

# Music rendering through facial interaction

Katharina Zenke and Víctor Guerrero

Course DT2300 Sound in Interaction, KTH Stockholm

Supervisor Kjetil Falkenberg Hansen

## Abstract

In this project we were inventing a connection between face and head movements and music rendering. Therefore we tracked the face movements of a user and transmitted this information to the music renderer, which changes musical parameters in order to the movements. It is thereby possible to steer the character of a music piece just by the head. The movements were tracked by the facial recognition API of Kinect. OSC was carrying this information to the music rendering patch pDM in Pure Data. The main focus of this project lied on the realisation of emotional character changes in the music through their corresponding facial expression. Additionally there were also simple music parameter mappings with head movements.

## Background and Introduction

Many studies have been made about rendering music with body movements (e.g. [5]). But nearly all of them are treating full body movements. With this project we wanted to experience, whether it is also possible to steer music rendering through face and head movements.

Therefore our project was based on the state-of-the-art research in three different fields: face recognition, detection of emotional face expressions and emotional music rendering. There have been big developments in improving face detection software in the last years, so there were different programs available for this task. We had two of them available for our project: the Kinect from Microsoft and the freeware FaceOSC from Kyle McDonalds. We planned to compare them due to their performances and then decide for the better option. The recognition device was connected to Pure Data. Pd is a visual programming language for audio applications, which we used for the music rendering. Therefore we used a precast patch called pDM, which has been developed by Anders Friberg in KTH in 2006 [1]. Other studies provided information about emotional face expressions and about how to detect them properly [4].

In the musical part, our work was based on the research at KTH concerning emotional rendering in music pieces [2]. There have been investigations concerning the characteristics of emotions in musical pieces and about how a special emotion is transferred from the musician to the listener. Also an analysis of musical parameters available in music helped us to define the right mappings of movements and music changes.

## Set up/Work

In this project, we wanted to attach head and face movements with music parameter rendering. In a first stage we intended to match simple head movements to single musical parameter. After that we wanted to combine some emotional face expressions and the emotional equivalent in the music track.

### 1) Overview of the Setup

Our set up consisted of a face detection device which is connected to a normal user PC and sends data to a Pd patch called pDM in order to steer the music rendering (see figure 1). The three stages will be presented in particular.



Figure 1: Project set up

#### 1.1) Kinect: the face detection

##### Technology selection:

To develop the project, we looked at the different technologies available to achieve face recognition. We found that there were mainly two solutions: solutions based on 2D acquisition of images via normal camera and ones based on 3D retrieval of data. A study was carried to determine which of these options was more suitable and the next table was depicted. To do so, one representative of each was chosen: FaceOSC for the 2D based and Kinect API technology.

	2D based (FaceOSC)	3D based (Kinect face tracking API)
Features	A tool for prototyping face-based interactions which is built with the open source FaceTracker code. It's specially made for routing OSC and MIDI.	A tool to access data from Kinect and get real-time information about user's head and facial expressions.
Pro	<ul style="list-style-type: none"><li>+ code free available</li><li>+ no special hardware required</li><li>+ usable with a normal pc web cam</li><li>+ already connected to Pd</li></ul>	<ul style="list-style-type: none"><li>+ more robust</li><li>+ structures available to detect emotion</li><li>+ head orientation in 120 degrees</li><li>+ higher resolution</li></ul>
Cons	<ul style="list-style-type: none"><li>- face metrics not suitable for emotion detection</li></ul>	<ul style="list-style-type: none"><li>- not existent connection with Pd</li><li>- higher latency</li></ul>

Table 1: Comparison of face detection technologies

As we had the opportunity to work with a Kinect, the costs of the technology weren't a problem, so we decided to use it due to the pros it has. Also, it seemed a good option to develop an interface of it with Pd.

### Description of data available:

The data given by the Kinect face tracking API is the following:

- Tracking status
- 2D points:
  - 87 2D points corresponding to figure 2,
  - as well as 13 extra points that include: eye center (2), mouth corners (2), center of nose (1) and a bounding box (4) of the head.

