functionality. The components' internal configuration and the interaction between components can be configured via a single text-based configuration file. Thus, feature sets can easily be reproduced by researchers all around the world and feature configurations are very likely to remain functional and compatible with future versions of openSMILE. The same binary can be used for an on-line live recognition system and for an off-line batch feature extractor.

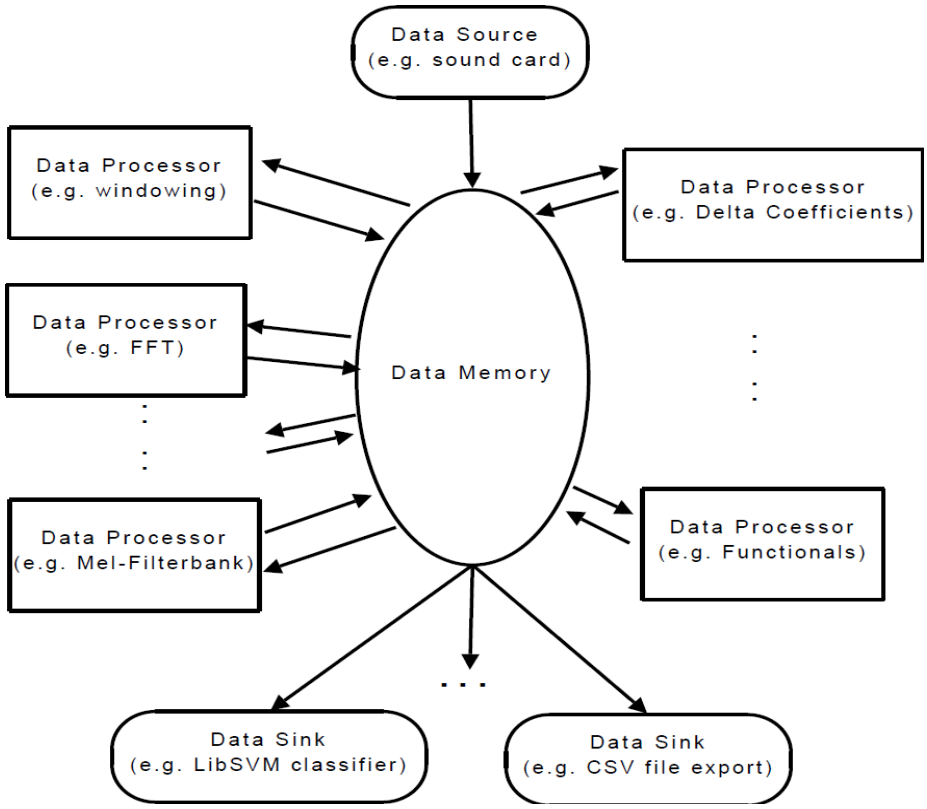

Figure 1: Architecture of openSMILE.

Besides exchanging data via the data memory, components in openSMILE can exchange data via a very simple messaging system, which is useful e.g. for classification results and turn start/end messages.

Each individual feature extractor (e.g. energy, spectrum, pitch, etc.) is implemented as one or more individual components. A full list of currently implemented components is found in Table 1. This list is sorted by component type: input/output components read/write data to/from files or devices, low-level feature extractors and signal processing components, filters and pre-/post-processing components, functional components which implement various statistical and transformation functionals which are used to summarise data, classifier output components, which classify/interpret data and output a result, and other components which do not fit in one of these categories.

