ON AUDIO PROCESSES IN THE ARTIFICIAL INTELLIGENCE [SELF.]

CALIBRATION, ANALYSIS, RECORDING, SEGMENTATION, PLAYBACK

Øyvind Brandtsegg and Axel Tidemann
Norwegian University of Science and Technology,
Trondheim, Norway
oyvind.brandtsegg@ntnu.no, tidemann@idi.ntnu.no

Introduction

This paper describes [self.], an open source art installation that embodies artificial intelligence (AI) in order to learn, react, and respond to stimuli from its immediate environment. Biologically inspired models are implemented to achieve this behavior, and Csound is used for most parts of the audio processing involved in the system. The artificial intelligence is physically represented by a robot head, built on a modified moving head for stage lighting. Everything but the motors of the stage lighting unit was removed and a projector, camera and microphones added. No form of knowledge or grammar have been implemented in the AI, the system starts in a ``tabula rasa" state and learns everything via its own sensory channels, forming categories in a bottom-up fashion. The robot recognizes sounds and faces, and is able to recognize similar sounds, link them with the corresponding faces, and use the knowledge of past experiences to form new sentences. Since the utterances of the AI is solely based on audio and video items it has learned from the interaction with people, an insight into the learning process (i.e. what it has learned from who) can be glimpsed. This collage-like composition has guided several design choices regarding the aesthetics of the audio and video output. This paper will focus on the audio processes of the system, herein audio recording, segmentation, analysis, processing and playback.

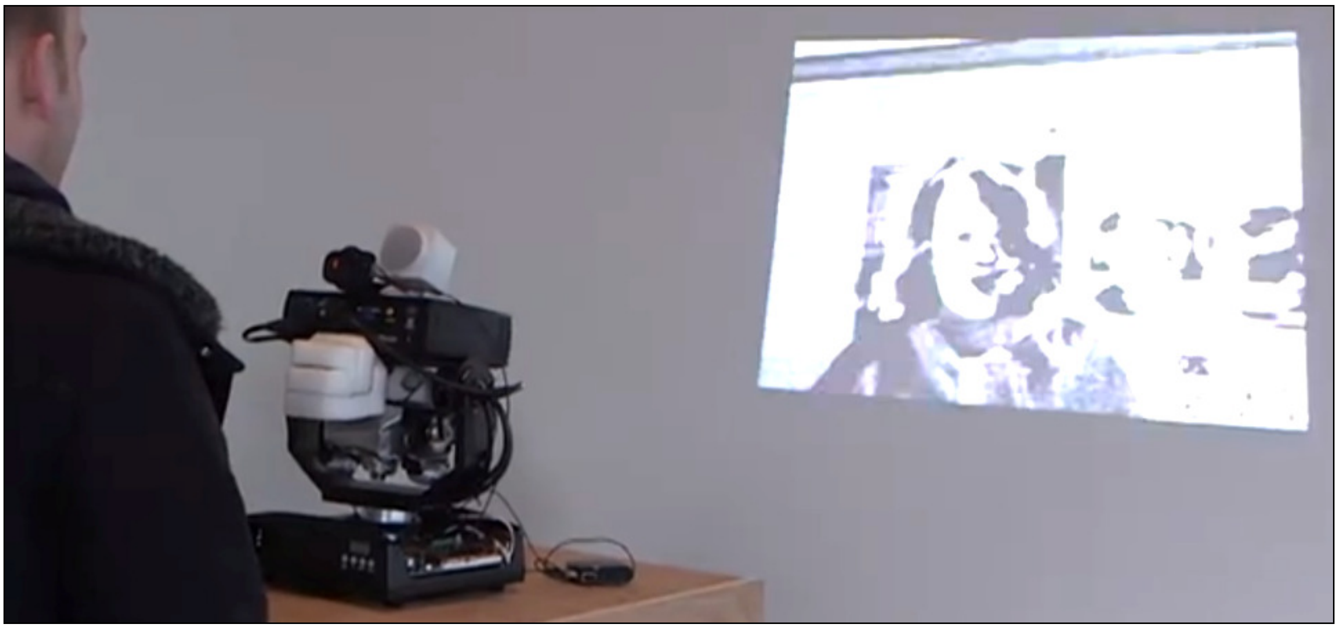

Figure 1 [self.] talking to a person

1 Background and overview

The project started out as an attempt to reflect on the effect AI will have on society in the general case, and how artificially intelligent machines will interact with human beings. The original intent was to explore a biologically inspired architecture that would strive towards the ultimate goal in AI; consciousness. The authors are well aware that this goal is out of range for the current architecture; still it has been a guiding principle in the design and conception of [self.]. The ability to learn and categorize concepts learned from interaction with the environment without a prior knowledge base is an example of such a design choice. As an artwork, [self.] plays on the relationship between technology and humans, and it relates to language, philosophy and the contemporary (over-)focus on self-realization. Ideas from Derrida (and his interpretation of the myth of Echo and Narcissus [1]), Heidegger (the essence of technology [2]), Benjamin (reproduction [3]), Rutsky [4], and Stiegler (transindividuation [5] and [6]) can be said to form a philosophical back-drop for the work. The intent is not for the artwork to relay in any rigorous manner the content and ideas of these great thinkers, but for the aesthetic object to embody aspects of and perspectives on the technology currently in question. As the techno-philosophical backdrop also relates to cultural and economic questions of access to the technology, it is only fitting that the source code[11] for the whole project is made available under an open source license.

Technical overview of [self.]

The robot has two forms of input: video (USB camera) and audio. Two microphones are mounted in a simple X-Y stereo configuration for the purpose of sensing the horizontal position of the sound source. The stereo image analysis is used to rotate the robot's head towards the position of the incoming sound, so it will turn towards the person speaking to it. This works in tandem with image analysis, where the face tracking is used for fine adjustment of the robot head orientation. The robot uses its sensory input to learn. The

[11] github.com/axeltidemann/self_dot

signals are sent to three different modules; these are face tracking (video), audio/video-association (audio and video) and sound processing (audio). The outputs of these modules are propagated to higher-level functions (``Learning'' and ``Responding'').