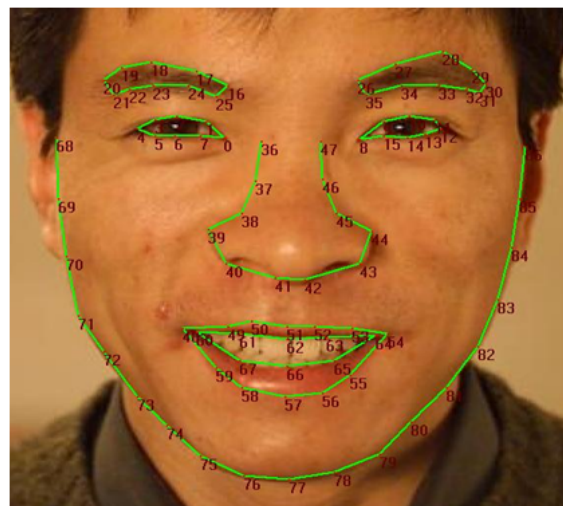





Figure 2: 2D face tracking points of Kinect

- 3D head pose: yaw, pitch and roll of face orientation
- Animation Units (AUs): these are values that represent different weights of different specific points of the face. For instance, the API offers 6 AUs, which can be seen in table 2.

AU0 Upper lip raiser	AU1 Jaw lowerer	AU2 Lip stretcher
(-1: down ↔ 1: up)	(<0: closed ↔ 1: fully open)	(-1: straight ↔ 1: stretched)
		


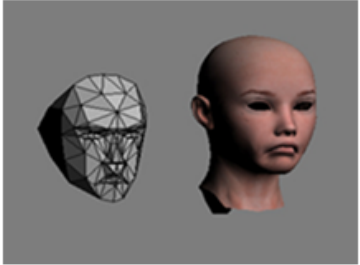
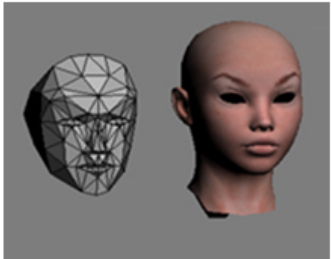
AU3 Brow lowerer	AU4 Lip corner depressor	AU5 Outer brow raiser
(-1: raised ↔ 1: lowered)	(-1: happy smile ↔ 1: sad)	(-1: lowered ↔ 1: raised)
		

Table 2: Description of AUs and its values

## 1.2) OSC: the connection between API and Pd

To communicate kinect face detection API with Pd we decided to make a binding of C++ code that access the API with Pd using OSC communication which is the usual way of communicating with Pd. We have built the kFace Pd patch (see figure 3) which exposes the next values: yaw, pitch, raw, AUs and if face is detected.

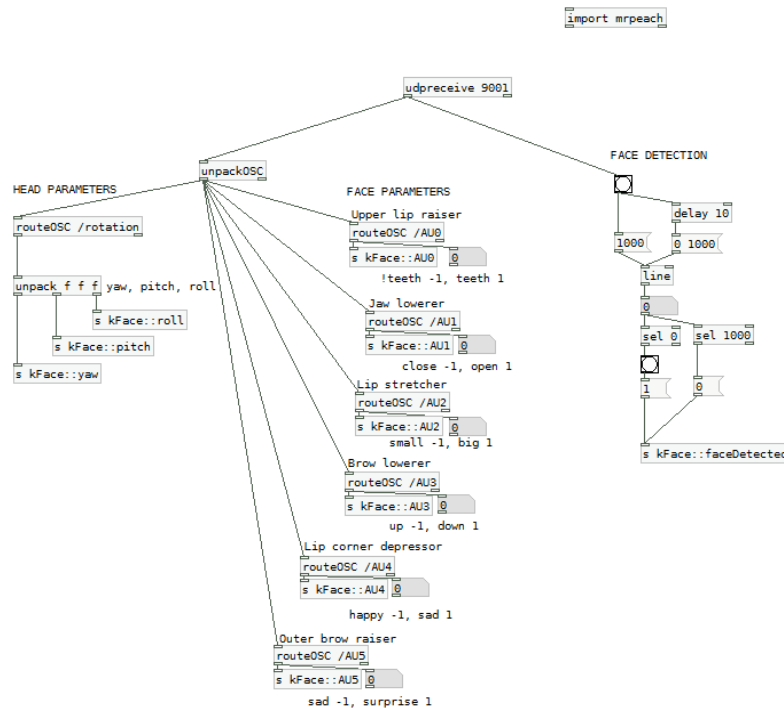


Figure 3: kFace Pd patch

### 1.3) pDM: the music renderer

At the beginning of the project our aim was the control of music through face recognition but then pDM was suggested to perform the rendering of emotions. pDM is a patch for Pd by Anders Friberg which aims to allow real-time rendering of emotions in a music track by changing its parameters according to the studies developed in KTH.