Component	Function
I/O components	
cWaveSource	Reads audio samples from uncompressed RIFF-PCM files (WAVE). Any possible sampling rate is supported, 8, 16, 24, and 32-bit (integer and float) sample formats are supported as well as mono, and stereo files and files with an arbitrary number of channels. Mixing down of a multi-channel file to a single channel is also supported.
cWaveSink	Writes data to uncompressed RIFF-PCM files (WAVE). Any possible sampling rate is supported, 8, 16, 24, and 32-bit (integer and float) sample formats are supported as well as mono, and stereo files and files with an arbitrary number of channels.
cWaveSinkCut	Writes data to uncompressed RIFF-PCM files (WAVE). The output filename can be changed on the fly via openSMILE messages sent by other components. Any possible sampling rate is supported, 8, 16, 24, and 32-bit (integer and float) sample formats are

	supported as well as mono, and stereo files and files with an arbitrary number of channels.
cArffSource	Reads data from a WEKA Attribute-Relation feature file (ARFF). Only numeric attributes are currently supported. Each “instance” in the ARFF file is treated as one time frame. The attribute names are also read and used as data field names in openSMILE.
cArffSink	Writes data to a WEKA Attribute-Relation feature file (ARFF). Only numeric attributes are currently supported. Each “instance” in the ARFF file resembles one time frame. The attributes frameTime and frameIndex can be added optionally. Information about the target class for each instance can also be specified in the configuration file or passed via the command line.
cHtkSource	Reads data from binary HTK parameter files.
cHtkSink	Writes data to binary HTK parameter files.
cCsvSource	Reads data from a Comma Separated Value (CSV) file. Such files can be easily exported from popular spreadsheet applications. Each line thereby contains one frame, the columns represent features/fields. The first line may optionally contain the names of the features (data fields) separated by a delimiter character. The delimiter character can be configured and defaults to a comma.
cCsvSink	Writes data to a CSV file. Each line thereby contains one frame, the columns represent features/fields. The first line may optionally contain the names of the features (data fields) separated by a delimiter character. The delimiter character can be configured and defaults to a comma.
cDatadumpSink	Dumps data to a binary file representing a float (32 bit) matrix. A two value header (two 32 bit float values) is written to the beginning of the file. The first value specifies the vector (frame) size and the second value gives the number of vectors (frames).
cLibsvmSink	Data output in LibSVM data format (text). Support for appending target class labels to each instance is provided.
cNullSink	NULL output. This component discards all data it reads from the data memory. One can use this to avoid dead ends in the processing chain.
cPortaudioSource	Audio data (PCM) input (recording) from a local audio device (sound-card). The open-source library PortAudio is used for a platform independent interface to sound-card hardware. Recording capabilities depend on the underlying operating system and the sound-card hardware.
cPortaudioSink	Audio data (PCM) output (playback) to a local audio device (sound-card). The open-source library PortAudio is used for a platform independent interface to sound-card hardware. Playback capabilities depend on the underlying operating system and the sound-card hardware.
Pre-/post-processing and filtering components	
cFramer	Creates frames of values in a data contour (e.g. audio data). Thereby the frame size and the overlap can be specified in seconds or input samples/frames.
cPreemphasis	Applies a high pass filter for pre-emphasis of the speech signal to a data contour.
cVectorPreemphasis	Applies a high pass filter for pre-emphasis of the speech signal to a data frame. Thereby no context information is stored across frames. The last sample at the beginning of a frame is assumed to be zero (HTK implements this method, thus this component is provided for HTK compatibility).
cWindower	Applies a windowing function to a data frame. The following windowing functions are supported: Hanning, Hamming, Rectangular, Triangular, Gaussian, Sine, Blackman,