Audio input and output is done via Csound. Automatic calibration of the input signal level and capture of a background noise image is done at startup and during runtime. The input signal is analyzed for transients and this is used as a crude segmentation marker. Transient detection is based on amplitude slope and as such is independent of the absolute amplitude. Silent (or noisy) parts of the recording are stripped before passing on the audio to other parts of the system. A biologically inspired model of the cochlea[7] is used to analyze the raw WAVE files created by Csound. The model performs sound analysis based on a bio-mimetic cascade of asymmetric resonators with fast-acting compression. The output is called a neural activation pattern (NAP). This is somewhat similar to a spectrogram based on FFT, but different in the respect that a NAP has features that correspond to auditory physiology both in terms of the resolution and distribution of the frequency bands, and in terms of the dynamic compression algorithm used on the amplitude data. The NAP is used for learning audio concepts, as well as for building a transformation from audio to video. The AI aspects are covered more in depth in other articles by the authors ([8]and [9]) , and are only briefly recounted here, however, some contextualization is given in section III. The focus of this article is on the low level functions of audio input and segmentation as well as the aesthetic shaping of the audio output.

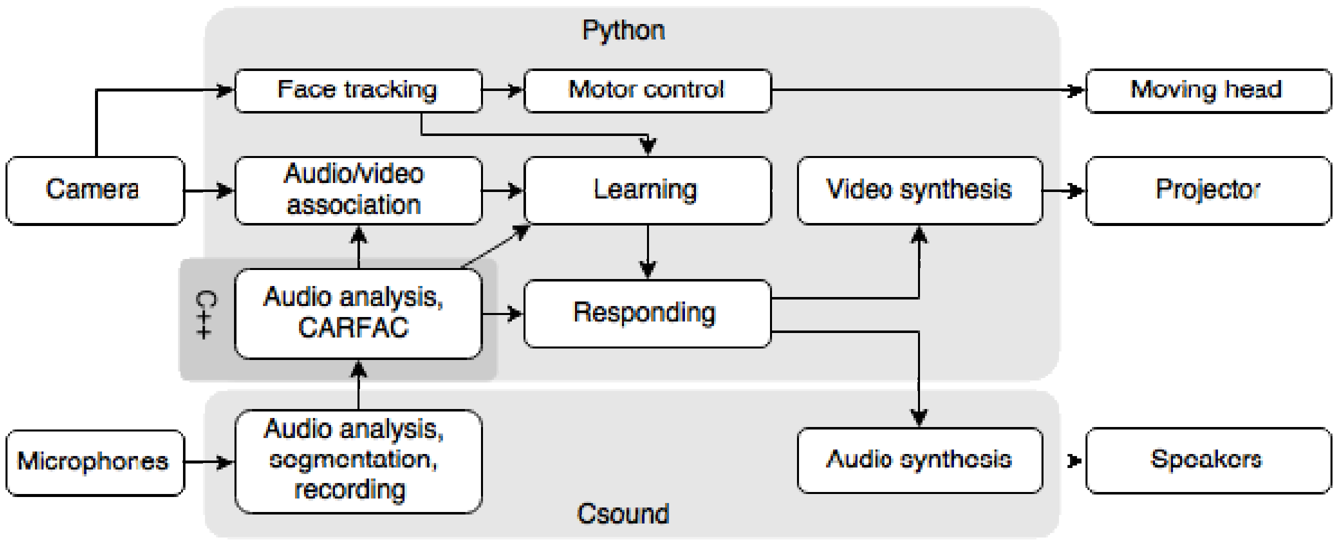

Figure 2 Overview of signals and modules, also indicating implementation languages for the different modules

2 Audio input processing

Calibration and noise reduction

The [self.] robot is intended to operate as an art installation in a public gallery, where the level of background noise may vary according to audience attendance. Similarly, the audio input level from a person speaking to [self.] may vary significantly due to the type of conversation, closeness to the microphone etc. Mechanical vibration from the robot's motors can also result in disturbances of the audio input via the microphone, and the sound that [self.] makes through the speakers can also be picked up by its microphone.

To counteract these signal irregularities, a number of techniques were combined. We use automatic calibration of the input noise floor, a spectral analysis of the background noise, suppression of [self.]'s own output (both spectrally, and as a simple amplitude ducking), and finally the input is muted whenever the robot's head need to move so much that significant vibrational bleed occurs.

The background noise print is taken by analyzing a couple of seconds of input audio, assuming that no intended communication with the robot is taking place during the analysis. The audio is analyzed via *pvsanal*, spectral frames transferred to an array, and we make a running average of the spectral content, writing the result to a Csound *ftable*. The spectral template in this *ftable* is then subtracted[12] from the spectrum of the live audio input during normal conversation. A scaling factor can be applied to set the degree of background noise filtering. With the spectral filtering in place, a measure of the level of remaining background noise is taken, using this as the general noise floor throughout the signal chain. A noise gate *ftable* is generated on the basis of the noise floor measurement, allowing sounds above the noise floor to pass unmodified, while damping softer sounds. Sound softer than 8 dB below the noise floor is muted altogether. The noise gate is implemented by using the RMS level of the input signal as a lookup to the noise gate table and using the table value as an amplitude adjustment value.