Since previous work was done in the control of music performance using face recognition we decided to go further and try to control music performance not by direct manipulation of the parameters but with an emotion mapping just allowed thanks to the work done with pDM.

pDM was created to communicate with humans offering a GUI for the information input, so we had to tweak it to allow communication with other data. Following there is the explanation of the interactions that we allow the user to do along with the modifications of the pDM path for each one of them.

## 2) Parameter mapping

First we wanted to achieve some simple one-to-one mapping of movements and music parameters. Therefore we used all possible head movements (see figure 4): horizontal (yaw) and vertical (pitch) head movements as well as the tilting of the head to the left and the right (roll). These movements were connected to the very basic musical parameters volume, tempo and pitch.

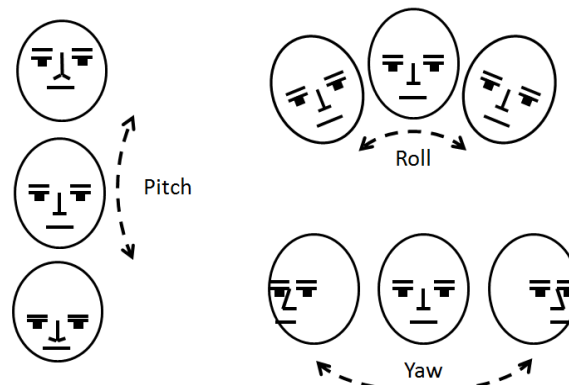


Figure 4: Possible head movements

All of these parameters were be continuous, which means that the effect on the music piece should increase if the movement was performed stronger. For example turning the head to an angle of 40° had the biggest effect on the the corresponding musical parameter.

For choosing the best combination of movements and music parameters, we tested the mapping in two simple evaluations with six subjects, who were not trained before and not related to this research field. First we let the subjects experience aevery music parameter for every different head movement. They should rate the grade of agreement of movement and parameter on a scale of 0 (no agreement) to 10 (best agreement). The result are shown in figure 5a. They show that the volume can matched good for yaw and pitch while the pitch goes well with pitch. For the tempo there were no significant preferences seen.

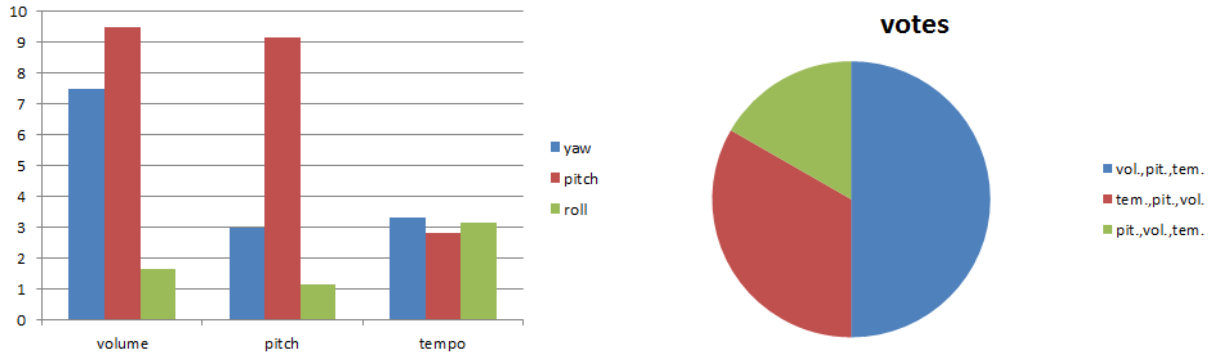


Figure 5: a) Evaluation 1 b) Evaluation 2

In a second evaluation we had three presets of matching all parameters developed out of the previous experiment. The majority of the subjects decided for mapping yaw with volume, pitch with pitch and roll with tempo (see figure 5b). We adopted this decision and set the extreme values for the mapping as seen in table 3.

Head movement	Musical parameter	Range of Movement	Range of Parameter
Yaw	Volume	$[-40^{\circ}, 40^{\circ}]$	$[-20, 20]$ dB
Pitch	Pitch	$[-20^{\circ}, 40^{\circ}]$	$[-5, +15]$ half tones
Roll	Tempo	$[-50^{\circ}, 50^{\circ}]$	*50 - /50

Table 3: Parameter matching

In order to achieve these volume, pitch and tempo changes we modified the pDM patch as followed:

a) **Volume:** Through modifying the Pdm\_expressive\_sequencer file we have another value in addition to the sound control, the “soundLevelChange” given in this case by the head’s yaw (see figure 6).

