SHELF COMPONENTS FIRST RELEASE

Conveying Affectiveness in Leading-edge Living Adaptive Systems

CALLAS

Project IST-34800

Deliverable D121 WP1.2

Deliverable
Version 1.0 – 31 October 2007
Document. ref.: callas.D121.UOA.WP1.2.V1.0

Programme Name: ..... IST
Project Number: 34800
Project Title:..... CALLAS
Partners:..... Coordinator: ENG (IT)
Contractors:
VTT Electronics, BBC, Metaware, Studio Azzurro, XIM, Digital Video, Humanware, Nexture, University of Augsburg, ICCS/NTUA, University of Mons, University of Teesside, Helsinki University of Technology, Paris 8, Scuola Normale Superiore di Pisa, University of Reading, Fondazione Teatro Massimo, HITLaboratory New Zealand

Document Number: callas.D121.UOA.WP1.2.V1.0
Work-Package:..... WP1.2
Deliverable Type: ..... Document
Contractual Date of Delivery: ..... 31 October 2007
Actual Date of Delivery: ..... 31 October 2007
Title of Document: ..... Shelf Components First Release
Author(s): ..... FPMS, HMW, ICCS, UOA and VTT

Approval of this report .....

Summary of this report:.....

History:.....

Keyword List: .....

Availability..... This report is: public

Table of Contents

Executive Summary

WP 1.2 is responsible for developing the shelf components within CALLAS that provide the technology for analyzing affective input. This deliverable gives a first overview about the first part of the components within the shelf.

In the first year, tasks 1.2.1 – 1.2 4 have been tackled. Each tasks resulted in two or three shelf components that are described in this deliverable. After the first year, CALLAS provides a rich repertoire of components that provide information that may be utilized to assess the user's emotional state. While previous work focused on offline recognition, CALLAS deals with the challenging task of online recognition. All shelf components developed within WP1.2 are able to provide information relevant to emotion recognition (features or emotion assessments in terms of categories or dimensions) in real-time while the user is interacting with an application. Most of the shelf components have been integrated in one or several showcases.

In section 1 we describe the work conducted in task 1.2.1 (Speech-Based Recognition of Emotions). It describes the components available for speech-based emotion recognition. The components consist of a keyword spotter, emotion recognition form acoustic signals and an emotion recognizer from text.

Section 2 describes the components developed within task 1.2.2 (Audio-Video Feature Extraction). This is separated into audio feature extraction and video feature extraction.

Section 3 describes the work part of task 1.2.3 (Gesture and Body Motion Tracking). There are three components available within this task: HumanGlove, gesture recognition from mobile phones and gesture expressivity recognition from video signal.

Section 4 is describing the components of task 1.2.4 (Facial Expression Recognition). It consists of two components. One for detecting facial features and the other for detecting gaze and estimating pose.

Each component gives a short description about its functionality, describes the usage, the requirements, status and availability.

1. Speech-based recognition of emotions

1.1 Keyword Spotting

1.1.1 Overview

Understanding what a user says can serve as input to drive a lot of applications. The component presented here aims at recognizing any spoken utterance inside a short vocabulary list. We describe hereunder the different processing parts of the component, explaining why the task is hard and conditions of utilization are restricted (small vocabulary list, rather quite environment), before giving a brief documentation about the speech recognition system, giving its requirements, status and availability.

1.1.2 Description

The task of Automatic Speech Recognition (ASR) is split into several modules. The classical architecture of an ASR system is given in Figure 1. The acoustical wave is measured and analyzed in order to extract the linguistic information and derive a sequence of words. The different blocks of this pipe-line architecture are presented hereunder.

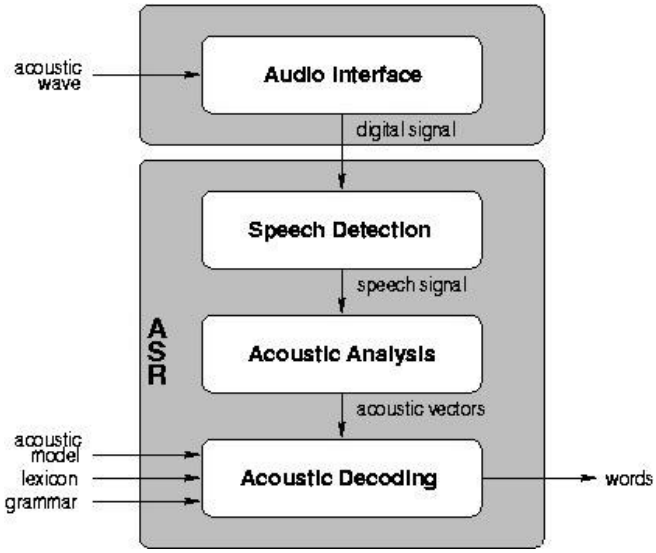


Figure 1: Classical architecture of an automatic speech recognition system

Audio Interface - An analog-to-digital conversion is applied here and the microphone signal is discretized both in time (sampling) and amplitude (quantification). A digital signal is finally obtained as a sequence of samples that give the amplitude of the microphone signal at discrete time instants and every sample amplitude is represented in its digital form, i.e. a form workable by the computer. The sampling frequency is typically related to the application under consideration and the operating platform (for example, 8000 Hertz for applications over telephone lines and 16000 Hertz for multimedia applications).

Speech Detection - This block aims at detecting the segments of speech activity in the digital signal. Only these segments that compose the speech signal are transmitted to the

following block. The purpose of the speech detection block is to limit the computational cost and avoid triggering excessively the ASR process when unexpected acoustic events happen. The function of this block is sometimes implemented manually: the speaker is asked to push a button while speaking in order to activate the ASR system (mode push-to-talk).

Acoustical Analysis - Speech is by nature highly variable. Even if the same words are pronounced by the same speaker, unlikely to measure two totally identical speech signals. Because of this variability, the ASR problem is extremely hard to solve. The goal of the acoustical analysis block is to process the speech signal in order to reduce the variability while preserving its linguistic information. A time-frequency analysis is typically performed. The speech signal is observed through a finite-length analysis window that is regularly shifted along the speech samples. Classically, the analysis window length and shift are set equal to 30 milliseconds and 10 milliseconds, respectively. For every location of the analysis window, the envelope of the spectrum (i.e. the distribution of energy across frequencies) of the observed speech samples is estimated. This estimation is concisely expressed as a vector of about 10 coefficients. Repeating the frequency analysis for every time location of the analysis window, we obtain a sequence of acoustic vectors that describe the time evolution of the spectral envelope of the speech signal. There exist many algorithms for computing acoustic vectors. They all aim at getting acoustic vectors that represent the linguistic information encoded in the speech signal and are as little sensitive as possible to non-linguistic variability sources such as the speaker identity, the acoustical environment (e.g., background noise) or the transmission channel (e.g., a telephone line or a low-quality microphone).

Acoustical Decoding – In order to recognize a word, the ASR system has to learn how the acoustic realizations of this word look like in terms of acoustic vector sequences. During a training phase, the ASR system is presented with several examples of every possible word, as defined by the lexicon. A statistical model is then computed for every word such that it models the distribution of the acoustic vectors. Repeating the estimation for all the words, we finally obtain a set of statistical models, the so-called acoustic model, which is stored in the ASR system for further use. However, word-based acoustic modeling becomes problematic as the number of words increases (>50 words). More especially, it becomes difficult or even unrealistic to gather the speech data that are required to properly train the acoustic model. It is generally preferred to use linguistic units that are shorter than words but in a limited number to completely describe the language. A classical choice is the phoneme. Most languages are entirely characterized by a few tens of phonemes. They define the elementary speech sounds that compose every word, every sentence. During the training phase, each phoneme is used separately to estimate its own statistical model and the word-based models can be obtained by concatenating the phoneme-based models. Such an approach requires knowing the phonetic transcription of every word, i.e. how to pronounce it in terms of phonemes. This information is contained in the lexicon that provides one or more phonetic transcriptions for every word. Nowadays, most common acoustic modeling technique is based on Hidden Markov Models (HMM).

The component we released does the whole pipeline illustrated in Figure 1. Let us specify now how the different blocks are implemented in this component:

- The audio interface was realized using Portaudio, which is an open-sours and cross-platform standard library for audio acquisition. Portaudio enables to perform the acquisition with standard functions quasi-independently from the installed hardware, and thus hopefully to be able to run the ASR on every computer. It provides easy functions to start and stop audio streams for recording, which is used in our case, or playing sounds. During the audio acquisition, a buffer is filled with the recorded samples. Each time the buffer is full, it is given to a “callback” function defined by the programmer, while the acquisition goes on (until the stream is stopped). This enables to perform real-time audio analysis: sound can be processed without having to wait for the stream to be stopped to start processing the input samples.

  In our case, the filled buffers are directly sent to the ASR system, called EAR. EAR is a component developed conjointly by MULTITEL ASBL and the Faculty of

Engineering, Mons, to use in runtime another software they developed to perform ASR, called STRUT (Speech Training and Recognition Unified Tool). ACAPELA GROUP[1] is marketing software products for automatic speech recognition that are based on STRUT and EAR. The speech detection is either performed through "pushing a button" (audio acquisition is started when a socket is received and stopped when a second socket arrives) or through an algorithm to separate speech from other sounds. The ASR task is more error-prone in the "automatic speech detection" mode than in the "push-to-talk" mode.

- The Acoustic Analysis block supports the most classical frequency analysis methods (MFCC, PLP, LPCC).
- The Acoustic Model is based on Hidden Markov Models to render the time evolution of the signals. The acoustic probability of each phoneme given an acoustic vector can be estimated by the use of Gaussian Mixtures or Multi-Layer Perceptron (MLP). In the implementation we released, HMMs are combined with MLP outputs. Of course, we also released a MLP thoroughly trained for English.

  The lexicon and grammar are transcribed in a JSGF (Java Speech Grammar Format) file gathering the list of allowable utterances and the phonetic transcriptions of each word. We used the assets of our components: we do not try to make a large-vocabulary speech recognition but we restrict, through the grammar, the recognizable utterances to a short list (around 50) of expressions. Indeed, for the targeted applications, it seemed us better to have a strong recognition system to identify utterances in a small list (related to a certain context) than trying to recognize anything in a large vocabulary, yielding in more recognition errors and the need for high-level post-processing to interpret what has been recognized, which would also introduce errors. In the package we released, there is thus a compiled grammar that was built for the Interactive TV showcase.

The speech recognition performed by this component is speaker-independent and "user real-time" (<1s).

1.1.3 Documentation

The first release of our component was developed according to the needs of the Interactive TV showcase. It is thus raw: it does all the sequence of tasks previously described and outputs the recognized words, but we did not yet integrate that into a graphical interface since the application did not need one. In consequence, there is very few documentation needed to explain how to use the component so far. We are currently developing a nicer-looking version in JAVA, which will include a graphical interface to command the speech recognition and will display graphics in real time to illustrate some of the computational steps applied to the initial sound wave during the recognition process.

The released component is commanded through UDP sockets on port 22556: a first socket must be sent on this port to start an audio stream and begin the ASR task, and a second socket will stop the stream and terminate the ASR. The recognized utterances are also outputted through UDP sockets, on port 22557. The output socket consists of the recognized words, a confidence level for each of them and a timestamp indicating when the recognition of the utterance was finished.

Depending on the context (driven by the recognized words themselves or any other input of the global application), it might be wished to change the list of recognizable expressions during the application. This can be done instantaneously with the released component, provided that the new grammar file was already defined before the application (with a JSGF file with the grammar and the phonetic transcriptions), otherwise it takes a little longer (and we did not release tools to do that since it is not needed for the targeted applications).

---

[1] http://www.acapela-group.com/

EAR has a lot of settings. For example, it is possible to output not only the most likely spoken utterance but the N most likely ones. There are also functions to get access to a lot of computational information (probabilities of each phoneme, time information), to facilitate vocabulary changes, etc. In this release, we compiled everything in a default mode, but if future CALLAS applications request it we will use more of the possibilities given by Ear in the next releases.

To run, the ASR needs to have access to all the information called by the Acoustic Decoder: the MLP for computing acoustic probabilities for each phoneme and the compiled grammar (+ phonetic transcriptions).

1.1.4 Requirements

Rather quiet environment: noise is a big problem for ASR. The released component is tolerant to a certain level of noise, but performance decrease when used in a noisy environment.

1.1.5 Status

The first version of the component was already released. We performed some tests in the lab and during the last CALLAS plenary meeting. The component is thus usable, but we are still developing the "user-friendly" version and we will go on improving the integration of the component with more options according to the Showcases and the Framework requests.

1.1.6 Availability

The component is available. We will build new grammars for CALLAS applications on demand. Future versions with new features (vocabulary changes, etc.) of the device can also be asked.

1.2 Emotion recognition from Acoustic Features

1.2.1 Overview

Automatic emotion recognition from speech has in the last decade shifted from a side issue to a major topic in human computer interaction and speech processing. The aim is to enable a very natural interaction with the computer by speaking instead of using traditional input devices and not only have the machine understand the verbal content, but also more subtle cues such as affect that any human listener would easily react to. This can be used in spoken dialogue systems, e.g. in call center applications. However, so far real-time emotion recognition has scarcely been attempted. We provide a first full working component for emotion recognition.

1.2.2 Description

An automatic system for the recognition of emotions from speech has three main tasks: the *segmentation* of the incoming audio signal into suitable analysis units, the extraction of emotion relevant *features* from these acoustic units, and the *classification* into an emotional state.

Usually linguistic units such as words or utterances serve as units of analysis for speech emotion recognition. In an online application, however, these would have to be extracted by a an automatic speech recognition system, which is both error-prone and time-consuming. Therefore, a voice activity detection algorithm[2] is used here, which segments the incoming audio signal in real-time into voiced segments not longer than about 1 second. It solely

2 from ESMERALDA (http://sourceforge.net/projects/esmeralda)

operates on the acoustic signal and makes no use of any linguistic knowledge.