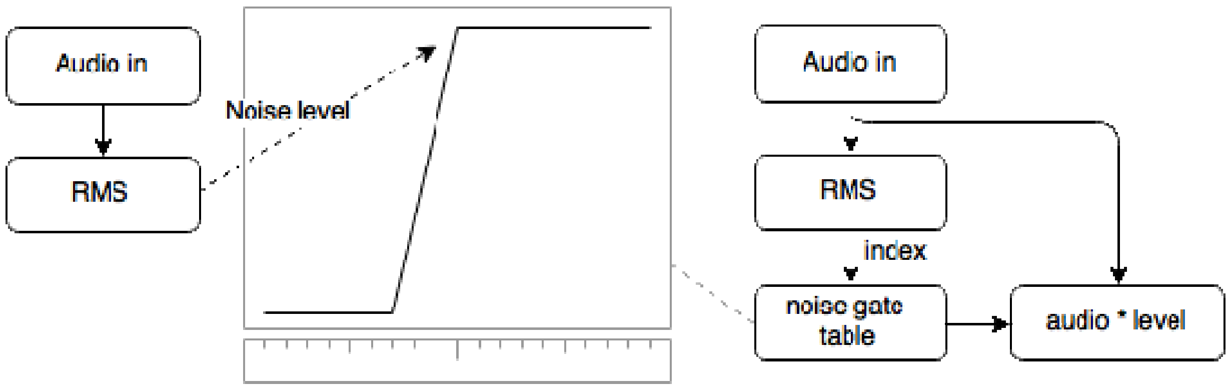

Figure 3 Analysis of noise level in input creates the noise gate shape, application of gate to the right

To enable suppression of [self.]'s own output from the speakers to the microphone, we need to measure the roundtrip latency of the audio system. This is done by generating a series of enveloped noise bursts at 1-second intervals, registering the time of the noise burst onset and comparing this to the time when the noise burst is received via audio input. The method assumes that no significant external sound is being made during the measurement and that the physical sound from the speaker can be picked up by the microphone. The resulting roundtrip latency of the audio system is written to file, so this measurement does not have to be done again until the audio configuration is changed. The Python program (*self.py*) will determine if a valid roundtrip latency file is found whenever audio calibration is initialized, and force latency measurement if the file is missing.

The audio output from [self.] is (internally) fed back to the input section, delayed according to the roundtrip latency measurement, and analyzed with *pvsanal*. The output spectrum is then subtracted from the input spectrum to suppress [self.]'s own sounds from the input.

[12] Subtracting the amplitudes of one spectrum from the other on a bin by bin basis

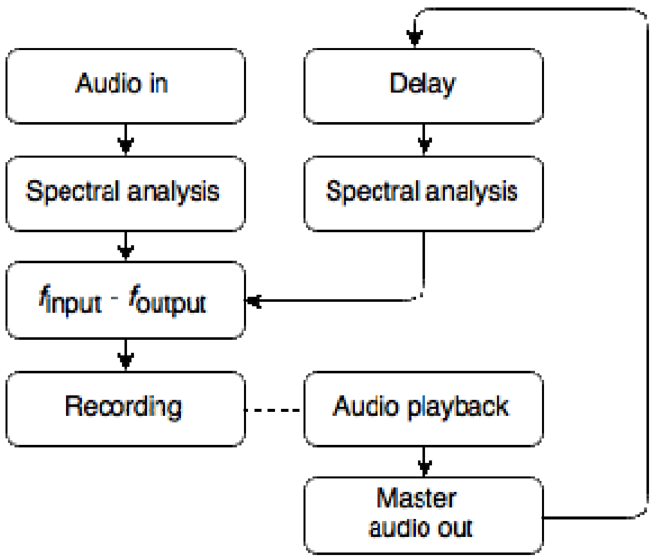

Figure 4 Subtraction of [self.]'s own output from microphone input

The audio calibration routines run at system initialization, but can also be triggered during runtime. The rationale for this being that the external conditions regarding ambient noise etc. can be expected to change drastically during the span of a day in the gallery. A large group of schoolchildren can make a relatively dense ambient noise and the relative vocal input to the microphone may vary to a high degree. Conversely, it is also common for spectators to become shy of speaking to a machine, and so they will speak quite softly. A trigger for recalibration during runtime can be the lack of sound recording for a long time, which could be due an input level or noise suppression error. Excessively long recordings will also trigger recalibration as it may be a sign of too high release thresholds.

Transient detection and audio status

Amplitude transient detection is used as a crude segmentation marker in [self.], it will initiate the recording of audio segments, and transients within recorded segments are used as "word" markers. The transient triggers are also used to update the stereo image analysis (described later), initiating movement of the robotic head towards the sound source. Transient detection is made using rate of change analysis of the filtered amplitude. The signal is conditioned with an envelope follower filter (*follow2* opcode) and mapped to a perceptual scale (dB) before transient detection. The sensitivity of the transient detection can be adjusted with an *attack threshold* parameter. To limit the amount of false trigging, some filtering methods have been implemented. When a transient is detected, the current signal level is recorded, and a *decay threshold* sets the relative negative change needed before a new transient is allowed to be registered. The envelope filtering has an adjustable *envelope release time* to smooth out fluctuations after a peak in the signal, and this works in tandem with the decay threshold. Finally, a timer is used as a secondary means to limit the rate of transients, ensuring that a certain *double trig limit* amount of time must pass after a transient has been recorded before another transient is allowed. The transient detector is wrapped as a user defined opcode, and has also been used in other projects for transient detection on a variety of signals.