	Blackman-Harris, Bartlett, Bartlett-Hann, Lanczos.
cDeltaRegression	Computes regression coefficients of a data contour (HTK compliant). Connect multiple of these components in series to compute higher order regression coefficients.
cContourSmoother	Smoothes data contours using a simple moving average (SMA) filter with a configurable window length.
cVectorConcat	Concatenates vectors from multiple input levels to one vector in a third level.
cDataSelector	Selects data fields from input frame based on a list of field (=feature) names given via the component configuration.
cValbasedSelector	Selectively throws away or lets data frames pass, based on a threshold applied to one of the values in the input frame. This value can optionally be removed in the output frame.
cDbA	Applies psychoacoustic dB(A) weighting curve to the input vector. This only makes sense for magnitude or power spectrum as input.
cVectorOperation	Applies various static mathematical operations to the fields of a frame. These currently include scaling by a factor, adding an offset, and normalisation of the vector length to one.
cVectorMVN	On-line and off-line mean and variance normalisation. This component supports saving and loading of normalisation parameters, on-line updates of parameters using different update methods (exponential, fixed buffer, etc.).
cVectorHEQ	On-line and off-line histogram equalisation and mean normalisation. This component supports saving and loading of histogram and mean/variance parameters, on-line updates of parameters using different update methods (exponential, fixed buffer, etc.).
cBuffer	The “NULL” processor: it reads data from one level of the data memory and writes it to another level without modification. The target level, e.g. may have a different buffersize, etc.
Low-level signal processing and feature extraction	
cEnergy	Computes Root-Mean-Square (RMS) energy and logarithmic energy (HTK compatible).
cIntensity	Computes sound level intensity and an approximation of the psychoacoustic loudness (*not* frequency selective).
cMzcr	Computes zero-crossing rate, mean crossing rate (rate at which the signal crosses its mean value), maximum value and minimum value of input data contour.
cTransformFFT	Implements a fast Fourier transformation. Input is a real valued vector and output is a vector in the complex domain.
cFFTmagphase	From the complex output of the cTransformFFT component this component can compute the magnitude spectrum and the phase spectrum.
cLpc	Computes linear Predictive Coefficients (LPC) of configurable order p. The output of reflection coefficients is also supported.
cLsp	Computes line spectral pairs (line spectral frequencies, LSF) from LPC coefficient data.
cMelspec	Computes a Mel- or Bark-spectrum from an FFT magnitude or power spectrum. The number of bands and the frequency range is configurable.
cMfcc	Computes Mel-frequency cepstral coefficients from a Mel-spectrum.
cAcf	Computes the autocorrelation function (ACF) or the Cepstrum of the input data frame. (Note that Cepstrum computation is similar to ACF computation. The only difference

	is the logarithm which is applied in the frequency domain. Thus, these two things were united in one component.)
cAmdf	Computes the auto Average-Magnitude difference function of the input frame.
cPitchACF	Computes voice fundamental frequency, the probability of voicing, and a Harmonics to Noise ratio (HNR). Thereby a combined autocorrelation/cepstrum approach is used. Thus, this component requires acf *and* cepstrum as input data.
cSpectral	Spectral features: arbitrary user-defined spectral roll-off points, energy in user-defined spectral bands, spectral flux, spectral centroid, position of spectral maximum, position of spectral minimum, spectral entropy.
cTonespec	Computes a semi-tone bin spectrum from an FFT magnitude or power spectrum having a sufficiently high resolution.
cTonefilt	Continuously computes a semi-tone bin spectrum from PCM frames using a simple correlation method. No FFT is used and the reference samples used for the correlation are exactly tuned to semitones. This component is slower than cTonespec, it may, however, in some cases produce cleaner results.
cChroma	This component implements a warping of a semi-tone spectrum to one single octave (Chroma features). The output of cChroma is always a k*12 dimensional vector, whereby k>=1.
cChromaFeatures	Various statistics and derived features based on Chroma output. These features were derived from expert knowledge and music theory.
cFunctionals	Meta-component for applying functionals to data contours to map contours of variable length to a fixed set of descriptors. The next section of this table describes the actual computation components which can be instantiated by this meta-component.
Functionals (all component names in this section begin with "cFunctional". It is omitted here.)	
Means	Various means: arithmetic, quadratic, and mean of absolute values. It is also supported to compute the mean only of all non-zero values. In this case also geometric mean and the number of non-zero inputs can be computed.
Moments	The statistical moments variance, standard deviation, kurtosis, and skewness.
Extremes	Extreme values: maximum value, minimum value, range (max-min), relative position of maximum and minimum value, (arithmetic mean), distance of maximum value to arithmetic mean, distance of minimum value to arithmetic mean.
Precentiles	Quartiles, inter-quartile ranges, unlimited user defined percentiles and unlimited user defined inter-percentile ranges among all user defined percentiles.
Regression	Linear and quadratic regression coefficients, corresponding linear and quadratic approximation error for each set of coefficients, and data contour centroid.
Peaks	Number of peaks (local maxima) found in the data contour, mean distance between peaks (normalised to input segment length), mean of all peaks, and distance of arithmetic mean to the mean of all peaks.
Segments	Number of segments based on delta thresholding, average segment length, minimum segment length, maximum segment length.
Onsets	Position of first onset in the input (i.e. the first time the contour takes a value higher than a configurable threshold), position of the last offset, number of onsets and offsets in total.
Crossings	Zero-crossing and mean-crossing rate
Discrete Cosine Transf.	User-definable number of Discrete Cosine Transform-II coefficients