From these voiced segments then acoustic features are extracted that are in general relevant for vocal emotion. They are mainly statistics like mean, maximum, etc. over the analysis unit derived from pitch, energy, MFCCs, the frequency spectrum, duration and pauses resulting in a vector of 1316 features. In order to speed up classification and to make the features more specific to the particular task, a feature selection can be performed on the basis of acoustic test data. A more detailed description of the feature calculation can be found in [3].

For the final recognition of emotional states from the acoustic feature vectors, currently, two classification algorithms are integrated into the speech emotion recognition system, Naïve Bayes and support vector machines (SVM)[4]. Though SVM is generally slightly more accurate, Naïve Bayes is recommended for real-time applications, because of SVM's slower build and recognition times.

1.2.3 Documentation

Before using the emotion recognizer, the system must be trained on specific emotions and speakers. The system is independent from emotional categories. It depends on the training and can be e.g. [joy, boredom, anger] or [aroused, neutral, calm]. There is no restriction on the amount of categories. But, the fewer categories used, the higher the accuracy and the categories should be mutually exclusive.

EmoVoice Tools consists of three applications, EmoVoiceRecorder, EmoVoiceRecorderFT (optional) and EmoVoiceClassifierTool. EmoVoiceRecorder is a tool to record audio samples with given sentences stored in a XML file. EmoVoiceRecorderFT lets you record free speech and tag it with an emotion. EmoVoiceClassifierTool helps you to calculate the features and the classifier and gives you some information about the quality[5] of your corpus.

EmoVoiceRecorder

EmoVoiceRecorder is a tool for recording audio samples with help of prepared sentences which must be stored in a XML file. You can easily edit this file. You must define the character encoding in the first line of the file. In most cases ISO-8859-1, -15 or UTF-8 should work, depending on your operating system and text editor. You can add emotion classes within the XML file. The emotion class tag must consist of letters only (a-z, A-Z). If you look at `ExampleSentences.xml` you get an idea how simple it is. We recommend recording at least 40 samples per class to get a good classifier.

---

3 T. Vogt and E. André, "Comparing feature sets for acted and spontaneous speech in view of automatic emotion recognition", in Proc. IEEE Int. Conf. on Multimedia & Expo (ICME). 2005

4 LIBSVM - a Library for Support Vector Machines is used (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)

5 We only can proof the quality regarding the number of samples and the recognition rate with a "10 percent 10-fold cross-validation". Bad result here could occur from a corpus which samples *prosodic* quality does not suffice.

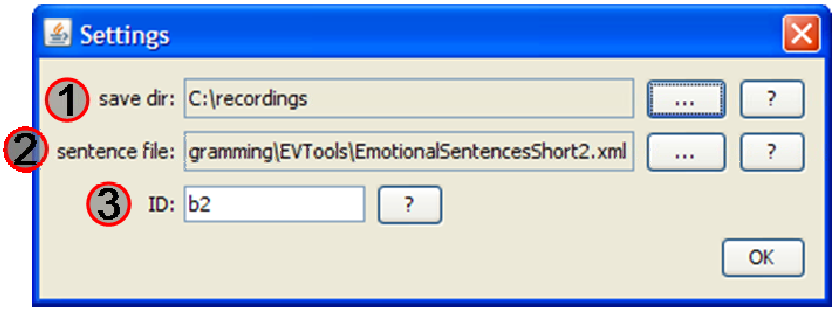


Figure 2: EmoVoiceRecorder (Settings)

To start EmoVoiceRecorder double click it (under Windows) or start with 'java -jar EmoVoiceRecorder.jar'. The first dialog (see Figure 2) lets you specify following things:

①. Specify a directory where the recorded samples will be saved.

②. Specify the XML file with the prepared sentences.

③. Give a unique ID that identifies each person.

Press OK and all settings are saved. See Figure 3 for the recording dialog. The drop down menu (①) lets you choose the emotion class defined in the XML file. You can select a sentence with < (②) and > (③). In ④ you can see the current sentence to record.

Figure 3: EmoVoiceRecorder (Application)

For recording a sample press *Record* (⑤) and speak into your microphone and press *Stop* (⑥) after speaking. Be aware that everything between pressing *Record* (⑤) and *Stop* (⑥) is used as one audio sample and should be one emotional expression (sentence or whole utterance).

The files will be named and saved automatically in the defined *save dir* after pressing *Stop*.

Tips for the recording:

Please read each utterance first for yourself and then say it loud. Please don't worry about slips of the tongue. Just go ahead. If you have the impression that an utterance does not support you in feeling a particular emotion, please feel free to edit the file `ExampleSentences.xml`.

EmoVoiceClassifierTool

EmoVoiceClassifierTool helps you creating the classifier and checks the quality of your corpus.

Calculate features and build classifier (see Figure 4):

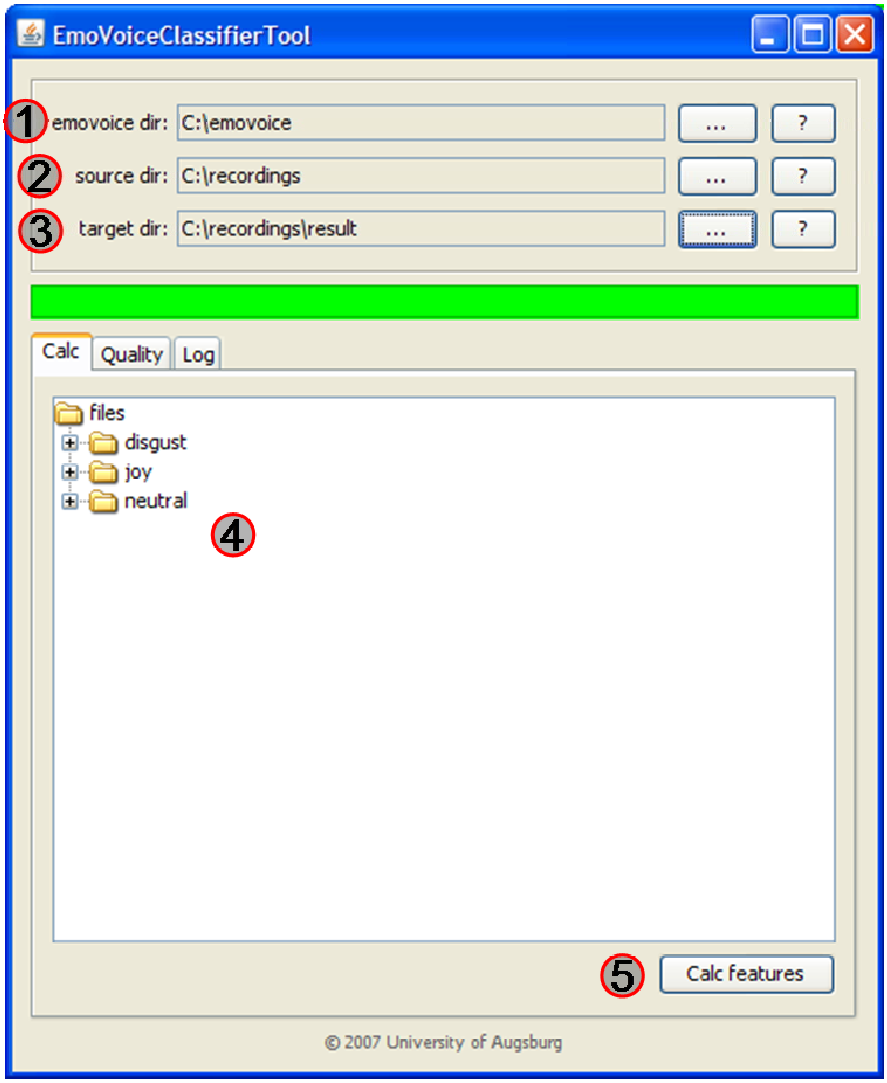


Figure 4: EmoVoiceClassifierTool (Calculate features and build classifier)

①. Choose where you store the EmoVoice applications.

②. Choose where you store the audio sample files, you recorded with EmoVoiceRecorder. If you have several folders with samples that should be used for the classifier copy them into one folder and select this folder as *source dir*. All valid files in the folder will be taken for building the classifier. If you want to select specific files you must delete them from the corpus and redefine the *source dir*.

③. Choose where you want to save all the feature files and the classifier. Please choose an empty directory.

After you defined the directories you should see a tree (④) with all emotion classes. Here you can check if all files are attached to the correct class. To calculate the features and to build the classifier press *Calc features* (⑤). This will take a while dependent on your system and the amount of samples. You can follow the process in the Log tab. The red bar indicates that the application is busy. You will find the classifier named `classifier.cl` in the target directory.

Check quality (see Figure 5):

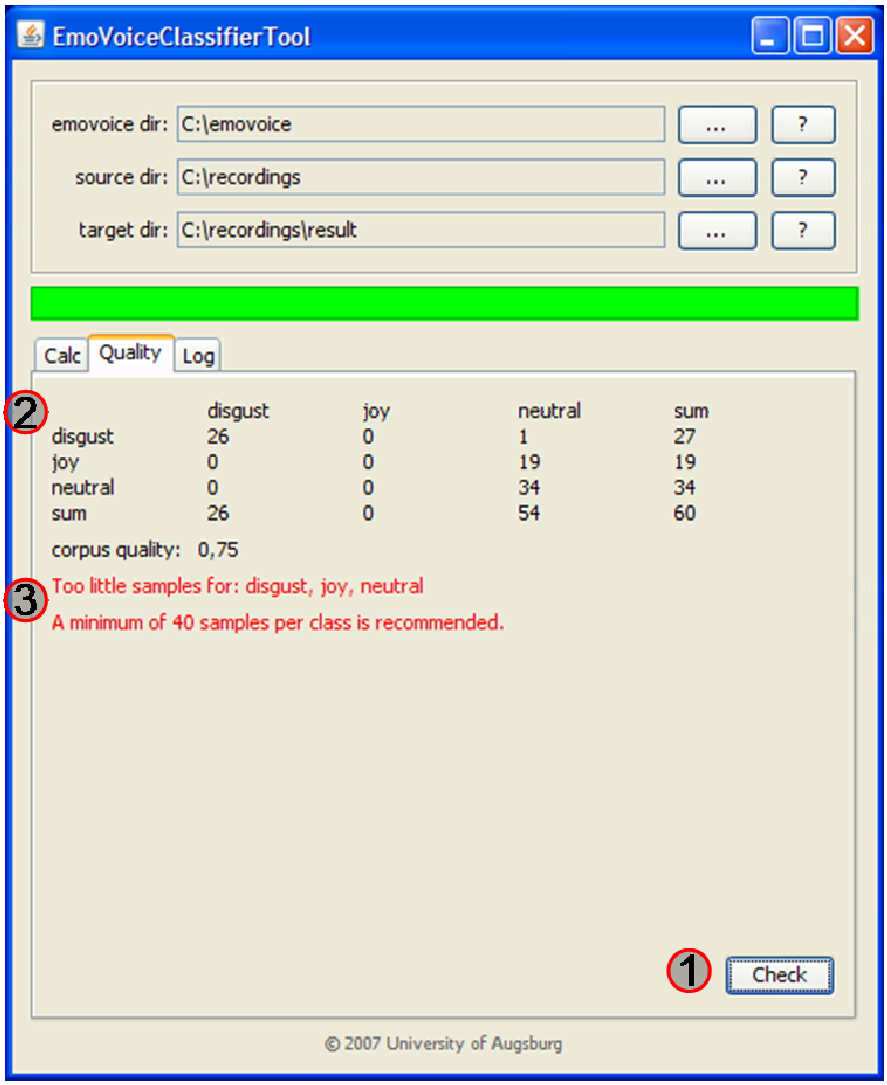


Figure 5: EmoVoiceClassifierTool (Check quality)

If you want to check the quality of your corpus switch to the *Quality* tab. If you just build the classifier you do not have to change any directories, as the quality check takes the *target dir* as input directory for proving the quality. To check the quality press *Check* (①). This will take a while again. The green bar indicates a busy application and you can follow the process in the *Log* tab again.

After the check is done you will see a confusion matrix (②) and a factor that indicates the classification quality. A higher quality factor (maximum is 1.0) promises a lower

misclassification. But, a quality factor of 1.0 does not mean a misclassification of 0%. In case you have less than 40 samples you will see a warning (③). You can ignore it, if you're aware that the amount of samples could be too less for a good classification result afterwards.

Some remarks, if you use the tool with Windows:

It is possible to run EmoVoice under Windows with the help of Cygwin. That means, EmoVoice is not a native Windows application. Therefore there are some restrictions, if you use EmoVoiceClassifierTool under Windows:

- You cannot use folders containing blanks.
- Do not start the application from a Cygwin console with '`java -jar EmoVoiceClassifierTool.jar`'. You must start it either from a Windows console or via double-click.

EmoVoiceRecorderFT (optional)

EmoVoiceRecorderFT is a tool for recording audio samples in free text mode, in case you plan to record emotion with free speech.

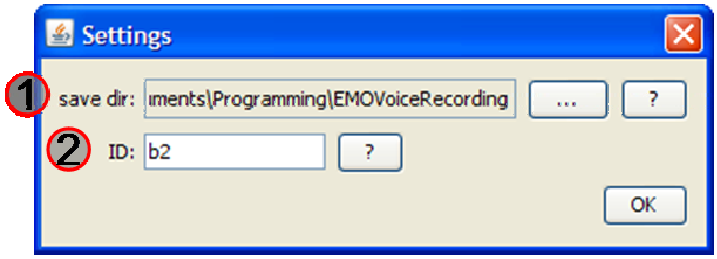


Figure 6: EmoVoiceRecorderFT (Settings)

To start EmoVoiceRecorderFT double click it (under Windows) or start with '`java -jar EmoVoiceRecorderFT.jar`'. The first dialog (see Figure 6) lets you specify following things:

①. Specify a directory where the recorded samples will be saved.

②. Give a unique ID that identifies each person.