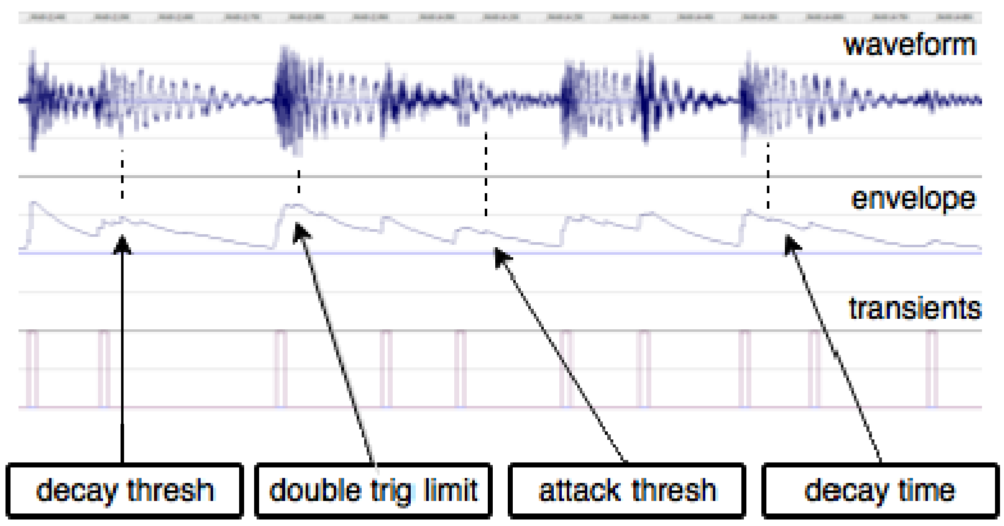

Figure 5 Transient detector. The arrows from the parameter names indicate skipped transients due to adjustment of that parameter.

Following the transient detection, an *audio status* indicator is generated. This indicates sections of realtime audio input where something interesting is happening, something we will want to record and analyze. The audio status is set to 1 when a transient is detected, keeping its value until the signal has dropped below a *release threshold* (typically -9 dB), similar to the *decay threshold* of the transient detector. In addition, we wait for an extra period of time (*status release time*) before setting the status to zero. Audio recording follows the audio status indicator. In this manner, several transients are allowed within one audio status segment (one audio file), and we can keep a contiguous recording of one complete utterance (sentence) received by [self.]. This mechanism is intendedly low level and provides a course selection of sounds that we may want to keep. Higher level processes later in the signal chain refines this selection, the objective at this point is to grab everything that may be of interest.

Stereo image analysis

To allow the robot to turn towards the person speaking to it, an analysis of the stereo image of the input sound is made. If the input sound is coming from the left, the robot will turn in that direction, and vice versa if the sound is coming from the right. The pan position value is a float in range 0.0 to 1.0. Since the utility of this analysis is to trigger relative movement of the robot head, we will use a center pan position of 0.5 to indicate that no adjustment to the robot head position is necessary. An imbalance in the stereo image occurring in tandem with an amplitude transient will trigger relative movement. The amplitude transient is thus used to limit the rate of spatial-induced movement. Otherwise, the head would spin rather erratically and nervously. Two methods of stereo image analysis have been tested, one based on relative amplitude analysis of the left and right channel with this formula:

$$\left(\frac{amp_L - amp_R}{amp_L + amp_R} + 1\right) * 0.5$$

The other spatial analysis method is based on correlation between audio signals at different fractions of the expected time delay between the two input channels, similar to the interaural time difference (ITD) in human hearing. Based on a microphone spacing of 20 cm, and using 340 m/s as an approximation of the speed of sound, we have a maximum ITD of 0.6 milliseconds. For practical purposes, we assumed that a resolution of 10 steps in the resolution of the stereo position was sufficient. 10 copies of the *left* signal were made, each being delayed by a fraction of the maximum IDT. The correlation between each delayed copy and a reference signal (input *right*, delayed by ITD*0.5) were calculated by writing each signal to a temporary table, multiplying each delayed signal with the reference signal and use the one with the highest sum as the estimate of the stereo position of the sound.

The correlation based method was significantly more prone to noise and spurious peaks, and also more expensive to calculate than the amplitude difference method. We assume that the method could be refined by filtering the audio signal to remove frequency content outside of the range where ITD cues are reliable (< 1500 Hz). For the current version of [self.] the amplitude difference method was used, as it gave sufficient precision of stereo position analysis for our purposes.

Recording and segmentation

Whenever someone speaks to [self.], it records the received sound to file for further processing. Audio recording is initiated by the audio status indicator and continues as long as the status indicator is positive. The sound files are automatically named using the date and time of the start of the recording. To accompany each sound file, a marker file with the same name is generated, containing information about the time of each detected transient during the recording. The marker file also contains additional segment info, like interquartile mean values for amplitude, pitch and centroid for each segment, maximum amplitude and total duration. These values are later used to determine the most significant (stressed) word in a sentence, and determining whether the utterance was a question (pitch profile). This marker file is written by Python, and the data is transferred using common functions of the CsoundAPI. A separate continuous pitch data file is also recorded, to be used for pitch synchronous granular synthesis playback of the sound segments (see later description of the output sound characteristics of [self.]).