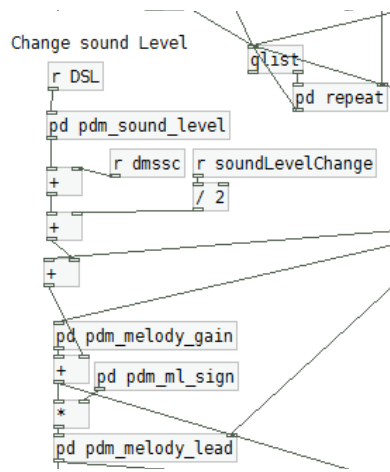


Figure 6: volume modifier

b) **Pitch:** Some patches that allow pitch shifting exist (e.g. [G09.pitchshift.pd](#)) but they are for analog output. As no pitch shifting was found for midi output we change it by adding a certain value to the midi output note in the rendering pipeline (see figure 7). We set a threshold at  $10^\circ$  in both directions to avoid changing the harmony of the music piece with every small move.

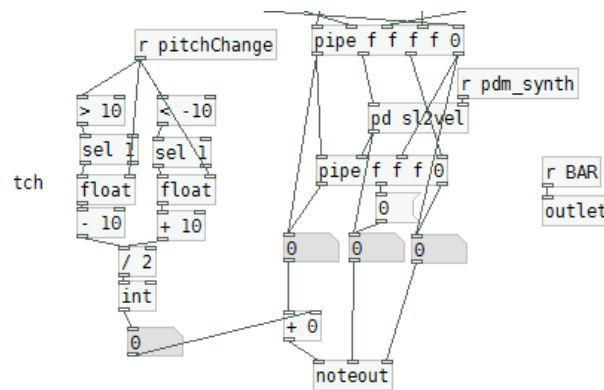


Figure 7: pitch modifier

c) **Tempo:** This parameter was directly exposed by the pDM, as is one of the parameters used to express the different emotions. The same procedure as for volume was followed (see figure 8).

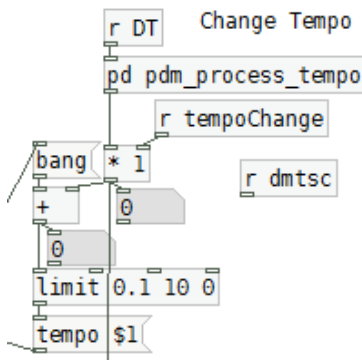


Figure 8: tempo modifier

### 3) Emotional mapping

The second part of our project contained emotion mapping. For this, four different emotions were expressed by the person's face. They were detected by Kinect and translated into their musical equivalent by using pDM. The emotions we used were: happy, sad, angry and tender. As a fifth emotion neutral was used to reset the changes when no emotion was detected.

To detect the emotions we took use of the AUs the API software is providing us (see 1.1). Therefore we had two different approaches to achieve the detection:

### a) Machine learning approach

Our first try to the emotion detection was using the machine learning software RapidMiner which provides several algorithms that allow to train neuronal networks as well as developing other techniques in this fields. As our main focus wasn't applying machine learning itself, we tried to develop a decision tree that helped us to classify the different emotions using supervised learning. To get it, we followed the next steps:

1. get data of the different emotions to provide knowledge: five emotions were made: happy, sad, angry, tender and neutral with three different grades of them (soft, neutral, exaggerated). This data was provided by both of us having a total of six samples of six values (the AUs) per each one of the emotions.

2. create the decision tree using RapidMiner: preprocess of the data was done discarding the outliers (1) and replacing its value by the mean of the other values. What we got was the following decision tree, shown in figure 9:

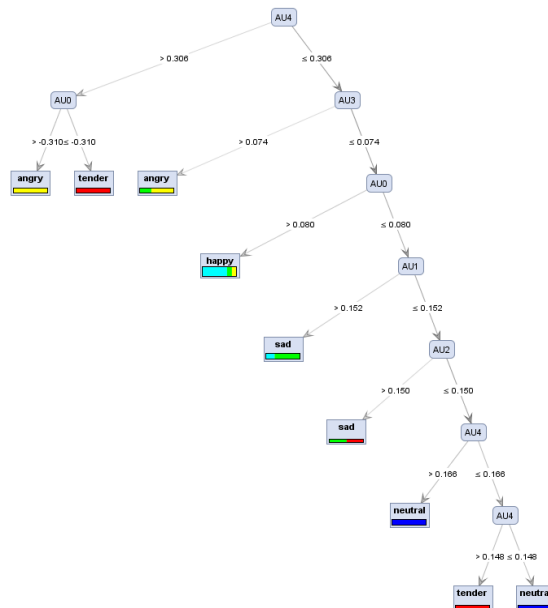


Figure 9: Decision tree by RapidMiner

It can be observed that the resulting decision tree seem to offer a good classification (just 1 of the possibilities show a 50% possibility of wrong classification).