Press OK and all the settings are saved. See Figure 7 for the recording dialog in free text mode. The *emotion tag* drop down menu (①) lets you choose your current emotion you plan to record. If your emotion is not available in the list, you simply enter it in the drop down menu.

Figure 7: EmoVoiceRecorderFT (Application)

For recording a sample press *Record* (③) and speak into your microphone and press *Stop*

(④). Be aware that everything between pressing *Record* (③) and *Stop* (④) is used as one audio sample and should be one emotional expression (sentence or whole utterance).

The files will be named and saved automatically in the defined *save dir* after pressing *Stop*.

The *tag list* option (②) shows who many samples of each emotion you already recorded.

In case you have chosen a *save dir* where you already recorded some samples before, you will have automatically the existing emotion tags ready in the drop down menu.

EmoVoice Online Recognition

For recognizing emotions online you need a classifier file and the command line tool *emo_online*.

You will find two versions of EmoVoice. One (*EmoVoice_Linux*) is pre-compiled for Linux (32bit, gcc 4.1.2) and the other (*EmoVoice_Windows*) is usable (with Cygwin[6]) under Windows.

emo_online opens a line to the microphone and starts continuously analyzing the input. By default the output is send to the console. You also can send the output via socket (add option '`-e [port]`').

Use `emo_online classifier.cl` for sending the recognition result to the standard output (in most cases this should be the console) or use e.g. `emo_online -e 3669 classifier.cl` for sending the recognition result via socket (you can use any port number you like).

For further information start *emo_online* with `emo_online -h`.

1.2.4 Requirements

- Java 6 or newer (http://java.com/en/download/index.jsp)
- If you plan to use EmoVoiceClassifierTool you also need EmoVoice (emo_asegment_file, emo_fextract_file, emo_ctrain and emo_cclassify_file).
- Microphone

1.2.5 Status

A full working version of EmoVoice is available. Currently the component used with three emotional classes achieves a recognition rate above 70% on user dependent classifier training.

1.2.6 Availability

The component is available for all CALLAS partners. Please find more detailed information on the CALLAS wiki page[7].

1.3 Emotion recognition from Linguistic Features

1.3.1 Overview

The task of the component is the classification of emotions that are conveyed through simple texts. The computer system gets an input text that has to be classified e.g. a movie review or

[6] You do not need to install Cygwin. *cygwin1.dll* is added and that is all you need. You can run the Windows applications from a Windows command line.

[7] http://wiki.callas-newmedia.eu/twiki/bin/view/Main/AccousticRecognition

a sentence and calculates its emotional meaning.

1.3.2 Description

An automatic system for the recognition of emotions from texts distinguishes two types of texts: long and short. Long texts (more than 200 words) are processed by a statistical approach, and short texts (a sentence) are managed by a semantic approach.

The statistical approach is based on the standard data mining algorithm. It extracts linguistic features (words) from the analyzed text and evaluates them by counting their occurrences. For the final recognition of emotional states from the linguistic feature vectors the support vector machines (SVM) are used[8].

The approach was already tested on the following English corpora: the Pang corpus[9], the Berardinelli movie review corpus[10], a corpus with spontaneous dialogues (the SAL corpus)[11], a corpus with product reviews[12].

A more detailed description of the feature extraction and evaluation can be found in[13].

1.3.3 Documentation

Statistical Text Analyzer

Before using the analyzer, it must be trained. An example training dataset is included with the distribution. The system is independent of emotional categories and can classify emotions expressed e.g. in star notation [zero, one, two, three, four] stars (zero stars – poor, four stars – excellent). There is no restriction on the amount of categories although, the fewer categories, the higher is the accuracy of recognition.

8 Weka SVM – from the WEKA data mining toolkit (http://www.cs.waikato.ac.nz/ml/weka/)

9 Pang, B., Lee, L., Vaithyanathan, S. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. Proc. of EMNLP-02, the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

10 The corpus was collected from the epinions.com and contains 11,000 reviews on digital cameras.

11 Cowie, R., Douglas-Cowie, E., Savvidou, S., McMahon, E., Sawey, M., Schröder, M.. 2000. 'FEELTRACE': An instrument for recording perceived emotion in real time. In: Proceedings of the ISCA Workshop on Speech and Emotion, Northern Ireland. pp. 19–24.

12 The corpus containing 215 movie reviews from www.reelviews.net.

13 Alexander Osherenko and Elisabeth André. Lexical Affect Sensing: Are Affect Dictionaries Necessary to Analyze Affect?. In *Proceedings of Affective Computing and Intelligent Interaction (ACII)*, Springer, 2007.

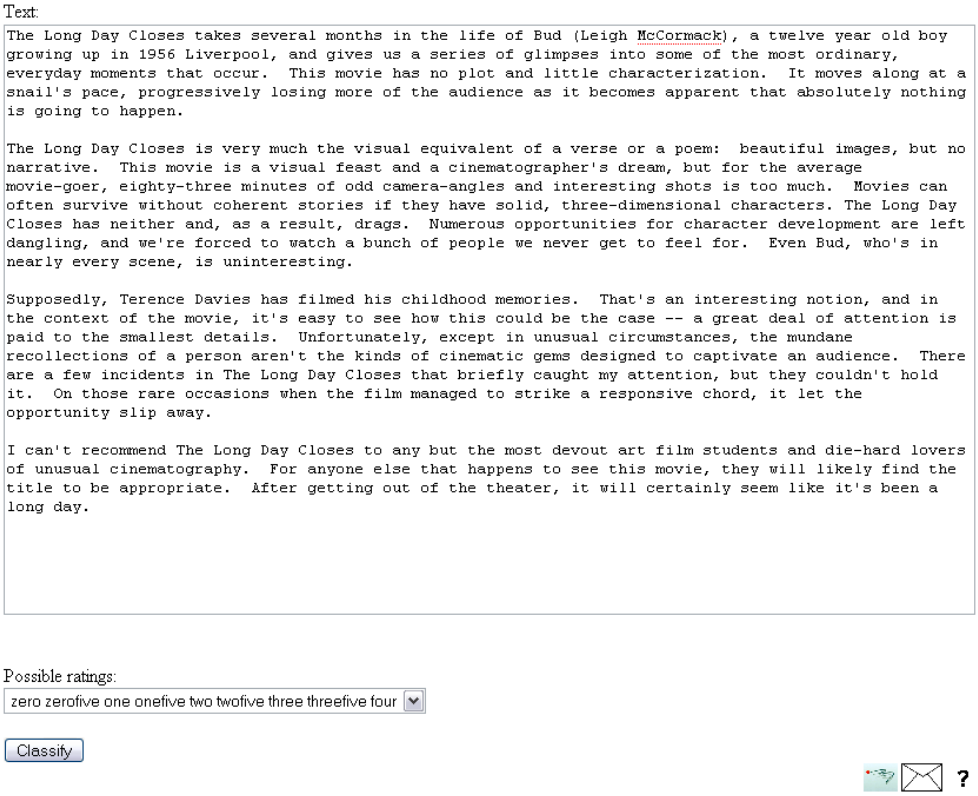


Figure 8: Statistical Text Analyzer

To classify the text regarding its emotional meaning (Figure 8):

1. Enter a text to classify e.g. a movie review in the text field *Text*.
2. Choose a set of emotional categories the system recognizes in the list *Possible ratings*.
3. Click the *Classify* button.

The result of emotion recognition containing the calculated emotional rating and the processed text are shown in Figure 9:

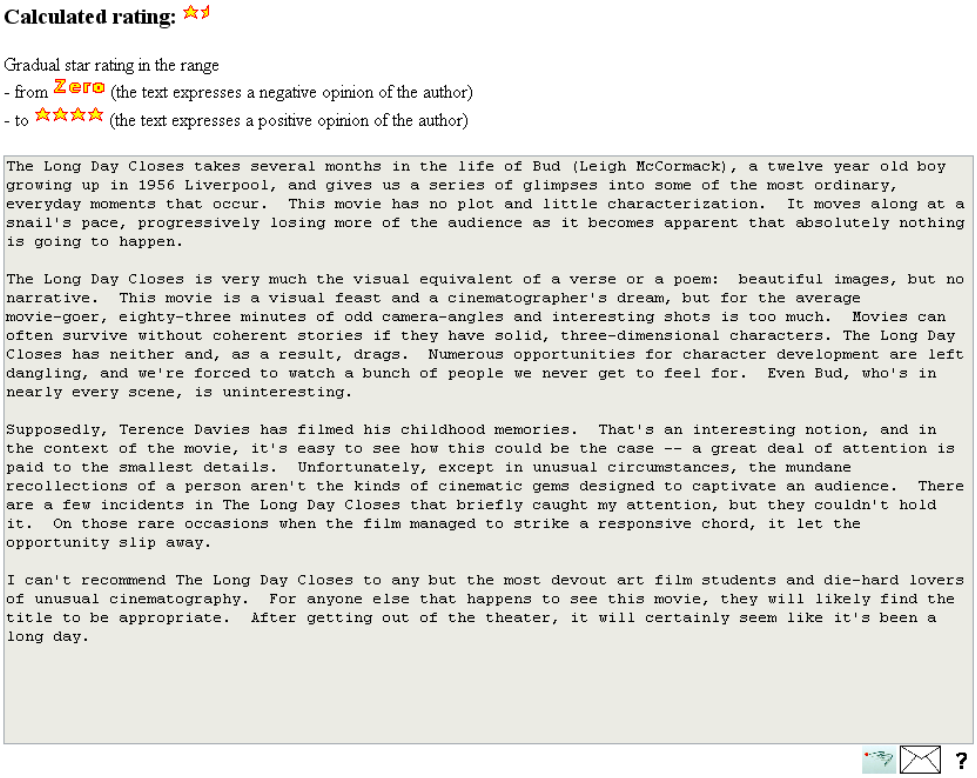


Figure 9: Result of Emotional Classification by the Statistical Text Analyzer

Figure 8 and Figure 9 show at the bottom from left to right three buttons (homepage, email, and info) that are used for as a link to the homepage, for sending an email and for displaying an info message resp.

Semantic Text Analyzer

The system does not require training. It is independent of emotional categories and can classify emotions e.g. [high negative, low negative, neutral, high positive, low positive] stars. There is no restriction on the amount of categories although, the fewer categories, the higher is the accuracy of recognition.

The system parses the sentence using two parsers (the SPIN parser[14] and the Stanford Parser[15]), extracts its emotional words and calculates the emotional meaning. To classify a sentence regarding its emotional meaning (Figure 9):

1. Enter a sentence to analyze.
2. Click the *Calculate* button.

14 Engel, R. 2006. SPIN: A Semantic Parser for Spoken Dialog Systems.In *Proceedings of the Fifth Slovenian And First International Language Technology Conference (IS-LTC 2006)*, 2006.

15 Klein, D., Manning, C. D. 2003. Accurate Unlexicalized Parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pp. 423-430.

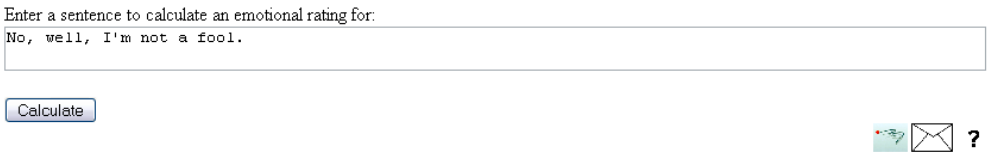


Figure 10: Semantic Text Analyzer

The result of emotion recognition containing the calculated emotion as a text and as a descriptive photo illustrating the expressed affect is displayed (Figure 10):

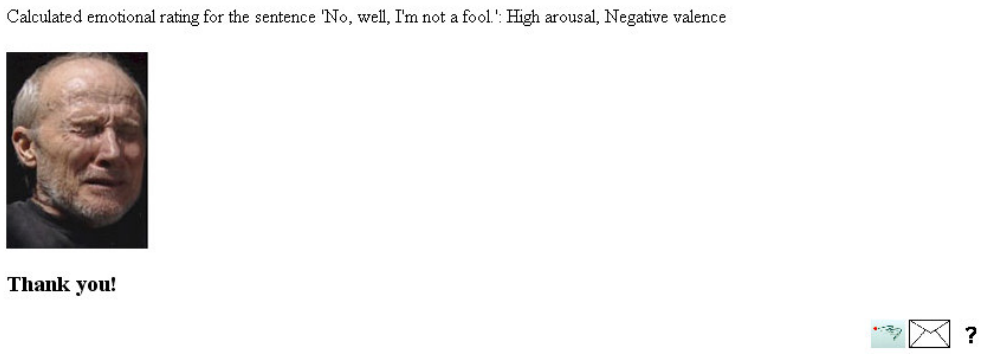


Figure 11: Result of Emotional Classification by the Semantic Text Analyzer

Figure 10 and Figure 11 also show at the bottom from left to right the three buttons (homepage, email, and info) that are used for as a link to the homepage, for sending an email and for displaying an info message resp.

1.3.4 Requirements

- Java 6 or newer (http://java.com/en/download/index.jsp);
- Perl, for instance, ActivePerl 5.8.7 (http://www.activestate.com/Products/activeperl/);
- TreeTagger (http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/);
- Stanford Parser (http://nlp.stanford.edu/software/lex-parser.shtml);
- SPIN parser (http://www.dfki.de/~rengel/).

1.3.5 Status

The statistical text analyzer works is ready because it relies mainly on the already ready SVM.

The semantic text analyzer is conceptually ready and has to be extended with SPIN rules what can be done ongoing depending on concrete user requirements.

1.3.6 Availability

Online versions of both analyzers (statistical and semantic) are available here [http://emotion.informatik.uni-augsburg.de:8080/WebInterface].