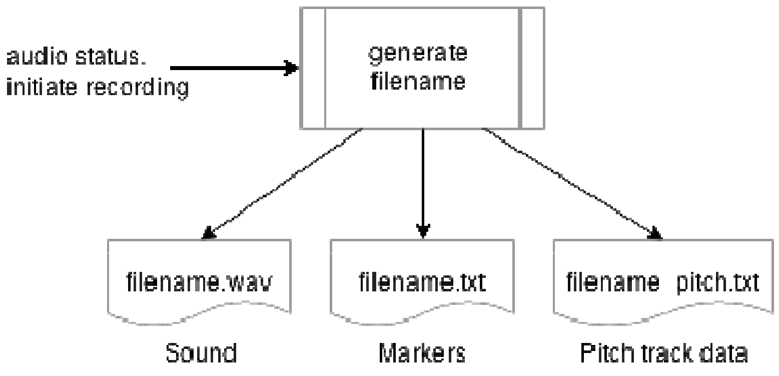

Figure 6 Each recording consist of a set of 3 files

In spite of the signal conditioning previously described, there is a realistic chance of intermittently recording background noise. Also, since the audio status indicator requires a quite generous drop of signal intensity followed by a release time, we will (always) have some silence (or noise) at the end of recordings. Similarly, we may have periods of silence or noise in the middle of a recording, for example as pauses between words in a sentence. To minimize the amount of audio that needs to be analyzed and handled by the system, we want to strip off these unwanted sections of the recordings, and we want to discard altogether recordings that (after the fact) shows signs of being accidentally recorded noise. Now, there is an inherent problem here, since we want [self.] to be able to respond as quickly as possible to a received sound, we want to do segmentation and recording in realtime, but some indicators of the validity of a recording can only be determined after the complete sound segment has been recorded. For example, if an intermittent noise (like the closing of a door or clacking of a heel) initiates recording, and the recording continues for N seconds (*status release*), we would want to discard this recording, but there is no way of predicting if a valid utterance has been made during this period until the recording stops. Similarly, the duration of the silence (or low level noise) at the end of a recording cannot be determined to be at the end of a recording until the recording has stopped. At the same time, we want to allow [self.] to analyze and process the sound as quickly as possible to minimize the total response time, so we would prefer to filter out silences and noise as soon as possible, preferably even before writing the audio to file. To facilitate this, incoming audio is temporarily written to a buffer (table), skipping sections that has a signal level dropping below a *record threshold* (set generously low, due to the previous noise gating). A running time indicator for the recorded segments and the skipped segments is kept, so that we could restore the skipped silences if needed. When *audio status* finally drops to zero, we have a compact recording with silences stripped in the buffer, and a k-rate loop is used to quickly write this buffer to file.

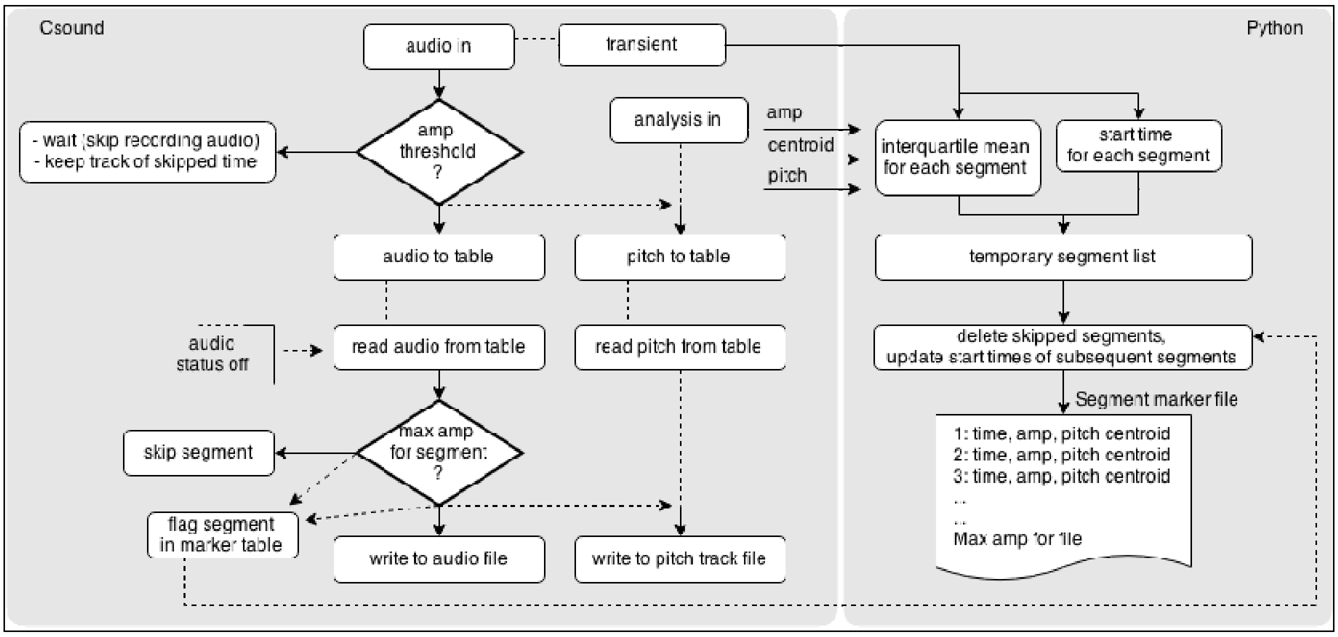
Csound
Python
audio in
transient
amp threshold ?
analysis in
amp
centroid
pitch
- wait (skip recording audio)
- keep track of skipped time
interquartile mean for each segment
start time for each segment
audio to table
pitch to table
temporary segment list
audio status off
read audio from table
read pitch from table
delete skipped segments, update start times of subsequent segments
Segment marker file
1: time, amp, pitch centroid
2: time, amp, pitch centroid
3: time, amp, pitch centroid
Max amp for file
skip segment
max amp for segment ?
flag segment in marker table
write to audio file
write to pitch track file

Figure 7 Temporary recording to buffer, clean up (optimization) of audio, pitch data, and marker file

The maximum signal level of all recorded segments in the file is saved, this serves two purposes: if the maximum signal is relatively low (currently set to 20dB above the noise floor), the whole recording is deleted before being written to file (as we can assume it was an accidental recording of intermittent noise). Secondly, the max level is written to the marker file to be used for normalization purposes during later playback of this sound. The pitch track signal is subjected to a similar buffering as the audio signal and is kept or discarded according to the same rules. The objective is to keep the pitch analysis synchronized with the audio recording during these optimizations. Segment markers are written to lists in Python, and a marker table is kept in Csound indicating segments to be kept or discarded. When recording stops, the segment marker list in

Python is filtered according to the Csound marker table, discarding and updating entries as needed. The valid segment markers are then written to the marker file along with the total duration and the max amplitude for the file. The whole recording optimization operation completes within 1 k-rate period after audio status has been set to zero, and the completed file is passed on to further analysis, as described in the next section.