3. evaluation of the decision tree: we tried to repeat the test again doing more faces but the results observed showed that the recognition wasn't very good and that sad wasn't well detected. In the next table the different emotions of the second approach can be observed as rows and the percentage of detection as columns.



	Happy	Angry	Sad	Tender	Neutral	(true +)
Happy	4	2				66%
Angry	2	3			1	50%
Sad			1	1	4	17%
Tender			2	4		66%
Neutral		1		1	4	66%
(false +)	33%	40%	66%	33%	55%	<b>53%</b>

Table 4: Evaluation of the Rapidminer decision tree

### b) Straightforward approach

Our second approach was done by trying to simplify the emotions in order to find out which parameters were significant for their detection. We used the data we had from the previous approach and made a first detection with the two most significant parameters of each emotion. By testing whether the emotions were well detected, we iterated and added more parameters until having the final parameters shown in table 5:

Happy	Sad	Angry	Tender
<i>AU0</i> : upper lip raised <i>AU2</i> : mouth very wide <i>pitch</i> : head tilted upwards	<i>AU0</i> : upper lip down <i>AU1</i> : mouth closed <i>AU3</i> : brows lowered <i>AU4</i> : mouth curvature changed <i>pitch</i> : head tilted downwards	<i>AU0</i> : upper lip raised <i>AU1</i> : jaw down <i>AU3</i> : brows lowered strongly <i>AU5</i> : outer brow down <i>pitch</i> : head tilted downwards	<i>AU2</i> : mouth tight <i>AU3</i> : raised brows

Table 5: AUs used for the emotion detection

For each of the different emotions we choose a pool of the more representatives AUs of the level of that emotion and studied which one had the widest range of values to allow a better control of the emotion. We tracked different degrees of each emotion to watch the range of values. In all the cases the pitch was discarded as a controller of the emotion since it was already mapped to the pitch changing.

Happy	Sad	Angry	Tender
<i>AU0</i> : (0.15,0.8) > 0.65 <b><i>AU2</i>: (0.2,1) &gt; 0.8</b>	<b><i>AU0</i>: (0.3,0.9) &gt; 0.6</b> <i>AU3</i> : (-0.05,0.3) > 0.35 <i>AU4</i> : (-0.5,0) > 0.5	<b><i>AU0</i>: (0.2,0.5) &gt; 0.3</b> <i>AU1</i> : (0.1,0.3) > 0.2 <i>AU3</i> : (0,0.3) > 0.3 <i>AU5</i> : (-0.3,-0.1) > 0.2	<i>AU2</i> : (-0.5,-0.3) > 0.2 <b><i>AU3</i>: (-0.5,-0.1) &gt; 0.4</b>

Table 6: Range of values of the parameters of each emotion

Having chosen which parameter is directly connected to the degree of each emotion, we send its value to the pDM grid of emotions and convert that value in a 2D point. The implementation in Pd of the emotion detection is as followed: we have a emotion detector which has as inputs the specified parameters in table 5 and has a bang as output. When that emotion is detected and has a value between 0-1, that will be used for the evaluation of the level of the emotion (e.g. detection of tenderness in figure 10).

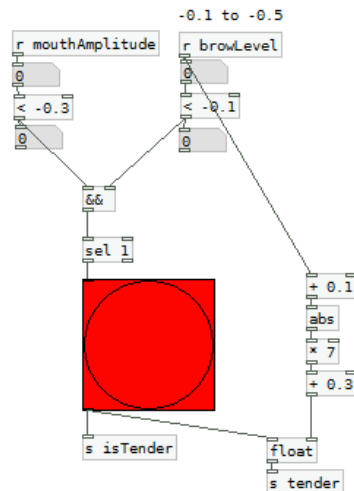


Figure 10: Decision borders for recognizing the emotion “tender”

The level of the emotion is received in the pDM patch and converted. To do so, the next modification was made which converts a value in a 2D point in the greed (see figure 11).