2. Audio-Video Feature Extraction

2.1 Audio Feature Extraction

2.1.1 Overview

The main area where audio features and audio analysis is utilized is in different multimedia devices and their novel applications are handling increasing amounts of multimedia content such as video, audio, images, messages and music. Audio features and their analysis enable automatic metadata extraction from video and audio recordings enable the development of sophisticated multimedia content management applications which can help users to manage their personal recordings. Audio based metadata extraction concentrates on general audio, speech and music analysis. General audio analysis attempts to segment and classify the audio signal to different events. The detected events may contain semantic meanings i.e. speech and music but the segments may only represent audio signal with different properties.

Many of the recent research on audio content analysis and segmentation concentrate on material from news archives, digital libraries and TV programs/movies [Lu2002], [Wu2005], [Lin2005]. Analysing the amateur created video material with mobile phones has also increasing interest among the researchers, [Mäkelä2006], [Vuorinen2007]. Such a data has a new challenge since the lack of professional structure and quality of the data. In Callas project the purpose is classify online audio to different categories, close to this are audio analysis applications concerning context awareness by analysing environmental audio. Korpipää & al. used multiple sensors for context awareness, but with plain audio they reached accurary 87.6% of correct positive recognition. They used large set of audio features and classified them with HMM's [Korpipää2003]. HMM based environmental audio analysis in [Ma2006] reached overall accuracy of 92% for 11 acoustic environments. Also surveillance applications have growing interest in audio analysis for detecting suspicious events as in [Radhakrishnan2005]

Typically audio content is classified first into a basic set of main audio categories. Generally these classes contain speech, music, silence and additionally different noise classes or mixed classes, i.e. speech with music. Two typical approaches are hierarchical rule-based classification and statistical classification. Short explanation An example of the rule-based classification is in [Li2000] where they classified seven basic audio types achieving classification accuracy over 90% for audio from movies and TV programs and movies. A kNN classifier was used together with hierarchical classification system in [Lu2002]. They achieved an accuracy of 96.51% for three classes for news material.

[Lu2002]L. Lu, H.-J. Zhang and H. Jiang, “Content Analysis for Audio Classification and Segmentation,” *IEEE Trans on Speech and Audio Processing*, vol. 10, no 7, pp 504 – 516, Oct. 2002.

[Wu2005]C.-H. Wu, C.-H. Hsieh, “Multiple Change-Point Audio Segmentation and Classification Using an MDL-Based Gaussian Model”, *IEEE Transactions on Speech and Audio Processing,* Accepted for future publication, vol PP, is 99, pp.1 – 11, 2005.

[Lin2005]C.-H. Lin, S. –H. Chen, T.-K. Truong, Y. Chang, “Audio Classification and Categorization Based on Wavelets and Support Vector Machine,” *IEEE Transactions on Speech and Audio Processing,*vol. 13, is. 5, Part 1, pp. :644 – 651, Sept. 2005

[Mäkelä2006] Mäkelä S.-M., Peltola J., Myllyniemi M.,”Mobile Video Capture Targeted Narrowband Audio Content Classification”, Proceedings of the IEEE Internation Conference on Acoustics, Speech, and Signal Processing, ICASSP 2006, May 15-19, 2006, Toulouse, France

[Vuorinen2007] Vuorinen O., Peltola J., Mäkelä S.-M."Unsupervised Speaker Change Detection for Mobile Device Recorded Speech", IEEE Internation Conference on Acoustics, Speech, and Signal Processing, ICASSP 2007, April 15-20, Honolulu, Hawaii, USA

[Korpipää2003] P. Korpipää, M. Koskinen, J. Peltola, S-M Mäkelä and T. Seppänen, "Bayesian Approach to Sensor-Based Context Awareness", Pers Ubiquit Comput, 2003, 7:113 -124.

[Ma2006] Ling Ma, Ben Milner and Dan Smith,"Acoustic Environment Classification", ACM Transactions on Speech and Language Processing,Volume 3 , Issue 2 (July 2006), pp 1-22, 2006

[Radhakrishnan2005] Radhakrishnan, R.; Divakaran, A.; Smaragdis, A.;"Audio analysis for surveillance applications", Applications of Signal Processing to Audio and Acoustics, 2005. IEEE Workshop on
16-19 Oct. 2005 Page(s):158 - 161

[Li2001] Li D, Sethi I.K, Dimitrova N and McGee T, "Classification of General Audio Data For Content-based Retrieval", Pettern Recognition Letters 22(2001), pp 533- 544

2.1.2 Description

The audio feature extraction classifies the audio stream to 5 different sound classes. The classes of current version are speech, music, silence, constant noise (i.e. car engine noise) and variable noise (i.e. restaurant noise). The component output is corresponding audio class for each audio frame. The component will take as input either live or recorded audio.

The audio feature uses for classification is MFFC (mel frequency cepstrum), and 12 coefficients are calculated for 30 ms frame with 10ms overlap. The classification is based on HMM (Hidden Markov Models) statistical classification and the system utilises TORCH (http://www.torch.ch/) library for classification. HMM models are trained with 3 states and 2 mixtures for each class. The models are generated left-right model for constant noise, silence and speech and ergodic models music and variable noise. The database contained 72 minutes training data for all the classes together. The classification result of frame based classification is filtered with in 1 second window to avoid small fluctuation of the frame based classification results.

2.1.3 Documentation

Before using the audio feature component AA_HMM, the three the external libraries which are listed in the requirements section must be installed. The algorithm has predefined models for each audio class and they are installed in model\ directory.

The audio component has two different versions for recorded audio and live audio input. Apart of different input format the architecture of the component is the same. The off line component is delivered for testing and play around purposes. The Figure 12 has the outline of the audio analysis component.

Figure 12: Architecture of AA_HMM component

First the MFCC feature is calculated for each audio frame. MFCC's are buffered for 1 s period and those segments are classified, by shifting the buffer one frame at the time. The classification results are smoothed to avoid temporal fluctuation of the results in frame basis. The output of the component is the classification result for each second.

For offline console application the input audio file has to be in 16kHZ, mono, .wav format. The function is called as follows:

AA_HMM.exe [test_wav_file]

and the result will be in written in result.txt file, where each second of the classified audio has the category label.

The live version of the audio component requires a microphone attached to the audio file and the component started. The component supports 16 kHz and 32 kHz live audio input. The component displays a counter on a screen for each classified audio segment. Number on a counter is increased by 1 when the current segment is classified to corresponding class. Explanation for the counter abbreviations in the screen shot below: M music, Sp speech, Si silence, Cn constant noise and Vn variable noise.

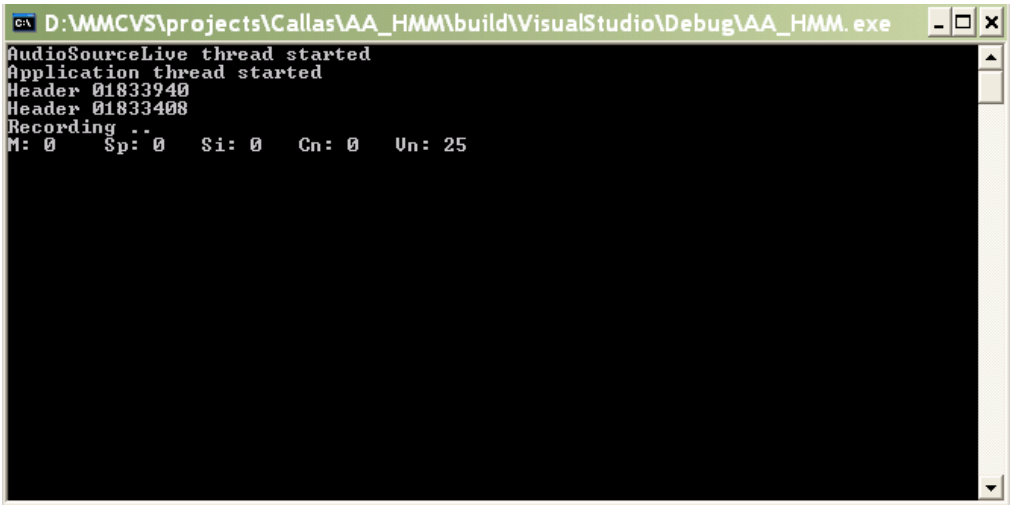


Figure 13: Screen shot of the live audio analysis component output.

2.1.4 Requirements

- The component supports Windows operating system
- For live audio input a microphone attached to the computer
- The component needs following libraries:

  FFmpeg (avcodec-51.dll, avformat-51.dll, avutil-49.dll)

2.1.5 Status

The current module recognizes the above 5 mentioned categories. By retraining the component it is possible to have different audio categories. The future development will concentrate on recognizing more human related sounds i.e. laughter, claps, that are more suitable and beneficial for Showcase purposes. The retrained component is not available yet.

2.1.6 Availability

The current module is available in CALLAS wiki

2.2 Video Feature Extraction

2.2.1 Overview

The video analysis is widely researched area and utilized in many application fields for instance in surveillance, multimedia content analysis, medical imaging. In multimodal interaction the main areas of video analysis has been concentrating in gesture recognition, pose/gaze recognition, head tracking and later with facial expression recognition. The video feature components goal is to obtain general information of the people participating to an event or installation. The approach is to use face detection for counting and tracking people, as well as orientation of the head for movement information.

There are many approaches to detect a head from a given image. (Video stream can be handled as a sequence of images.) The methods can be roughly divided to four approaches: knowledge based methods, feature invariant approaches, template matching methods and appearance based methods [1]. Once the area of head have been detected it is possible to define other attributes regarding the face including pose, expression etc.

The information of head pose has been utilized many ways in multimodal interaction. For example in [2] the pose was used for interactive dialog systems and in [3] for finding the direction of intention in video meeting system.

The head area can also used as an object for tracking purposes. Object tracking is a challenging task and there are multiple ways to overcome the problem. These methods are introduced in [4].

[1] Ming-Hsuan Yang; Kriegman, D.J.; Ahuja, N.; Detecting faces in images: a survey Pattern Analysis and Machine Intelligence, IEEE Transactions on Volume 24, Issue 1, Jan. 2002 Page(s):34 – 58

[2] Morency L-P, Darrell T: Multimodal conversational agents: From conversational tooltips to grounded discourse: head pose tracking in interactive dialog systems. Proc. of the 6th international conference on Multimodal interfaces ICMI '04 **,** October 2004

[3] Jilin Tu; Huang, T.; Yingen Xiong; Rose, T.; Quek, F.;Calibrating Head Pose Estimation in Videos for Meeting Room Event Analysis, IEEE International Conference on Image Processing, 8-11 Oct. 2006 Page(s):3193 – 3196

[4] Yilmaz A, Javed O, Shah M., Object tracking: A survey, ACM Computing Surveys, Volume 38, Issue 4 2006

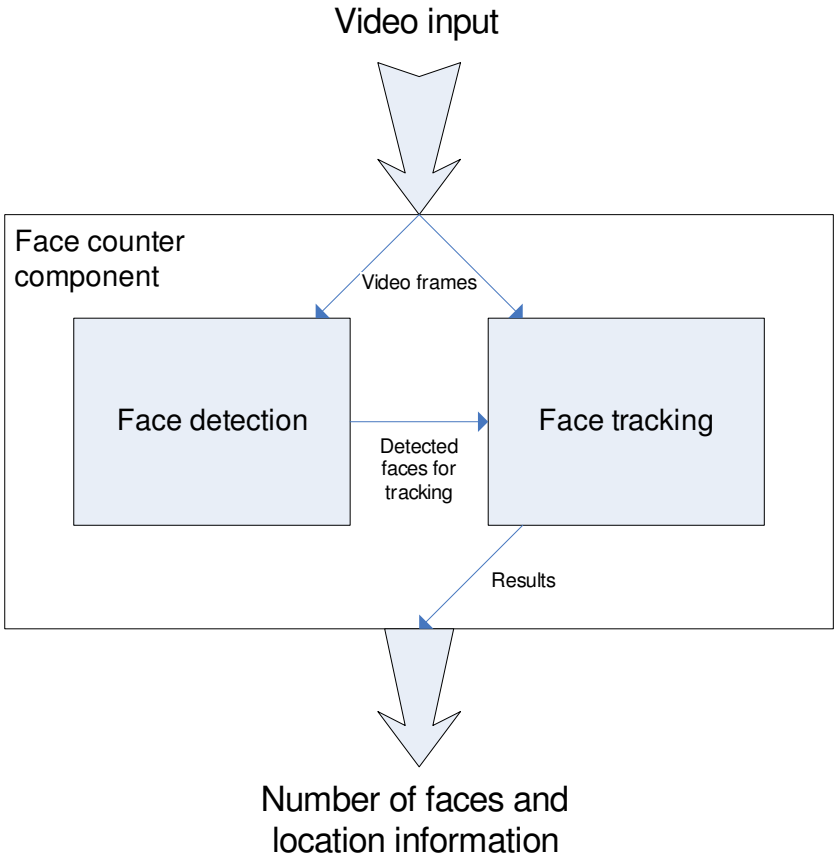
2.2.2 Description

Video feature extraction component extracts faces from video sequence or live camera feed. The component can extract the number of faces in the video frame. The component is also a video player and it plays video files and captures live feed from camera. The detected faces are marked to the played video and the information of location, size, and angle of the detected faces in the video frame are written out.

Video feature extraction component has two main functions. First, it decodes video frame from the file or captures frame from the camera. Second, it extracts faces from the video frame. Video decoding is based on external libraries that are included in the FFmpeg multimedia system. Face extraction is based on Open Computer Vision Library (OpenCV). Face extraction uses two functions object detection and object tracking. Object detection searches and detects faces from the video frame. The detected faces are then tracked using object tracking function. The component uses trained classifiers for detecting faces. It can use Haar cascade classifiers which are included in the OpenCV package or classifiers which are trained by user.

2.2.3 Documentation

Before using the video extraction component the external libraries which are listed in the requirements section must be installed. The classifiers for face detection can be trained or already trained classifiers can be used. Especially if the component will be used in the specific circumstances the classifiers can be trained. Otherwise it is recommended to use already trained classifiers.



Video feature extraction component consists of two separate applications: BCcomp-file for video files and BCcomp-cam for live camera input.