Audio analysis, Neural Activation Patterns, CARFAC

A number of spectral analysis methods was implemented based on the *timbre toolbox* [10], including *spread, skewness, kurtosis, flatness, crest* and *flux* in addition to the *centroid* already available in Csound. Several pitch tracking methods *(ptrack, plltrack, pitchamdf*) was tested and an additional methods has been implemented based on *epoch analysis* [11]. The raw spectral data was also transferred to Python together with the analysis data, with the intention of using the full set of analysis tracks as material for identification and learning. Although this data provided initial stepping stones and promising results for learning and recognition, it did not result in robust and reliable learning methods for the AI.

The current approach to analysis for recognition and classification is the CARFAC method [7] (Cascade of Asymmetric Resonators with Fast-Acting Compression), housed in an external C++ library and invoked as a separate subprogram running once on each sound file. This biologically inspired model of the cochlea has been utilized earlier for recognizing sounds using machine learning. The output is called a neural activation pattern (NAP). Like a spectrum based on FFT, a NAP has channels for different frequency bands with a corresponding intensity value over time. The distribution of frequency bands are based on the model of the cochlea, as is the fast acting compression method used in the intensity analysis. The compression method acts on each band both separately and coupled, on several different time scales and as such it does a good job of modeling some of the perceptual masking effects.

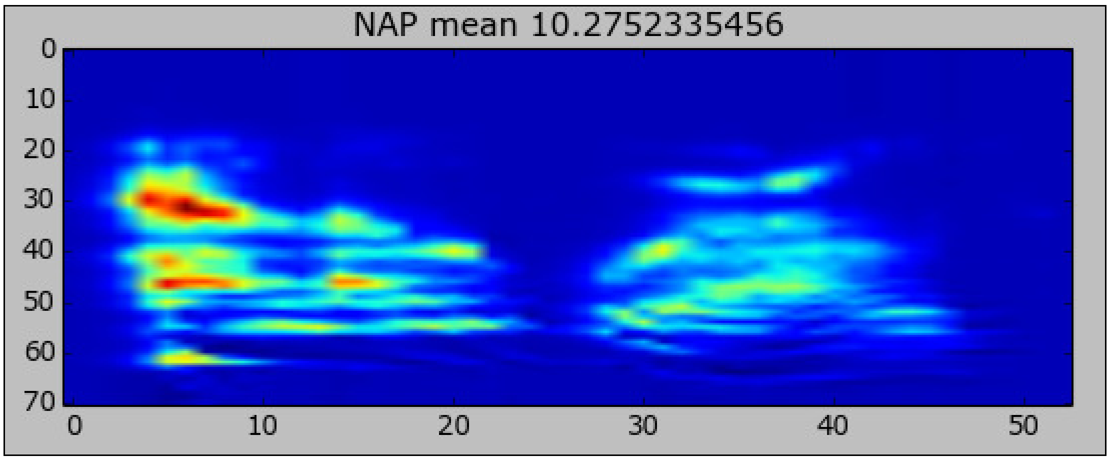

Figure 8 Example NAP, [self.] hearing the phrase “hello world”

The NAPs play a central part of the cognitive abilities of [self.], as is described in the following section. The NAP itself also serves as a filtering mechanism, as the system will discard sounds that have a very low overall neural activity. This has turned out to work well in conjunction with the techniques described in the earlier section.

3 The AI, learning and responding

Learning

Learning is the core element of [self.]. The robot clusters similar sounds together, based on the length and the similarity of the NAPs. The clustering is based on the Hamming distance [12] between the sounds of roughly the same length. This module also learns to recognize faces, and associate sounds with the face that uttered them. The facial recognition is done by a Support Vector Machine [13]. The various techniques implemented are chosen based on empirical experimentation. The live video input is also used to train an Echo State Network (ESN,[14]), which is a neural network with a huge hidden layer and fast training algorithm. The ESN was trained to predict a sequence of images based on the NAP of the corresponding sound. The network is then used generatively during [self.]'s response.

The recorded audio segments are analyzed with regards to the contexts in which the sound was perceived. This creates a set of quality dimensions for each sound and [self.] uses these as modes of association when it creates an output statement. One such context is the face recognition ("who said what"); another is the context of the sentence in which the sound was perceived. The position of the sound in a sentence (e.g. in the beginning or end) is also recorded, and there are also contexts for longer time spans (e.g. sounds perceived within a couple of minutes or within an hour). Sound duration and a similarity measure (Hamming distance) to other sound classes are used as yet another such context or quality dimension. In this manner, a multi-dimensional web of associations is built, where a dynamic weight can be applied to each. The balancing and weighting of the different associations is done in a manner inspired by fuzzy logic, where an item can have partial membership in each of the different relevant contexts. The most significant (loudest) sound in an input sentence is flagged, and this is used as the seed for the associations of [self.] when generating a response.

The initial weights for the association contexts are manually adjusted, but [self.] will evolve new sets of weights (during the *dream* state at night) using a genetic algorithm. The purpose of automatic evolution of weights is to tune the bias of association relative to the balance of features in the collected database of sounds. In addition, the dream state iterates over the sounds in each category formed in the audiovisual memory of [self.], and removes sounds that are too dissimilar to the other sounds in the same category. This can be regarded as a mental hygiene process of sorts.