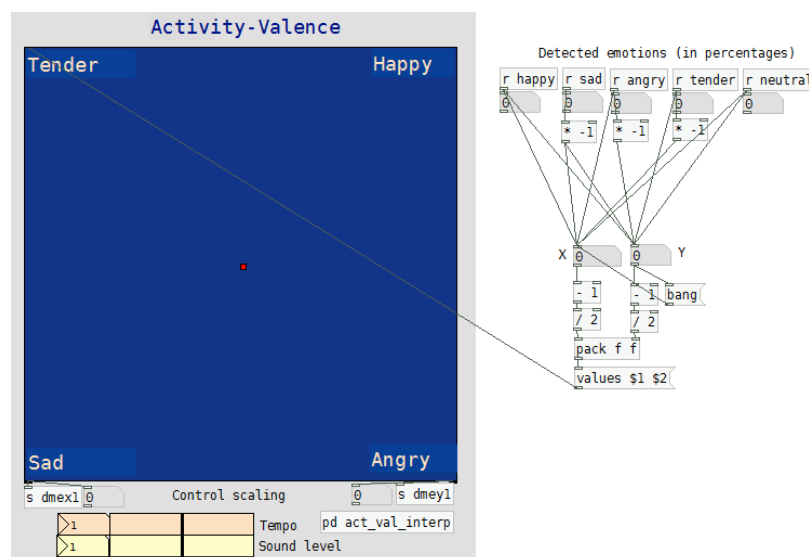


Figure 11: Activity-Valence Space and conversion

## Results

With the realization of this project we have managed to connect existent technologies to provide a new interaction with music performance allowing people to change its emotional intention as well as to control different of its parameters. We also achieved to create a interface which connects Kinect and Pd, which can be used in several applications.

After the realization of the interaction, we let people try it again with both interactions and we observed that people enjoyed more the direct manipulation of the music parameters despite thinking that the emotion rendering was more impressive because almost no one knew about the possibility to change the emotional intent of a musical piece in real-time.

This preference could be due to two main reasons:

- the emotion detection works well when you are facing the camera, but it loses accuracy when you move your head around and makes this interaction not very expressive
- it's difficult for people to control the emotion they are providing because people express emotions in different ways and they are not aware of in which way they change their face expression which each emotion and therefore are unsure about how to change the level of each emotion. This makes them feel that their behaviour is being interpreted instead of being directly controlled and makes the interaction loose connection between the user and the music.

Our results are also documented in a video [here](#):



Figure 12: QR code to project video

## Conclusion

It can be said, that emotional music rendering with using face movements is possible. The emotion detection is not as good as expected since after trying it in different set ups it seems that the values from the AUs can differ depending on the relative position of the Kinect and the user. This might be because the face detecting API was created recently for the PC version of Kinect. Microsoft still allows people to use Kinect SDK along with the old xbox360 Kinect version so due to be working with which doesn't allow the new near option that provides better values and allows people to stand as close as 20 cm the API may not work as well as expected.

As not official emotion detection is given by the API it could be interesting comparing both methods (using FaceOSC and using Kinect API). FaceOSC is based upon existent facial recognition software which has been developed for several years.

This project just includes an approach based on the face recognition isolated. Further work could include studying it along with body movements by combining both existent studies allowing better and more expressive interaction. This could be done by attaching a camera or the new Kinect for PC to user's head and having an extra camera/Kinect to detect the body movements.

## Bibliography

- [1] *pDM: An expressive sequencer with real-time control of the KTH music performance rules*, Anders Friberg, 2006, Computer Music Journal
- [2] *Emotion rendering in music: Range and characteristic values of seven musical variables*, Roberto Bresin, Anders Friberg, 2011, ScienceDirect
- [3] *Sonifications of physical quantities throughout history: a meta-study of previous mapping strategies*, Gaël Dubus, Roberto Bresin, 2011, ICAD
- [4] *Interactive control of music using emotional body expressions*, Daniel Bernhardt, Peter Robinson, 2008, CHI EA
- [5] *Expressive Control of Music and Visual Media by Full-Body Movements*, Ginevra Castellano, Roberto Bresin, Antonio Camurri, Gualtiero Volpe, 2007, NIME
- [6] *Real-time control of music performance*, Anders Friberg, Roberto Bresin
- [7] *FaceOSC*, Video on vimeo.com from the developer Kyle McDonalds, 2011 <http://vimeo.com/26098366> (last accessed on 13.10.2012)