BCcomp-file

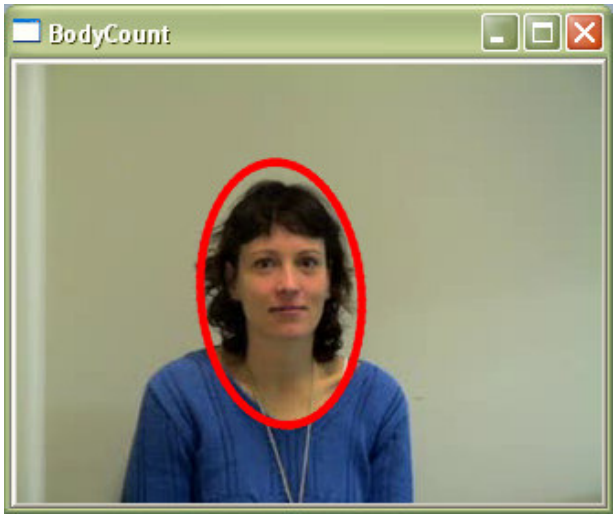
BCcomp-file is a console application for extracting faces from video file. To start the application use the following command line:

BCcomp-file.exe [video-file] [cascade-file]

Where, *video-file* is the input video file. The application supports several file formats through the libavformat library. *Cascade-file* is Haarcascade XML-files that are trained classifiers for detecting objects of a particular type, e.g. faces.

The application plays the input video file where the detected faces are bounded. In addition the output information that includes location, size and angle of detected faces is printed to the screen (see Figure 14). Output information:

Frame: [number] [ROI number] [Center-X] [Center-Y] [Width] [Height] [Angle]



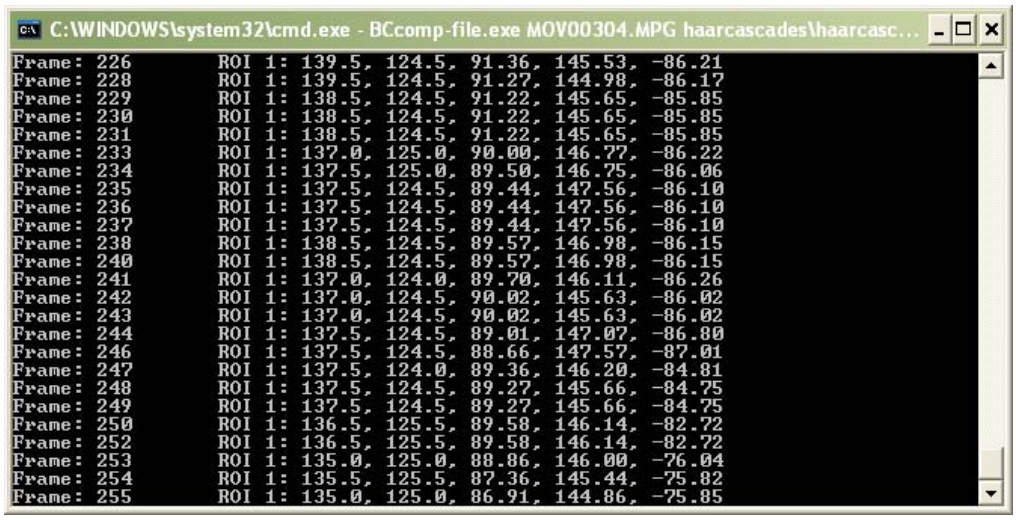


Figure 14: BCcomp-file (Output)

BCcomp-cam

BCcomp-cam is a console application for extracting faces from live feed from video camera. Connect the camera to the computer using usb or firewire connection. The camera driver must be properly installed. To start the application use the following command line:

BCcomp-cam.exe [[cascade-file]

Where, *cascade-file* is Haar cascade XML-file that is trained classifier for detecting objects of a particular type, e.g. faces

The application shows the input video feed where the detected faces are bounded. In addition the output information that includes location, size and angle of detected faces is printed to the screen. Output information:

Frame: [number] [ROI number] [Center-X] [Center-Y] [Width] [Height] [Angle]

2.2.4 Requirements

- The component supports Windows operating system
- For video capturing camera and suitable drivers for cameras are needed
- The component needs following libraries:
  - OpenCV
  - FFmpeg (avcodec-51.dll, avformat-51.dll, avutil-49.dll)

2.2.5 Status

The first version of the video feature extraction component is ready. The component is console application and currently there are separate applications for capturing video from the camera and for reading video from file. The application plays the video sequence where the boundaries of the detected faces are marked. The output information that includes location, size and angle of detected faces is printed to the screen.

2.2.6 Availability

The applications are currently available for test purposes.

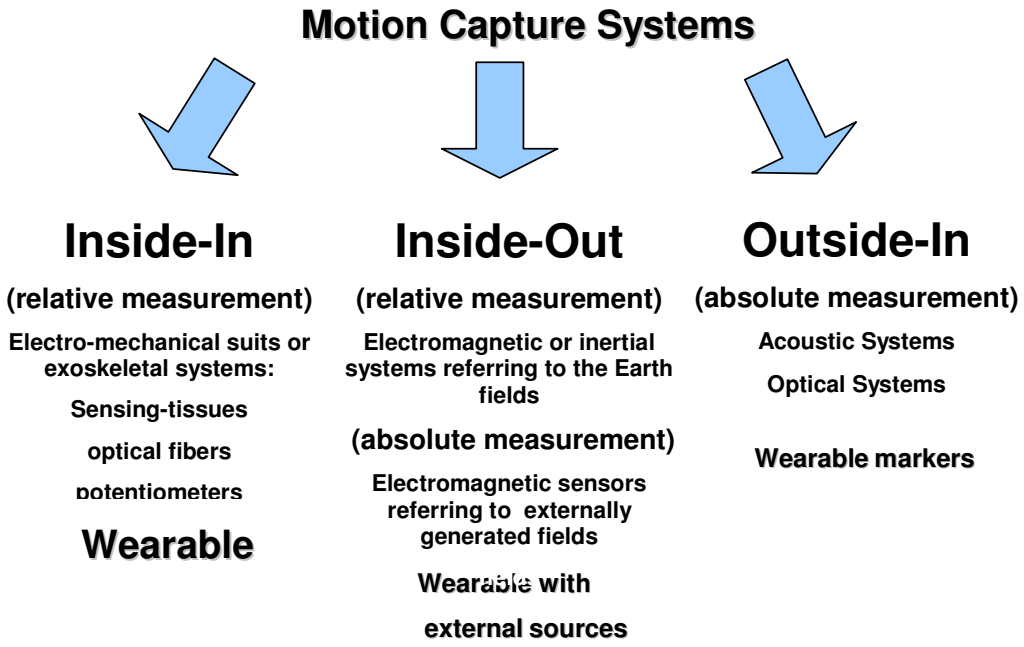
3. Gesture and Body Motion

3.1 HumanGlove

3.1.1 Overview

Regarding motion and gesture detection, we can list three main categories:

- *Inside-in technologies*, in which both the transducers and eventually the source of the field to be measured lie in the device (sensing suits or exoskeletons with Hall effect based sensors, potentiometers, magnetoresistors, optical fibers or even sensing tissues)
- *Inside-out technologies*, in which the transducers are on board the sensing device but they sense the magnetic or gravitational field of the Earth or a generated external (magnetic) field
- *Outside-in technologies*, in which the sensors are not on the links or on the joints but located in the surrounding environment. In some cases these technologies make use of active or reflective markers.



The aim of wearable devices developed by Humanware during the Callas Project is the motion tracking in non-structured environment: there will be no need for controlled lights, optical markers or well positioned multicameras (as in optical systems, e.g. Vicon systems), no need for magnetic environments (as in Polhemus systems) o acoustic transmitter/receiver (as in Zebris system). The user will wear a suit or partial suit according to the body portion to be tracked. “Inside-in” motion tracking technologies will be exploited. The main purpose of this devices is puppeteering, motion tracking to replicate human motion in avatars or extracting emotions from gestures. Although these wearable autonomous devices are

suitable for any VR related purpose.

3.1.2 Description

In these first 12 months a glove tracking the motion of the hand has been released. The current release of the glove detects and measures all the finger flexions. Any link is endowed with a sensor unit, as described below, devoted to measure the flexion of the following joint. Hall Effect based sensors detect the magnetic field induced by permanent magnets rotating while the phalanges flex.

Referring to Figure 15 a brief description of the sensing unit follows: one end of the torsion spring 5 is fixed in the springholder 4; the other end of the spring is fixed in the magnet pulley 1. The Spring preloader 9 rotates the springholder 4, obtaining the desired pretensioning of the spring 5. When the desired tension is obtained, the spring preloader 9 is fixed on the sensor holder 7 by means of the screw 11. In the magnets pulley 1 there are the magnets housings to hold the permanent magnets 2 and 3 in the right position, while the sensor holder 7 has the housing for the Hall Effect sensor 8. The pulling cable 6 has one end stuck to the following phalange and the other end is stuck to the magnet pulley 1 around which is winded. When the phalange flexes the cable 6 is pulled and the pulley 1 turns. While the pulley 1 turns, the magnets rotate and the magnetic field induced on the sensor 8 varies. When the phalange extends, the cable 6 is released and the pulley 1 turns back because of the spring 5, winding the cable 6 around the pulley 1.

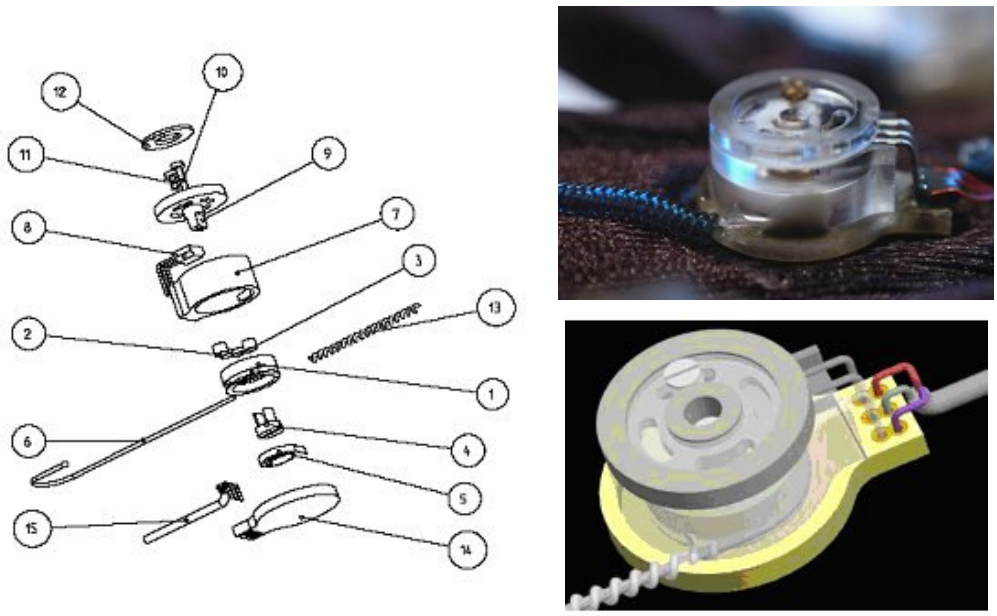


Figure 15: Sensing unit layout

3.1.3 Documentation

The HumanGlove is provided with:

- control unit converting the analog signals (the glove sensors output) into digital form, in order to transmit them through a common RS-232 line to a PC or to a workstation. The next release of the glove will be endowed with a virtual RS-232 via USB, and the data will be transmitted wirelessly via Bluetooth.

- a software showing a virtual hand replicating the hand posture of the user, for test and calibration purposes
- libraries developed with the aim of simplifying the communication with the control unit of the HumanGlove for anyone interested in developing applications that exploit the HumanGlove.
- user's manuals (software manual, developer's manual, installation manual)

The communication through the RS-232 occurs at 38400 baud 8n1 (8 bit, no parity bit and one bit of stop). The communication protocol between the control unit and the PC is the following:

1. The control unit remains in a waiting state until it receives the character 'A' (0x41);
2. The PC sends the character 'A' (0x41);
3. Once this character is received, the control unit sends a packet with the positions of the DOFs;
4. The PC reads the packet with the data;
5. The control unit returns to point 1;
6. The PC returns to point 2.

The data packet transmitted to the PC is made of 24 channels (DOF) and every channel is 16bit long (an `unsigned short`) for a total of 48 bytes. Every channel contains a value between 0 and 4095 (12bit resolution). The last 4 channels of the packet are reserved for future use and should therefore be ignored.

The control unit sends the 12 bits for each DOF starting from the most significant (bit n.11) to the least significant (bit n.0). After these 12 bits, it sends 4 bits for padding purposes, so that each DOF value is aligned to a 16bit-word. So, when a value is received, its bits are arranged as shown in Figure 16.

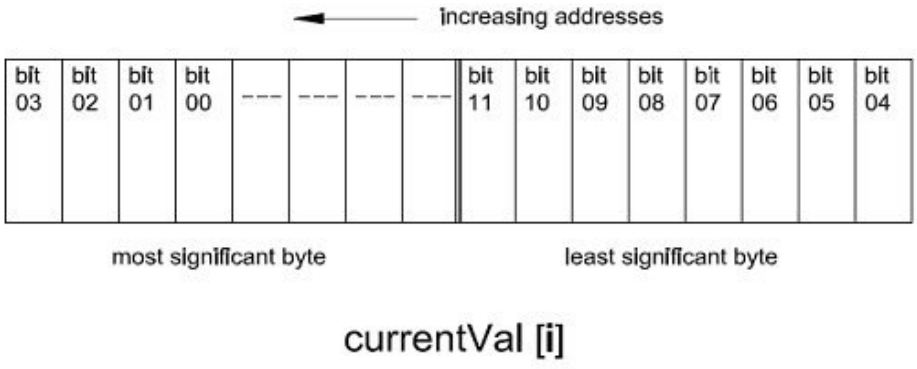


Figure 16: The currentVal array, as it is received from the control unit.