Responding

When a new sound (i.e. the NAP) reaches the responding module, the most similar sound in the audio-visual memory is retrieved, based on the Hamming distance to the other sounds of roughly same length. Using the most significant sound in the latest input sentence, [self.] looks for associations to this sound, to create a response sentence. The associations are created by looking up the different contexts as described above. In this manner a chain of associations is created, and this is used as a repository of sounds to construct a response. A sound's original position in an input sentence is also used as a filter or rule for selection and placement of sounds in a responding statement. The output from [self.] will be in the form of audio and video, where the audio is a processed version of the recorded sound, and the video is a sequence of live generated images from the ESN driven by the corresponding sound (i.e. NAP). The video is this affected by *how closely the original NAP corresponds to the new NAP*. Sounds that are relatively similar will produce sharp images where you can easily recognize the original training situation, while imperfections in the relation between input and memorized sounds will create more blurry images, noise, and in some cases oversaturated silhouettes.

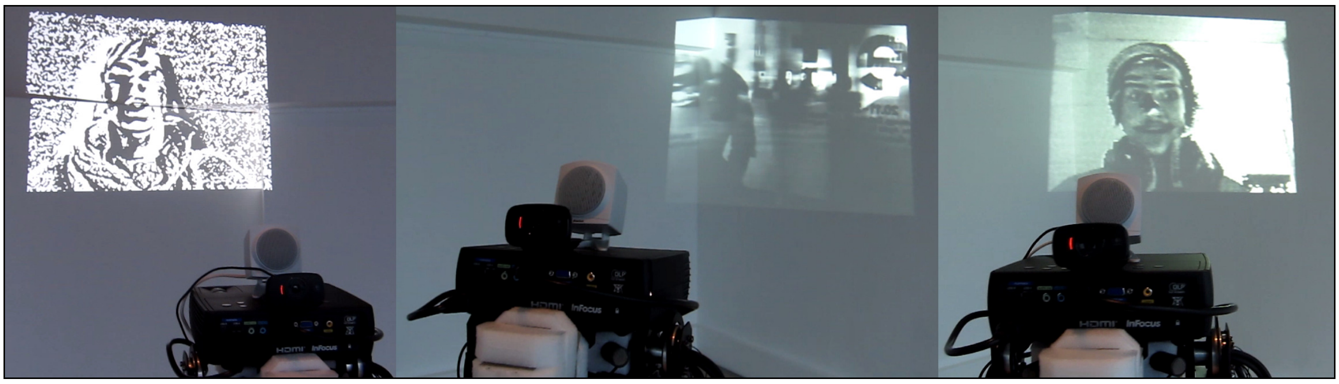

Figure 9 Examples of video output

4 Audio output processing

The audio output of [self.] is based on the recordings of the sounds it has learned, assembled into sentences by the association techniques described previously. Together with the neural network-generated video, this kind of collage can give an insight into the learning process of the AI by connecting each sound to the person it has learned the sound from. We wanted to give [self.] a characteristic voice quality while retaining recognizable features of the original input sound. To enable this, a number of alternatives for the voice of [self.] were developed, each providing a gentle twist to the sound file being played back. In addition to the main voice of [self.], we wanted to have an auditory display of the AI's secondary associations. We can think of this as "the things [self.] thinks of while considering its next response". The secondary associations are sounds that are related to the words in the response, but not quite top matches. These secondary associations are played back by an auxiliary speaker, a motorized ultrasonic beam speaker, projecting the sound spatially within the room, to create a "cloud" of sound, as if we somehow are inside the mind of [self.]

The available voice types for [self.] are made with spectral[13] techniques and with particle synthesis [15], [16]. The spectral voices utilize small amounts of frequency scaling and frequency shifts, in opposing values so the shifting to some degree will counteract the effect of the scaling. The purpose of this is to gently tilt the composition of the spectrum while retaining a quite natural sound. As an example, one of the voice types scales the frequencies up one semitone, while shifting all frequencies above 200 Hz down by 50 Hz. Some voice types may have individual shifting of several frequency regions, for example +600 Hz in the region from 1500 Hz and up, and +100 Hz in the region from 700 Hz to 1500 Hz. Some voice types also utilize a small randomization of the frequency values of the spectrum, creating a more robotic character. The sound file is analyzed and loaded into a *pvsbuffer*, so time modifications are available independently of pitch. Time modifications are currently only used for the secondary associations, providing them with a more sustained, slower and thoughtful,

[13] www.csounds.com/manual/html/SpectralRealTime.html

ethereal sound. We could conceivably also use fluctuating time modification as a means of subtle expressive phrasing for the main voice. Typically, the spectral voices also utilize small amounts of spectral blurring (in the range 0.15 - 0.2 secs) and smoothing (amplitude smoothing in range 0.2 - 0.4 secs and frequency smoothing generally very low, around 0.06 secs).

Particle[14] synthesis is used for two different kinds of voices, one which we may call "slowdown but at original speed" and the other based on pitch synchronous granular synthesis [17]. The basic principle of the first of these is to use 4 different *source waveforms* (containing the same sound, but using individual *samplepos* pointers, which in turn determine the position in the sound from where to start reading a grain of sound). The *samplepos* pointer progresses through the sound at half the original speed, and when it is lagging significantly behind the "real" time of the sound, we crossfade to another source waveform (via *channel masking*) and reset the *samplepos* pointer. With 4 source waveforms slowing down in interlapping patterns, we can create an illusion of the sound moving slower while playback still progress through the original recording at the original speed. The original speed can also be modified, independently of the overlapping slowdown procedure, creating additional time stretching effects. For this effect, we used quite high grain rates (120 Hz) and long grains (relative grain duration of 4.0). To minimize the artifacts of the time stretching, we utilize a gentle random deviation (approximately 1% of the duration of the source wave file) to the *samplepos* pointers, and we introduce a small irregularity in the grain placement (slightly towards asynchronous granular synthesis). The grain asynchrony is achieved both by using *grain distribution* (0.3) and by intermittently adjusting the grain clock via soft synchronization pulses to the *partikkel sync* input. This kind of time manipulation techniques can also be used to create reverb-type effect, as referenced in [18]

The second particle-based voice type is generated with pitch synchronous granular synthesis. Here we use pitch tracking of the source sound to control the grain rate. This allows modification of the voice formants without altering the perceived pitch of the sound. To some degree, the perceived pitch can also be altered independently, but we use this method to create a gentle octaviation effect (dividing the grain rate by 2). To adjust the presence of the octaviation effect, we can alter the grain durations (here using relative grain duration of 1.5) and the grain shape (using a fast attack on each grain and applying an exponential decay). The effect is softened somewhat by using a small amount of grain distribution (0.1).