Times	Relative amounts/times the input signal is above or below a relative threshold (up-level-time, down-level-time), rising or falling, or curved left or right. Total duration of the input in frames or seconds.
Classifiers / Interpreters	
cLibsvmliveSink	Provides a wrapper to the LibSVM library for Support Vector classification and regression. An extension to support saving and loading of binary model files for LibSVM has been developed for openSMILE. This significantly speeds up loading of large model files.
cTumkwsjSink	Interface to Julius Large Vocabulary Continuous Speech Recognition Engine for classification via Hidden Markov Models.
cPitchDirection	Estimates syllable nuclei and speaking rate from pitch data. Based on the syllable position an estimate of pitch direction is provided for each syllable. This estimate is restricted to the following classes: flat, rise, fall, rise-fall, fall-rise.
Other	
cVadV1	Heuristic and rule-based voice activity detection. See the next section for details.
cSemaineSpeakerID1	NN learner of user voice pattern and background noise. This learner adapts its models on-line based on its own output. An initial training phase is required using voice activity output from a rule-based module (e.g. cVadV1)
cTurnDetector	Currently simple turn detection module which decides whether the user has started / ended his turn based on the length of the voiced or unvoiced segments.
cBowProducer	Produces bag-of-words frames which can be combined with acoustic feature vectors. The bag-of-word frames are produced from result messages received from the cTumkwsjSink component. Output from the cTurnDetector component is required for synchronisation with acoustic feature vectors (usually produced by the cFunctionals component)
semaineEmmaSender	This component sends EMMA messages to the other components in the SEMAINE system. These messages include pitch direction, gender, interest level, and five dimensions of affect (arousal, valence, power, expectation, intensity).

Table 1: Currently implemented components

Voice activity detection with speaker adaptation

The voice activity detection (VAD) module uses two algorithms to robustly detect voice activity even in noisy environments. First, an adaptive rule-based voice activity detector is used. This detector analyses the low-level descriptors logarithmic energy, the line spectral pair frequencies, and the Mel-spectrum. A deviation factor from the line spectral pair frequencies (lsp), and the entropy of the Mel-spectrum is computed. Thus, three parameters remain upon which the VAD is built: energy, lsp deviation factor, and Mel-spectral entropy. The lsp deviation computation is based on the assumption that for unvoiced signals and silence the line spectral frequencies are distributed equally among the unit circle, i.e. distributed equally over the range from 0 to Pi. For voiced segments one or more lsp frequencies differ from their expected position. Thus, the deviation factor is computed as the sum of squared differences between the actual and the expected positions.

The raw contours of the three parameters are smoothed using an asymmetric first order exponential filter. Assuming a 10 ms frame rate, the smoothing factor alpha is 0.0488 for rising signals (i.e. the current value is greater than the last filtered value) and 0.3935 for falling signals.

The actual voice activity detection is performed using a fuzzy logic. For each parameter five thresholds are computed as the parameter's current mean plus/minus N times the parameter's standard deviation (these statistics are initialised from the first second of input data and then continuously updated). If the current parameter value is higher than a threshold N, the fuzzy score for this parameter is set to N*0.2. For the final decision a weighted sum of the three individual fuzzy scores is computed, exponentially smoothed, and a threshold of 0.5 is applied for the final binary voicing decision.