The order of the channels within the packet (and within the currentVal[] array) is the following:

short n.	DOF
0	Thumb Abd/Add
1	Thumb Flex/Ext 1
2	Thumb Flex/Ext 2
3	Thumb Flex/Ext 3
4	Index Abd/Add (index MP add)
5	Index Flex/Ext 1 (index MP flex)
6	Index Flex/Ext 2 (index PIP flex)
7	Index Flex/Ext 3(index DIP flex)
8	Middle Abd/Add (middle MP add)
9	Middle Flex/Ext 1(middle MP flex)
10	Middle Flex/Ext 2 (middle PIP flex)
11	Middle Flex/Ext 3 (middle DIP flex)
12	Ring Abd/Add (ring MP add)
13	Ring Middle Flex/Ext 1 (middle MP flex)
14	Ring Middle Flex/Ext 2 (middle PIP flex)
15	Ring Middle Flex/Ext 3 (middle DIP flex)
16	Little Abd/Add (little MP add)
17	Little Middle Flex/Ext 1 (little MP flex)
18	Little Middle Flex/Ext 2 (little PIP flex)
19	Little Middle Flex/Ext 3 (little PIP flex)
20	Reserved
21	Reserved
22	Reserved
23	Reserved

Figure 17: Number order of the DOFs of the HumanGlove

Calibration of the acquired values

The output values of the control unit are integers between 0 and 4095. Once they are acquired, these values must be transformed into angles.

Assuming a linear transform, the value recorded by HumanGlove is transformed into an angle using the following formula:

$$Angle = \frac{ChannelValuee*(MaxAngle - MinAngle)}{(MaxValue - MinValue)} + MinAngle$$

Meaning of the sampled values

The HumanGlove is able to record the position of 4 angles (1 abduction/adduction and 3 flexion) for each of the 5 fingers.

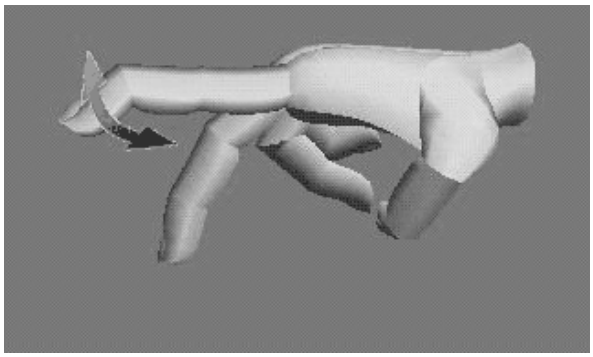


Figure 18: The HMW_SGLV_INDEX4 DOF

In the library *SerialLib* and *CalibLib* these angles have been numbered starting from the abduction/adduction (the first DOF) and the proximal metacarpal joint (the second DOF is the metacarpal-phalangeal flexion) to the distal joint of the finger (the fourth DOF is the distal-inter-phalangeal flexion).

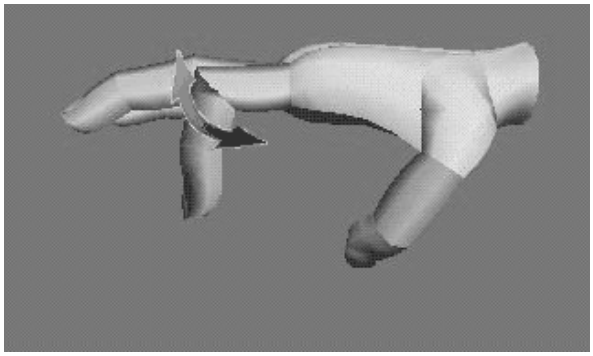


Figure 19: The HMW_SGLV_INDEX3 DOF

With this method,

- the DOF marked by index 1 is the index add/abduction DOF named *hmw_sglv_index1* (shown in Figure 21)
- the DOF marked by index 2 is the MP flexion DOF named *hmw_sglv_index2* is (shown in Figure 19 and Figure 20)
- the DOF marked by index 3 is the PIP flexion DOF named *hmw_sglv_index3* is (shown in Figure 19 and Figure 18)

- the DOF marked by index 4 is the DIP flexion DOF named *hmw_sglv_index4* is (shown in Figure 17)

Figure E shows the degree of freedom *hmw_sglv_index2* when the angle is 0 and Figure 20 shows the degree of freedom *hmw_sglv_index2* when the angle is about 90.

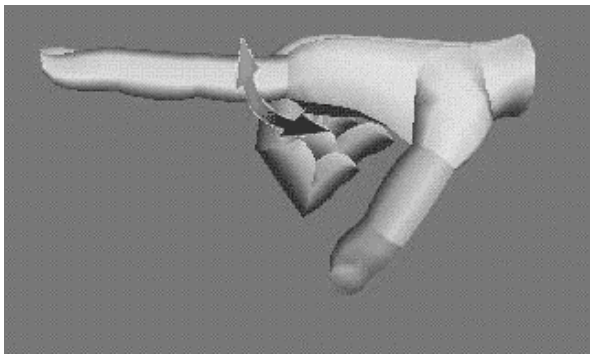


Figure 20: The HMW_SGLV_INDEX2 DOF (the angle is 0°)

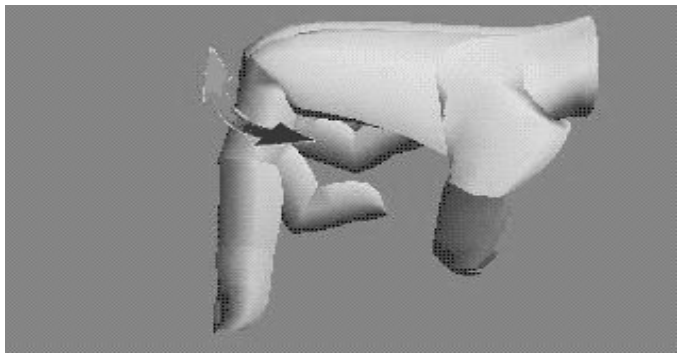


Figure 21: The HMW_SGLV_INDEX2 DOF (the angle is 90°)

The default output is set as follows: the value 90 degrees is associated with a complete flexion (90 degrees of the physiological joint), as in figure F, while the value 0 degrees is a full extension (0 degrees of the physiological joint) as in figure E. The abduction/adduction DOF is shown in Figure 21: in this case the zero corresponds to the second image.

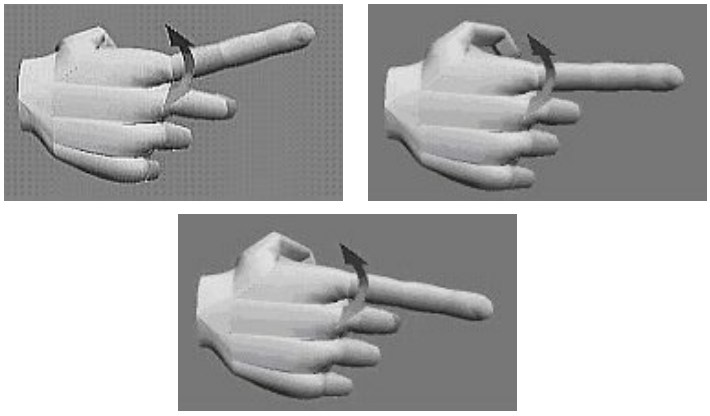


Figure 22: The HMW_SGLV_INDEX1 DOF (the zero corresponds to the second image)

3.1.4 Requirements

Unit sensing performance:

Non Linearity: < 2.5%

Accuracy: < 0.1V/2.5 deg

Range: 110 deg

The glove is connected to the host computer through a standard RS-232 at 38400 baud. It can be easily connected to any workstation, PC or Macintosh. In the next release (expected within M14 of the Callas Project) USB connection and Bluetooth connection will be available.

3.1.5 Status

A prototype, which relies on 15 DoFs, is available. Within the month 14 the wireless connection will be released and a new unit sensor for the ad/abduction will be tested. From Month 12 to 18 the upper limb of the user will be endowed with a partial suit. The task related to the glove can be considered 85% accomplished. The slight delay is because of the difficulty to set up a reliable wireless transmission. Aesthetic improvements are also expected according to the user's requirements.

3.1.6 Availability

A single prototype has been built and it will be available to the callas partners on demand.

3.2 Gesture Recognition from Mobile Phones

3.2.1 Overview

Previous work on gesture control and recognition can be classified into two main categories, camera-based and movement sensor-based. Camera-based recognition is most suitable for stationary applications, and often requires specific camera setup and calibration. A review on camera based methods can be found in [1].

The movement sensor-based approach utilises different kinds of sensors e.g. tilt, acceleration, pressure, conductivity, capacitance, etc. to measure movement. An example of such an implementation is GestureWrist, a wristwatch-type gesture recognition device using

both capacitance and acceleration sensors to detect simple hand and finger gestures [2]. Tsukada and Yasumura developed a wearable interface called Ubi-finger, using acceleration, touch and bend sensors to detect a fixed set of hand gestures, and an infrared LED for pointing a device to be controlled [3]. XWand, a gesture-based interaction device, utilises both sensor-based and camera-based technologies [4]. The creators of XWand present a control device that can detect the orientation of device using a 2-axis accelerometer, a 3-axis magnetometer and a 1-axis gyroscope, as well as position and pointing direction using two cameras. The system is also equipped with an external microphone for speech recognition. The user can select a known target device from the environment by pointing, and control it with speech and a fixed set of simple gestures.

Gesture recognition were used in Ambicence project Smart Design Studio to interact with virtual reality environment [5] and in Atelje project to navigate in multimedia presentation [6]. In both cased the developed gesture recognition platform utilised a technology is based on a SoapBox (Sensing, Operating and Activating Peripheral Box), which is a sensor device developed for research activities in ubiquitous computing, context awareness, multi-modal and remote user interfaces, and low power radio protocols [7]. The basic sensor board of SoapBox includes a three-axis acceleration sensor, an illumination sensor, a magnetic sensor, an optical proximity sensor and an optional temperature sensor. Acceleration sensors measure both dynamic acceleration (e.g. motion of the box) and static acceleration (e.g. tilt of the box). They provide useful information for the recognition of human hand gestures and a new way to add human motion to the human/machine interface. [8]

To use mobile phone with accelerometers as a sensor has advantages compared to special sensors. The price of the phone is lower than of the tailor made gadget and it is easy to purchase. Mobile phone is something that everybody posses and in future it can be a multipurpose tool even for accessing gesture driven multimodal interfaces. This encourages the development the gesture/body motion tracking by the mobile phone as a sensor.

[1] Mitra, S.; Acharya, T:Gesture Recognition: A Survey Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on Volume 37, Issue 3, May 2007 Page(s):311 – 324

[2] Rekimoto J (2001) GestureWrist and GesturePad : Unobtrusive Wearable Interaction Devices. Proceedings of the Fifth International Symposium on Wearable Computers, ISWC 2001, pp 21-31.

[3] Tsukada K, Yasumura M (2002) Ubi-Finger: Gesture Input Device for Mobile Use. Proceedings of APCHI 2002, Vol. 1, pp 388-400.

[4] Wilson A, Shafer S (2003) Between u and i: XWand: UI for intelligent spaces. Proceedings of the conference on Human factors in computing systems, CHI 2003, April 2003, pp 545-552.

[5] Kela, J., Korpipää, P., Mäntyjärvi, J., Kallio, S., Savino, G., Jozzo, L., Di Marca, S. Accelerometer-based gesture control for a design environment.Personal and Ubiquitous Computing special issue on Multimodal Interaction with Mobile and Wearable Devices, Springer-Verlag 2005.

[6] Iacucci G, Kela J, Pehkonen P (2004). Computational support to record and re-experience visits. Personal and Ubiquitous Computing, Vol 8 No 2, Springer-Verlag, pp 100-109, 2004

[7] Tuulari E, Ylisaukko-oja A (2002) SoapBox: A Platform for Ubiquitous Computing Research and Applications. First International Conference, Pervasive 2002, pp 26-28, 2002

[9] Kallio S, Kela J, Mäntyjärvi J, Plomp J: Visualization of hand gestures for pervasive computing environments. AVI2006: 480-483

3.2.2 Description

An automatic system for the recognition of gestures reads Nokia 5500 sensor data using Bluetooth communication and computing simple contexts (orientation and acceleration level). The output of the component describes the orientation, acceleration and if the “browser” button of the phone is pressed.

3.2.3 Documentation

Installation

- Nokia 5500 mobile phone

  Install sis-package to Nokia 5500 mobile phone using your preferred way (Bluetooth, data cable etc.). Via Bluetooth:

  1) Create the Bluetooth connection between PC and mobile

  2) Send .sisx package from PC to phone (as text message)

  3) Open the text message and install the application

- PC

  Unzip Gesture library to some directory in your computer. Copy GestureLibrary.ini file to your Windows directory (usually c:\windows). Edit GestureLibrary.ini LibPath to point lib-directory of Gesture library. Alternative location for GestureLibrary.ini is the directory where you start your PC-application. If you don't want to use GestureLibrary.ini at all you must launch PC-application from the lib directory of Gesture library.

Module description

The modules of gesture recognition system are shown in Figure 23.

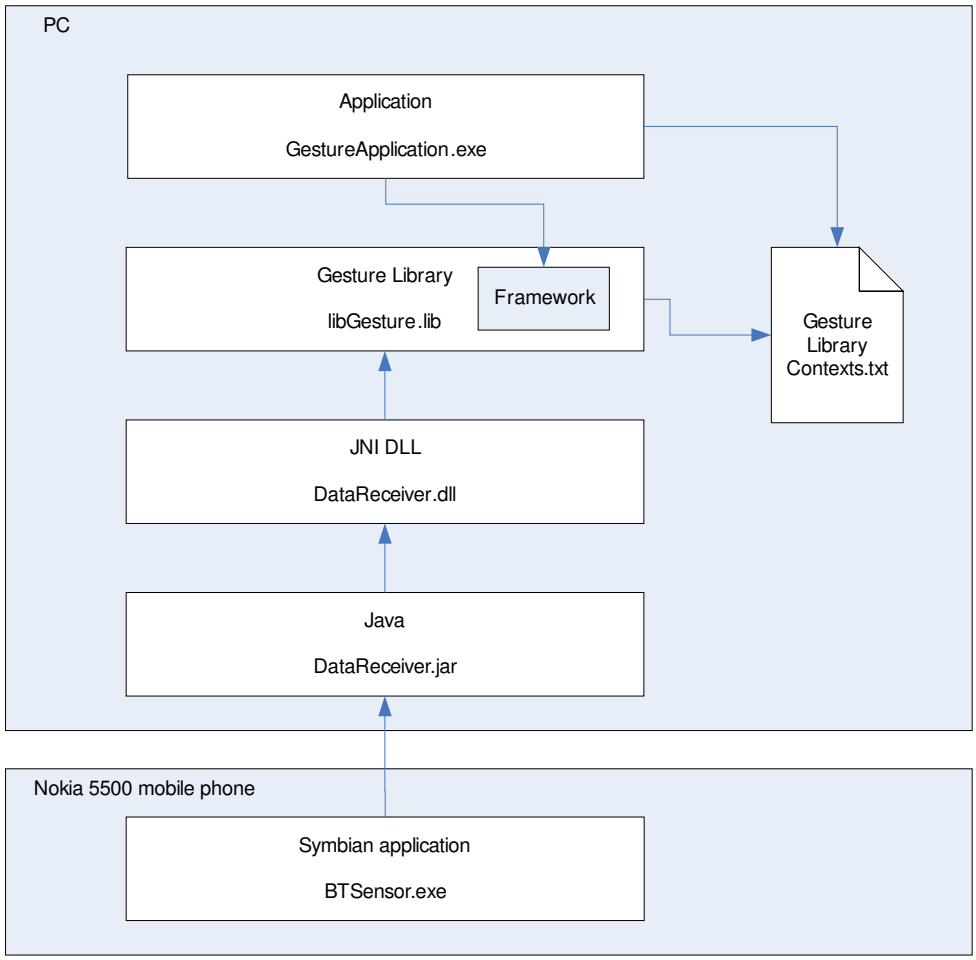


Figure 23: Gesture recognition system modules

When phone module (BTSensor) is launched it advertises Bluetooth service with identifier 0x10ff and starts waiting connection from PC.

When PC application containing Gesture library is launched, Gesture library kicks up Java for searching for Bluetooth devices which provide service identifier 0x10ff.

Once connection is established between mobile phone and PC, mobile phone starts sending sensor data to PC. Sensor data includes acceleration x, y and z. Incoming data is inserted to the Framework input buffer.

Each time when new data is inserted to the Framework, the Gesture library reasons new contexts values.

Computing orientation context

Orientation context is reasoned with the latest acceleration values. Acceleration values are first normalized with acceleration minimum and maximum values.

$$acceleration_{xnorm} = acceleration_x - \frac{(\min_x + (\max_x - \min_x)/2}{(\max_x - \min_x)/2}$$

Corresponding normalization is done also for acceleration y and z. Then calculate angles

$$angle_x = \arcsin(acceleration_{xnorm})$$

Corresponding angles also for y and z. Then finally confidences:

$$DisplayUp = \frac{angle_z}{coff}$$

$$DisplayDown = \frac{-angle_z}{coff}$$

$$DisplayLeft = \frac{angle_y}{coff}$$

$$DisplayRight = \frac{-angle_y}{coff}$$

$$DeviceUp = \frac{-angle_x}{coff}$$

$$DeviceDown = \frac{angle_x}{coff}$$

Divider *coff* is used to scale confidences to the range 0 to 100. The biggest confidence value will be the reasoned context for orientation. Reasoned context is inserted to the Framework output buffer. Also confidence values for all six contexts are inserted to output buffer. Results are also written to text file.

Computing acceleration context

Acceleration context is reasoned from five last acceleration values. Standard deviation is calculated for acceleration x, y and z. These deviations are summed

$$deviation_{tot} = deviation_x + deviation_y + deviation_z$$

If total deviation is more than threshold for high acceleration, context is *AccelerationHigh.* Else if total deviation is more than threshold for moderate acceleration, context is *AccelerationModerate.* Else if total deviation is more than threshold for low acceleration, context is *AccelerationLow.* Else acceleration context is *AccelerationStill.* Reasoned context is inserted to the Framework output buffer. Also acceleration value scaled from 0 to 100 is inserted to output buffer.

Obtaining contexts

Contexts reasoned by Gesture library can be obtained in two ways

- Read GestureLibraryContexts.txt. When Gesture library is running it generates all the time GestureLibraryContexts.txt file containing latest reasoned contexts. Example of GestureLibraryContexts.txt

  Orientation 101: 63, 0, 12, 0, 0, 0

  Acceleration 201: 0

  Button 302: 100

  Orientation context is 101 which means "display up", see below. In this example display was mostly up but also a bit left. Next six values are 0 to 100 values

indicating the confidence for each context (order is the same as in the following table). In this case you can think that display is 63% up but also 12% left. Acceleration context is 201 which means "still". Acceleration value is 0, maximum would be 100. Button context is 302 which means "not pressed". Button value is always 100.

```
// Orientation contexts
const int ORIENTATION_DISPLAY_UP        = 101;
const int ORIENTATION_DISPLAY_DOWN      = 102;
const int ORIENTATION_DISPLAY_LEFT      = 103;
const int ORIENTATION_DISPLAY_RIGHT     = 104;
const int ORIENTATION_DEVICE_UP         = 105;
const int ORIENTATION_DEVICE_UPSIDEDOWN = 106;
// Acceleration contexts
const int ACCELERATION_STILL            = 201;
const int ACCELERATION_LOW              = 202;
const int ACCELERATION_MODERATE         = 203;
const int ACCELERATION_HIGH             = 204;
// Button contexts
const int BUTTON_PRESSED                = 301;
const int BUTTON_NOT_PRESSED            = 302;
```

- Create C++ application and read contexts directly from Framework output buffer. The following piece of code reads the orientation information

```
int* data = new int[ SIZE_ORIENTATION ];
if ( TheFramework::Instance()->GetData (ID_ORIENTATION,
(void **) &data ) )
{
      int iContext = data[0];       // context, f.e. 101
      int displayUp = data[1];      // display up value
(0->100)
      int displayDown = data[2];    // display down
value (0->100)
      int displayLeft = data[3];    // display left
value (0->100)
      int displayRight = data[4];   // display right
value (0->100)
      int deviceUp = data[5];       // device up value
(0->100)
      int deviceDown = data[6];     // device down value
(0->100)
}
  else
```

```
{
    // buffer empty, orientation data not available
}
delete data;
```

3.2.4 Requirements

- Nokia 5500 mobile phone
- Microsoft Windows XP SP2 or newer
- Bluetooth hardware and bluetooth stack (Following Bluetooth stacks are supported: Microsoft, BlueSoil, WIDCOMM)
- Java run-time environment

3.2.5 Status

Currently the gesture and body motion component recognizes the mobile phone movement from the accelerator sensors of the phone. The future development will concentrate on gesture recognition for more specifically. The aim is to use also mobile phone with the integrated acceleration sensors in capturing the hand movements. Also direct measurement modality will be developed to track other type of free (i.e. no specific training needed) hand movements in addition to tilting.

3.2.6 Availability

The component is available.

3.3 Gesture Expressivity Features Extraction from Video

3.3.1 Overview

The localisation of regions of interest in the approach of the described component is achieved by detecting skin regions. The proposed method[16] is particularly efficient in terms of processing cost while simultaneously the recognition rate is particularly high. The assumptions that are imposed by the particular algorithm, and what will be reported below, are not particularly restrictive while the margin for improvement, both qualitatively and temporal is possible with the use of various heuristic methods.

The process is the following. Each frame is converted from the initial RGB color space in the YCrCb color space. Using the three resulting chromatic components the joint probability of each pixel being a skin pixel is calculated. This way for each pixel of the frame is assigned a probability denoting whether it contains chromatic information of skin or not (Figure 24). Following that, the chromatic skin mask is produced with use of a threshold which has been determined by the user. Applying the resulting mask in the initial frame, the regions that contain high skin probability are produced.

[16] G. Caridakis, A. Raouzaiou, K. Karpouzis, S. Kollias, "Synthesizing Gesture Expressivity Based on Real Sequences", Workshop on multimodal corpora: from multimodal behaviour theories to usable models, LREC 2006 Conference, Genoa, Italy, 24-26 May.

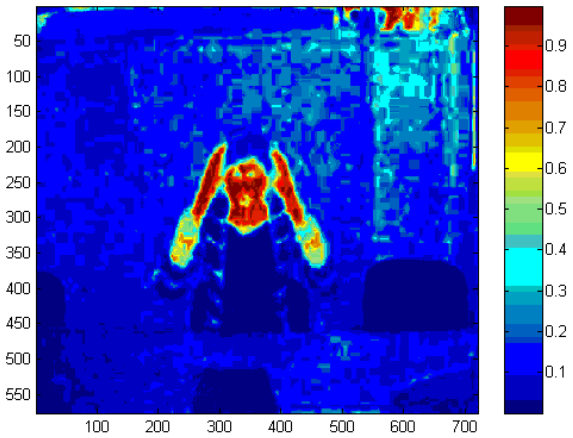


Figure 24: Skin probability

3.3.2 Description

The above process, except for the regions of interest that we wish to distinguish in the initial frame, usually produces also various regions that have chromatic characteristics very close to those of skin and which we want to avoid. To achieve this we follow the following process. A large number of these regions are much smaller than any of the region of interests, hands and head in the case in question. With this observation and selecting relatively big threshold in the creation of chromatic dermal mask we can avoid the import of noise in the frame with the form of small points that have been recognized as dermal regions. Then, in order to remove this noise which might survive in the picture even after the application of the previous method, we apply some morphological operators. Taking into consideration the size and form of what we expect the regions to have we apply suitable morphological operators using a structural element constituted by a disk, whose size is a fraction of the diagonal of parallelogram that is the convex hull of the head. Hence segments of the image which have been recognized as skin regions but have a smaller size are removed from the final mask. An overview of the proposed algorithm is depicted in Figure 25.

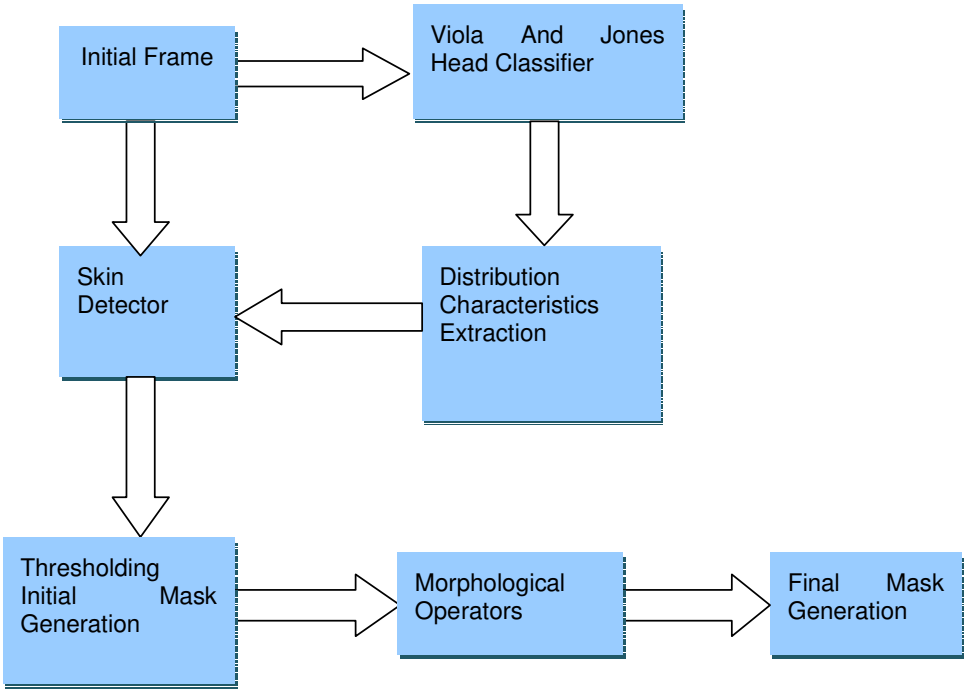


Figure 25: Overview of the algorithm

3.3.3 Documentation

The above algorithm was implemented in a platform combining various architectures and technologies. This platform constitutes an interface between .NET and two other technologies, namely OpenCV and ffmpeg. A snapshot of the application appears inFigure 26. .NET was used mainly for the possibility of using object oriented programming, something which gives us the opportunity to develop a wider and extensible platform. At the same time the user friendly environment and the accessibility of low level code were some of the advantages. Using methods of communication with lower level languages we enhance the platform with the OpenCV and ffmpeg capabilities. OpenCV is used for frame processing of the video and for the implementation of the Viola - Jones algorithm. ffmpeg was used for export frames from video stored in form of file while simultaneously we have the possibility of using all the models of compression which supports the particular library with particularly efficient way. Finally the treatment of pixels for the calculation of probabilities, centers of weight of regions as well as morphological operations and the graphic environment were implemented in the .NET environment, because it was decided that they did not constitute parts of the algorithm that added particular computational cost.