The binary output of the rule based voice activity detection is used to train a model of the speaker's voice and the background noise. Mel-Frequency cepstral coefficients are used as input features to the model, which is a simple nearest neighbour model containing the mean values of the input features for each class. An unknown input is assigned the class with the smallest distance between the class' mean vector and the input. This simple classifier allows for very fast and efficient on-line model updates and on-line classification.

After the two models for background noise and speaker voice have been trained with data of at least 5 seconds each, the system stops using the rule-based voice activity input. The classifier output is now used in a feedback loop as training label for new data, and the classifier output replaces the rule-based VAD output as the final voicing decision.

The final voicing decision changes from 0 to 1 and remains at 1 for at least 2 frames, a "speaking" message is sent to *state.user.emma* with the attribute "statusChange" equal to "start". When the voicing decision drops from 1 to 0 and remains at 0 for at least 2 frames a "speaking" message with "statusChange" equal to "stop" is sent. The VAD module is part of the openSMILE framework and thus currently part of the SEMAINE component TumFeatureExtractor. This is likely to change in future releases.

Incremental keyword-spotting

Since the SEMAINE system will have to face a large number of out-of-vocabulary words and since the dialogue management component only uses single keywords within the recognised text string, the speech recognition module developed in the first project year had been replaced by a phoneme-based keyword spotter that is now integrated into openSMILE (Eyben et al., 2009). In order to obtain universal models, a number of different speech corpora were used for training: the SAL corpus, the SEMAINE database, the Wall Street Journal corpus, as well as the AMIDA and the AVIC database. The SAL, SEMAINE, AMIDA, and AVIC databases contain spontaneous and partly emotionally coloured speech with a wide range of different speaking styles and dialects, which is important for robust phoneme models in the context of emotional speech (Steidl et al., 2009). Furthermore they contain non-linguistic vocalisations, which are essential in order to include the respective models into speech decoding.

Using these speech corpora, tied-state triphone models (based on a set of 39 monophones) were trained. Thereby 13 cepstral mean normalised MFCC features were used extracted via openSMILE, together with first and second order regression coefficients. Features were extracted at a common frame rate of 10ms applying a 25ms Hamming window. All phoneme Hidden Markov Models consist of three states with eight Gaussian mixtures each. To enable a faster model loading during the start of the keyword spotter component, models are saved in a binary format.

As system responses have to be prepared already before the user has finished speaking, the keyword spotter operates incrementally, meaning that the current best guess of the keywords contained in the utterance spoken so far is output and updated at a constant rate (the default rate is 600ms).

The applied speech decoder uses the Julius library which supports two-pass decoding. Thus, it processes the speech feature vector sequence in forward and in backward direction and can output a refined final hypothesis once the complete speech turn is available (i.e. once a silence period is detected). For every keyword contained in the final hypothesis a confidence in the interval from 0 to 1 is output, together with the exact keyword timings.

Currently, the dialogue management of the SEMAINE system uses 140 different keywords, whereas the keyword detector supports multiple pronunciation variants of the individual words. An additional set of keywords is used as a bag-of-word feature set for combined acoustic-linguistic affect recognition (see deliverable D3b).

The detected keywords are sent to the semaine topic *state.user.emma* including keyword start times and confidences. As detailed in D3b, the keywords are used to build bag-of-words feature vectors for linguistic affect recognition for the dimensions arousal and valence.

Enhanced flexibility of keyword recognition module

Compared to keyword detectors based on whole-word models, the keyword spotter as applied in the SEMAINE system offers the great advantage that new keywords can be added without having to retrain the system. Since the keyword spotting module uses vocabulary independent triphone models, it can be quickly adapted to support additional keywords by simply adding the corresponding words together with their pronunciations in the dictionary (Woellmer et al., 2009). Furthermore, it can be trained on any speech corpus, regardless of whether or not the keywords occur in the training corpus. Single phonemes and phoneme strings corresponding to keywords are modelled simultaneously during decoding. Thus, training an explicit "garbage model" on speech that does not contain any keywords is not necessary. Generally, using explicit garbage HMMs is problematic since a garbage model can potentially represent any phoneme sequence – including the keywords. Thus, phone-based approaches tend to outperform systems that use trained garbage models (Keshet et al., 2009).

The "aggressiveness" of the keyword search can be adjusted via a parameter that controls the a priori probability of keywords and "garbage speech". Consequently, the operating point of the Receiver Operating Characteristic (ROC) curve can be varied (trade-off between high true positive rate and low false positive rate).

Depending on how often a new hypothesis of contained keywords is required, the time interval for re-estimation of the progressive word hypothesis output can be configured. Also the trade-off between stability of results and amount of required memory can be adjusted by changing the stack size, meaning the number of hypotheses which can be stored on a stack during decoding.

Speaker adaptation on the feature level

In order to compensate for individual speaker characteristics and differences in the acoustic environment and recording setting, all features (besides those used for the keyword spotter) are normalised to zero mean and unit variance. When training the models this is done on a per speaker base, i.e. mean and variance are computed from all data of one speaker, and used to normalise data from this speaker only. For on-line recognition initial mean and variance values are computed using all available training data. During on-line recognition these initial mean and variance values are updated incrementally as new data arrives, thus building a "model" of the current user. Currently a change of user is not supported, as the means and variances are not reset.

For the MFCC features used for keyword spotting, histogram equalisation (HEQ) is applied to improve noise robustness (Schuller et al., 2009). HEQ hereby replaces the mean and variance normalisation. In short, for HEQ a histogram reflecting the distribution of the values for each feature is computed. The values are then transformed piecewise linear to have Gaussian distribution with a standard deviation of one and mean of zero. During training the histogram parameters are computed from each training turn individually. In the live recogniser, the histogram is computed for data of the last five seconds at every frame.

Audio recording / logging of features

The openSMILE audio input module supports recording of the full audio input stream as well as the turn segments (as detected by the voice activity detection) to uncompressed PCM WAVE files. This is essential for checking recording quality and functionality of the voice activity detector in a live set-up. It is also a great feature for developers who are modifying, testing, and debugging the voice input component. Additionally all on-line extracted features (low-level contours and functionals) can be dumped to common feature file formats such as WEKA Arff, LibSVM format, Comma Separated Value (CSV) files, and HTK parameter files.

2.2 Face features

Face Detection

The face detector implemented in SEMAINE uses the OpenCV implementation of the Viola and Jones face detector as its basis. To ensure that the face detector component takes as little time as possible, we constrain the search space based on the maximum expected velocity of the head in a conversational scenario. Because we know the size of the detected face area in the current frame, the maximum expected velocity towards and away from the camera places an upper and a lower bound on the size of the face in the next frame. Similarly, the maximum lateral and vertical velocity of the head with respect to the camera determine the maximum search area.

Within the SEMAINE framework, the face detector is part of the VideoFeatureExtractor module. It sends information about the detected face location (if any) to the topic analysis.features.video.face-detection, with a frequency equal to the rate of frame capture.

Facial Point Detection

We have developed and implemented a novel facial point detector that detects 20 fiducial facial points and the pupils in a near-frontal face (see Figure 2). The method, coined BoRMaN, employs Boosted Regression (BoR) to predict the location of a target point T relative to a location L, given a set of dense appearance descriptors extracted from the immediate surroundings of L. Thus, each point in the neighbourhood of the target location provides a prediction of where the target point is, which means we only need to try a small number of locations in order to get an accurate prediction of T. This is much faster than traditional search by a sliding window approach. In the latter approach, every single location in the target location's neighbourhood would have to be tested to de-

termine whether it was the target location or not. The regressors we used were Support Vector Regressors. They were trained using the most informative Haar-like features, as selected by an implementation of Drucker's AdaBoost regressor.

The search space is constrained by using Markov Networks (MaN) to model the spatial relations between pairs of points. Points are detected in groups, which are defined as sets of points that have strong spatial constraints on each other even in the presence of facial expressions. For each group of points we have trained a separate MaN. In our implementation of a MaN, every node is a relation between two points. It can thus be seen as the dual of the fully connected graph of the facial points in the group. Relations between nodes being modelled are their relative orientations and lengths.