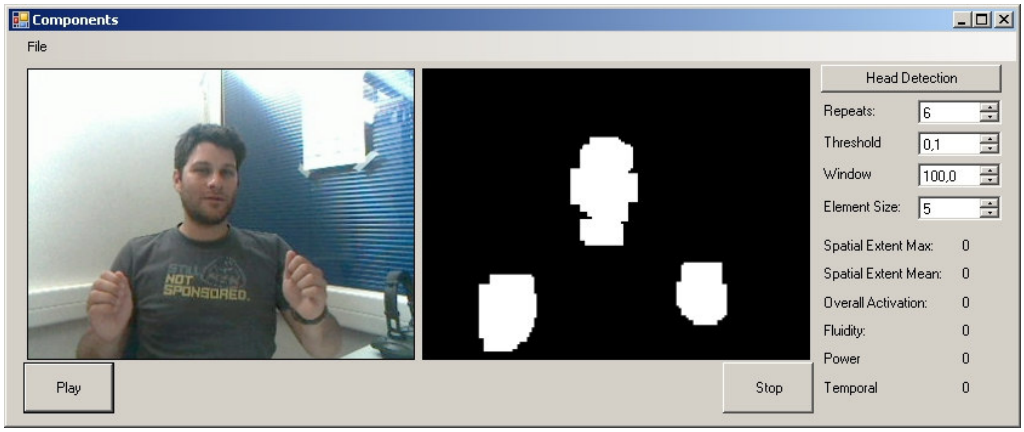


Figure 26: Application screenshot

3.3.4 Requirements

The component requires the installation of OpenCV and ffmpeg. In terms of input the camera should not have any special features e.g. high speed, multi resolution, etc. It has been tested in most Windows OSs (2000, XP).

3.3.5 Status

Beta version.

3.3.6 Availability

The application is still under development, but in a demonstrable stage.

4. Facial Expression Recognition

4.1 Facial Feature Detection

4.1.1 Overview

A system for the detection of facial characteristics is provided in this component. Facial feature detection is a crucial step for other applications, such as expression recognition, gaze detection, pose estimation, etc. The component constitutes a system for detecting facial features and tracking their path in a video sequence. The features detected and tracked are the eye corners and eyelid centers, as well as the eye centers. Mouth corners are also detected and tracked.

4.1.2 Description

Prior to eye and mouth region detection, face detection is applied on the face images. The face is detected using the Boosted Cascade method. The output of this method is usually the face region with some background. Furthermore, the position of the face is often not centered in the detected sub-image. Consequently, a technique to postprocess the results of the face detector is used. More specifically, a technique that compares the shape of a face with that of an ellipse is used. According to this technique, the distance map of the face area found at the first step is extracted. Here, the distance map is calculated from the binary edge map of the area. An ellipsis scans the distance map and a score representing the average of all distance map values on the ellipse contour *el*, is evaluated. This score is calculated for various scale and shape transformations of the ellipses. The transformation which gives the best score is considered as the one that corresponds to the ellipse that best describes the exact face contour. The lateral boundaries of the ellipse are the new boundaries of the face region.

A template matching technique is used for the facial feature area detection step: The face region found by the face detection step is brought to certain dimensions and the corresponding edge map is extracted. Subsequently, for each pixel on the edge map, a vector pointing to the closest edge is calculated and its *x,y* coordinates are stored. The final result is a vector field encoding the geometry of the face. Prototype eye patches were used for the calculation of their corresponding vector fields and the mean vector field was used as prototype for searching similar vector fields on areas of specified dimensions on the face vector field. The candidate region of the face that minimizes a search criterion is marked as the region of the left or right eye.

For the eye center detection, the normalized area of the eye is brought back to its initial dimensions on the image and a light reflection removal step is employed. The grayscale image of the eye area is converted to a binary image and small white connected components are removed. The final result is an eye area having reflections removed. Subsequently, horizontal and vertical derivative maps are extracted from the resulting image and they are projected on the vertical and horizontal axis respectively. The mean of a set of the largest projections is used for an estimate of the eye center. Following, a small window around the detected point is used for the detection of the darkest patch and its center is considered as the refined position of the eye center.

For the detection of the eye corners (left, right, upper and lower), Projection Functions are employed. More in detail, Generalized Projection Functions (GPFs), which are a combination of the Integral Projection Functions (IPFs) and the Variance Projection Functions (VPFs), are used. The integral projection function's value on row (column) *x (y)* is the mean of its luminance, while the Variance Projection Function on row (column) *x (y)* is its mean variance.

The GPF's value on a row (column) *x (y)* is a linear combination of the corresponding values of the derivatives of the IPF and VPF on row *x* (column *y*). Local maxima are used to declare the positions of the eye boundaries.

For the mouth area localization, a similar approach to that of the eye area localization is used: The vector field of the face and prototype images are used for the extraction of a prototype vector field of the mouth area. Subsequently, similar vector fields are searched for on the lower part of the normalized face image. However, as the mouth's luminance has many times similar luminance values with its surrounding skin, an extra factor is also taken into account. That is, at every search area, the mean value of the hue component is calculated and added to the inverse distance from the mean vector fields of the mouth. Minimum values declare mouth existence.

For the extraction of the mouth points of interest (mouth corners), the hue component is also used. Based on the hue values of the mouth, the detected mouth area is binarized and small connected components, whose value is close to 0^o are discarded following the light reflection removal technique employed for the eyes. The remainder is the largest connected component which is considered as the mouth area. The leftmost and rightmost points of this area are considered as the mouth corners. An example of detected feature points is shown in Figure 27.

Figure 27: Detected facial feature points

Once the positions of the facial feature points of interest are allocated on a frontal face, tracking is the next step. In this way, gaze detection and pose estimation can be determined, not only on a single frame, but on a series of frames. In our component, tracking was done using an iterative, 3-pyramid Lucas-Kanade tracker.

4.1.3 Documentation

The component has been written using C programming in Visual Studio 6, using a lot of basic functions of the Intel OpenCV library. Avi files and real time capturing from a webcam are currently supported. The output of the component is a sequence of coordinates (included in a text file) for each one of the detected characteristics, as well as visual feedback of the result. The component input (video file/ live cam input) and outputs (facial features coordinates) will be adapted to be compliant with the framework requirements.

For correct extraction of facial characteristics, the user has to face the webcam frontally and allow, to an extent, for satisfactory lighting.

4.1.4 Requirements

The component has been developed on a Windows XP platform. Testing on different versions of Windows requires Windows XP dll files which will be included in the first release of the component. Also, OpenCV dll files and various files, as results of training procedures need to be available.

4.1.5 Status

For the time being, the component gives satisfactory results at detecting eye points but further research will have to be conducted for mouth points' localization and tracking. Also, improvements in the tracking phase have to be made as rapid movements of the user (especially in low resolution webcams) may distort the results. Accuracy is crucial in the Facial Feature Detection step, as it serves as basis for other components (gaze/pose, expression recognition) within the frames of the CALLAS project. Furthermore, research towards detecting/tracking more features (eyebrows) is planned as future work.

4.1.6 Availability

A first non-demonstrable release of the module, developed on Windows, needs further research, in order to be combined with other modules and end up with final algorithm requirements. A demo can be soon available on demand for testing purposes.

4.2 Gaze detection and pose estimation

4.2.1 Overview

This component deals with the detection of the directionality of the eye gaze. Using a USB camera or an AVI file of a person facing towards the camera, a gaze estimation can be achieved. The component highly depends on the facial feature detection component (see section 4.1) which detects and tracks eye areas and points. Based on the detection of the eye areas, gaze estimation is achieved. Also, based on the detection of facial characteristics, head pose can be estimated.

4.2.2 Description

GAZE DETECTION

In recent bibliography, most gaze detection and pose determination techniques need special hardware setup. In other cases, intrusive devices have to be worn by the user, making the system less appropriate for wide-range applications. In the current work, features are detected and tracked, allowing for a relative freedom of the user, under good lighting conditions. Under these circumstances, the gaze directionality can be approximately determined and this is enough for attention recognition purposes, as well as for general decisions regarding one's gaze. For gaze detection, the area defined by the four points around the eye is used. Prototype eye areas depicting right, left, upper and lower gaze directionalities are used to calculate mean grayscale images corresponding to each gaze direction. The areas defined by the four detected points around the eyes, are then correlated to these images. The normalized differences between the correlation values of the detected eye area with the left and right as well as upper and lower mean gaze images are calculated.

The normalized value of the horizontal and vertical gaze directionalities (conventionally, angles) consists a weighted mean between the corresponding differences found before, weighted by a factor taking into account a fraction of the total amount of intensity of both eye areas. This fraction is used to weight the gaze directionality values so that eye areas of greater luminance are favored in case of shadowed face.

POSE ESTIMATION

To estimate the pose of a face based on the features detected, orthographic projection can be assumed for a linear system to be constructed, since depth information is not necessary for pose estimation. The pose of the face is a problem of estimating the direction of the face-plane which depends on the changes of the distances between facial characteristics. Thus, if the eye and mouth centers are considered, it is possible to initialize a triangle *A,B,C,* with *A,B*

being the left and right eye centers and *C* the mouth center at the frontal view. Let *A',B'* and *C'* be the displaced positions of these points, which are considered to be known since the points are tracked. The *α,β,γ* rotation angles around the *y,z,x*-axis are:

$$\gamma = \arcsin \frac{C_x^{'}}{C_y}$$

$$\beta = \arccos \frac{A_x^{'}}{A_x \cos \gamma}$$

$$\alpha = \arccos \frac{C_y^{'}}{C_y \cos \gamma}$$

Figure 28: Shows a typical estimation of pose detection.

Figure 29: Pose Estimation Result

A chart flow of the steps followed for gaze/pose estimation is summarized in Figure 30:

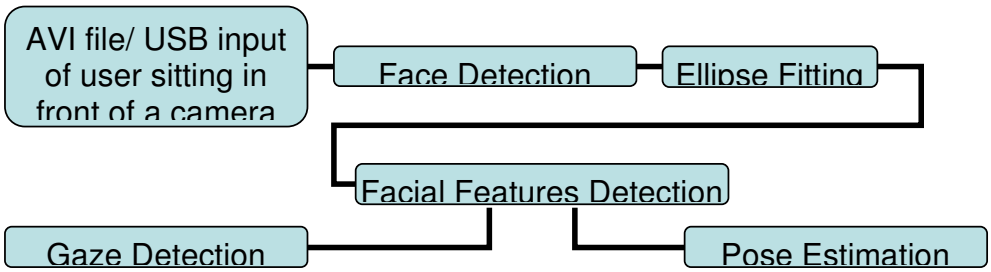


Figure 30: Gaze/pose estimation

Near future research will try to combine pose detection with gaze estimation, in order to extract conclusions regarding a user's visual attention.

4.2.3 Documentation

The component has been written using C programming in Visual Studio 6, using a lot of basic functions of the Intel OpenCV library. Avi files and real time capturing from a webcam are currently supported. The output of the component is a sequence of angles (exported to a text

file) for each one of the detected gaze/pose directions, as well as visual feedback of the result. The component input (video file/ live cam input) and outputs (gaze/pose directions) will be adapted in order to be compliant with the framework requirements.

Regarding the gaze detection part, a first version of the module is available on the Wiki of CALLAS. As gaze is being worked on independently of the facial features detection component, for practical reasons, the demo requires only a slot of the eyes area as input (see Figure 31)

Figure 31: Slot provided for gaze detection

In the directory of the demo, there is a configuration file. Among others, the configuration file can configure:

1. The size of the area that will be analyzed (more specifically, the fraction of the dimensions of the analyzed eyes area with respect to the original one)

2. The kind of input the component will use: setting “1” means that a USB cam will be used. Setting “0” means that an AVI file in the directory of the Demo will be used.

3. Filter type: The result of the directionality of the gaze is sometimes noisy. A filtered version of the result can be provided by using an FIR filter. Results are then smoothed and more accurate, at the expense of a small delay.

4. ROI is the region of interest where the user wants to place the slot to be analyzed.

For analyzing the gaze of an eyes’ area of a stored AVI, the steps to follow are the following:

1. Open the gaze_config file and make sure the video input variable is set to 0. Save the file and close it.

2. Run the gaze.exe file. You will be asked to give the name of the avi file to run. The folder already contains some sample videos, so you can use one of them as input file (e.g. eyearea1.avi). The output video shows an eye area with an arrow pointing to the gaze directionality (see Figure 32). Also, a moving window appears which is moving following the gaze.

Figure 32: Gaze Estimation Result

For analyzing the gaze of an eyes’ area using a USB cam, the steps to follow are the following:

1. Open the gaze_config file and make sure the video input variable is set to 1. Save the file and close it.

2. Run the gaze.exe file. The output window appears as an orthogonal video output showing

input from the webcam.

3. Place your face as in the sample videos (centered with the eyes under the horizontal line and the eyebrows above), quite close to the webcam (see Figure 33).

Figure 33: Placement of user's face for gaze detection using a USB cam

4. Having the output window selected with the mouse and the face correctly placed in it, looking at the center, press ENTER, stay relatively still and move your eyes right, left, up and down. In case tracking fails, you can select the output window and press ENTER again.

The results of the gaze estimation are stored in the gaze.txt file showing two normalized values of the gaze directions (angles).

The module, in general, requires satisfactory (but not really strict) lighting conditions to capture the differences of luminance in the eye area.

4.2.4 Requirements

The component has been developed on a Windows XP platform. Testing on different versions of Windows requires Windows XP dll files which are included in the first release of the component. Also, OpenCV dll files and various files, as results of training procedures need to be available.

4.2.5 Status

At the moment, the module works on slots of eyes areas. The component depends highly on the results of the facial feature detection and tracking module. There is a continuous need for improving the two modules in parallel. Especially, tracking shall be a combination between classic trackers (Lucas-Kanade) and techniques used for feature detecting, developed in ICCS. This is essential in order to achieve robust results in the case of detecting gaze using the whole frame of a person placed in front of a camera, rather than just a small part of it including its eyes' area.

Further research and experimentation is needed for a first module regarding pose estimation. However, first prototypes for testing can shortly be available for testing purposes.

4.2.6 Availability

A first demonstrable release of the demo on Windows has already been uploaded on CALLAS Wiki regarding gaze detection. Future releases of the module will provide with a demo that captures gaze directionality starting from the whole face, thus, not necessitating placing a user's face at a certain position (slot).