The point detector is initialised based on the prior probability of a facial point location given the detected face location. This gives a good initial guess. Thus, in combination with a further reduction of the search space by the MaNs and the elimination of sliding window based search by the BoR, we have developed an extremely fast point detector.

The point detector has been specifically trained on the MMI Facial Expression database (Pantic et al., 2005) and the FERET database (Phillips et al., 1998) to be able to cope with subjects of varying ages, sex, and ethnicity, and is able to cope well with various facial expressions.

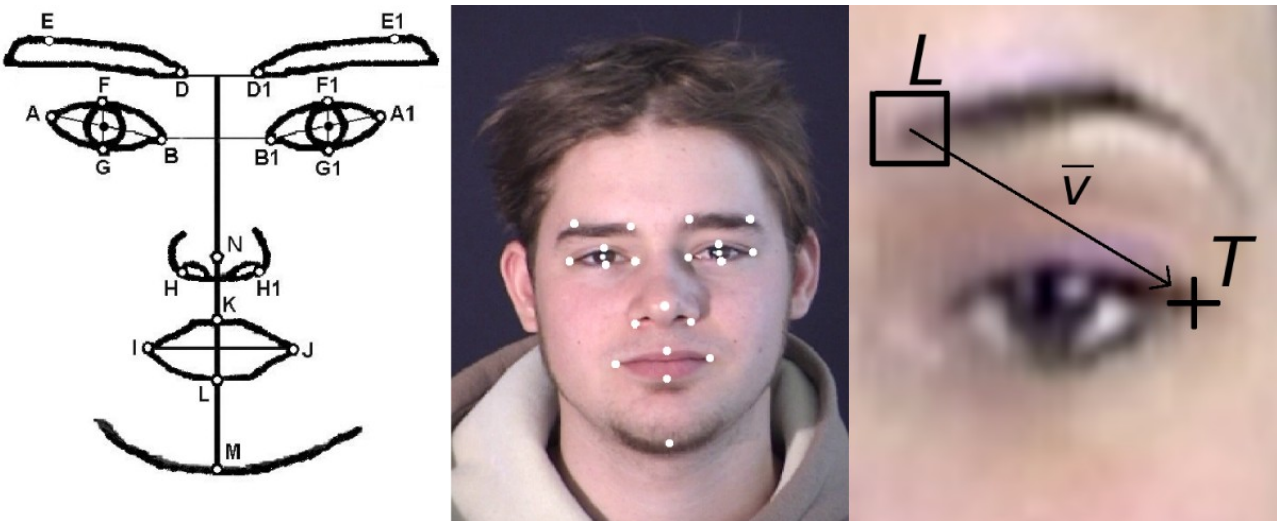

Figure 2: The 20 fiducial facial points detected by BoRMaN. The rightmost panel shows how a regressor should predict the vector v to arrive from a patch L to the target point location T.

The BoRMaN point detector has been implemented in the SEMAINE framework as part of the VideoFeatureExtractor module. It serves as the starting point for the facial point tracker (see next section).

Facial Point Tracking

The facial point tracker is initialised using the point locations returned by the facial point detector. To track facial points through all following frames, we use an adapted version of the point detector.

Because we know that the face of the person we're tracking does not change (i.e. we're tracking points of the same person), we can use the actual appearance of this subject's facial points to determine how good a predicted point location is. This allows us to create an accurate tracker.

Because we can use the point locations detected/tracked in the previous frame to initialise the search for the points in the current frame, the search space is significantly reduced and we need fewer iterations per point to achieve convergence.

The facial point tracker is implemented in the VideoFeatureExtractor module in the SEMAINE framework. Information about the tracked point locations is sent to the analysis.features.video.facialpoints topic with a frequency equal to the frame capture rate.

Global Head Motion Estimation

Global head motion estimation outputs the magnitude and the direction of the 2D head motion. This analysis is based on optical flow computation which is an intensity based method using pixels' local intensity variations to compute the movement velocities on the horizontal and vertical axes for a given number of images. Overall, it provides an approximation of the local image motion.

The optical flow computation is based on the following assumptions:

1. The changes in intensity are all due to the motion of the observed object (i.e., the face)
2. The observed object (i.e., the face) never gets occluded and the optical flow is due to the observed object.
3. The observed object (i.e., the face) is rigid, and the optical flow is due to the global movement of the object and not due to internal deformations.

In order to determine the magnitude and the direction of the 2D head motion, the optical flow is calculated between two consecutive frames. It is applied to a refined region (i.e., resized and smoothed) within the detected face area to ensure that the target region does not contain any background information. More specifically, an overall optical flow vector over the whole refined facial region is computed. Therefore, the resulting vector represents the global direction of the movement taking into account all pixels within that region. The angle and magnitude of the overall optical flow vector are calculated using the horizontal and vertical velocities.

The preliminary experiments have shown that mouth or eyes deformation do not have a significant impact on the movement direction computed using the refined facial region.

The 2D Global Head Motion Estimator is implemented in the VideoFeatureExtractor module of the SEMAINE framework. Information about the global head motion is sent to the analysis.features.video.2dheadmotion topic with a frequency equal to the frame capture rate. Features are only sent when a face has been detected previously.

3 License and availability

- voice feature extraction is available under the terms of the GPL within the SEMAINE system (included in SEMAINE download package; linux and windows versions available); the voice feature extraction module is also available to the research community as an open-source standalone library and command-line tool, called openSMILE
- keyword spotting is based on the open-source Julius engine, which is available as a third-party download under a BSD-style license
- facial feature extraction modules are all part of the VideoFeatureExtractor module, which is available as binary component within the SEMAINE system download package under the terms of a binary-only research licence

References

(Eyben et al., 2009) F. Eyben, M. Wöllmer, B. Schuller: openEAR - Introducing the Munich Open-Source Emotion and Affect Recognition Toolkit, in Proceedings of the 4th International HUMAINE Association Conference on Affective Computing and Intelligent Interaction 2009 (ACII 2009), Amsterdam, The Netherlands, pp. 576-581, 2009.

(Keshet et al., 2009) J. Keshet, D. Grangier, S. Bengio: Discriminative Keyword Spotting, Speech Communication, vol. 51(4): 317-329, 2009.

(Pantic et al., 2005) M. Pantic, M. Valstar, R. Rademaker, L. Maat: Web- based database for facial expression analysis, in Proc. of IEEE Int'l Conf. Multimedia and Expo (ICME'05), pp. 317-321, 2005.

(Phillips et al., 1998) P. Phillips, H. Wechsler, J. Huang, P. Rauss: The feret database and evaluation procedure for face-recognition algorithms, Image and Vision Computing Journal, 16(5): 295-306, 1998.

(Schuller et al., 2009) B. Schuller, M. Wöllmer, T. Moosmayr, G. Rigoll: Recognition of Noisy Speech: A Comparative Survey of Robust Model Architecture and Feature Enhancement, in Journal on Audio, Speech, and Music Processing, ID 942617, 2009.

(Schuller et al., 2009b) B. Schuller, S. Steidl, A. Batliner: The Interspeech 2009 Emotion Challenge, in Proc. of Interspeech, Brighton, UK, pp. 312-315, 2009.

(Steidl et al., 2009) S. Steidl, A. Batliner, D. Seppi, B. Schuller: On the Impact of Children's Emotional Speech on Acoustic and Language Models, to appear in EURASIP Journal on Audio, Speech, and Music Processing (JASMP), Special Issue on "Atypical Speech", 2009.

(Woellmer et al., 2009) M. Wöllmer, F. Eyben, B. Schuller, G. Rigoll: Robust vocabulary independent keyword spotting with graphical models, in Proc. of ASRU, Merano, Italy, 2009.

(Woellmer et al., 2009b) 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 Proc. of Interspeech, Brighton, UK, pp. 1595-1598, 2009.