Voice type is selectable on an event basis, and it is possible to use same range of voice types for the secondary as for the main voice. This aided in experimentation with the total aesthetic appearance during final mounting in gallery. In the exhibited version, we used a spectral voice type for the main voice and the slowdown partikkel voice type for the secondary voice. The secondary voice used a relative playback speed of 0.7 while the main voice used no time modification. The secondary voice was additionally treated with delay and reverb.

[14] www.csounds.com/manual/html/partikkel.html

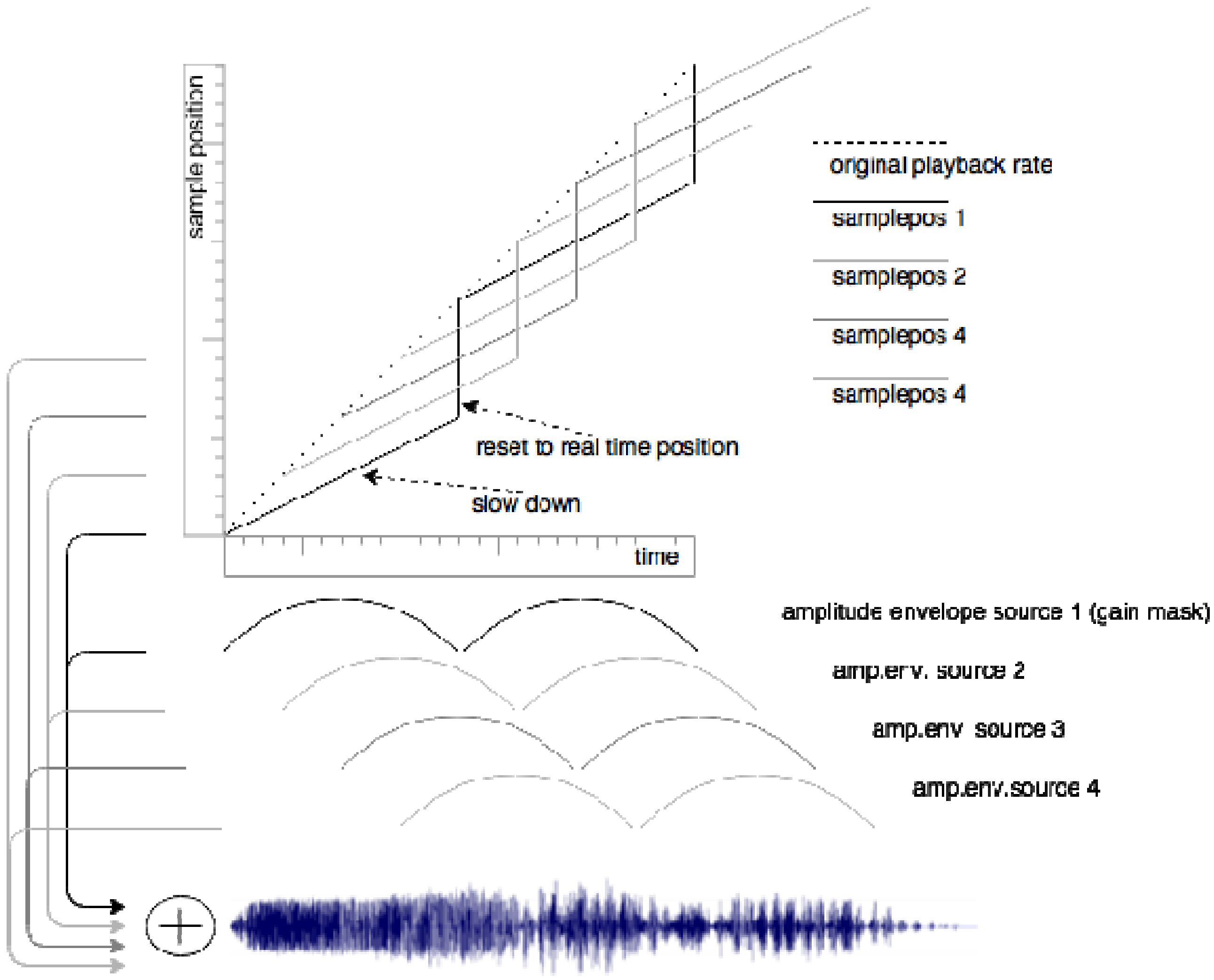

Figure 10 Read pointers in 4 source waveforms to create a slowdown effect at original speed

As an auditory backdrop to the voice sounds, we created an ambient texture using particle synthesis. This texture uses randomly reordered cut-ups from [self.]'s recently recorded sounds, played back in an asynchronous granular manner. Two parallel particle generators were used, with slightly different parameters. A low grain rate (12 Hz) was used with random deviation to the grain rate, relatively short grains (0.2) with clear attacks. Grain transpose was modulated in the range of +/- one semitone. The second particle generator used somewhat higher grain rates and generally longer and softer grains, with separate gain- and channel masking. Every few seconds one of the 4 source waveforms for the particle generators was replaced, choosing from the 10 latest input recordings. The intended effect is to create a fragmented background chatter based on recently perceived sounds. This assumed to create a sonic illustration of the ongoing mental processing of recent events. The stereo output of this ambient texture was sent to the main and secondary speakers. Even though the speaker setup does not really provide a clear stereo image in the traditional sense, it creates the impression of a spatially moving sound field.

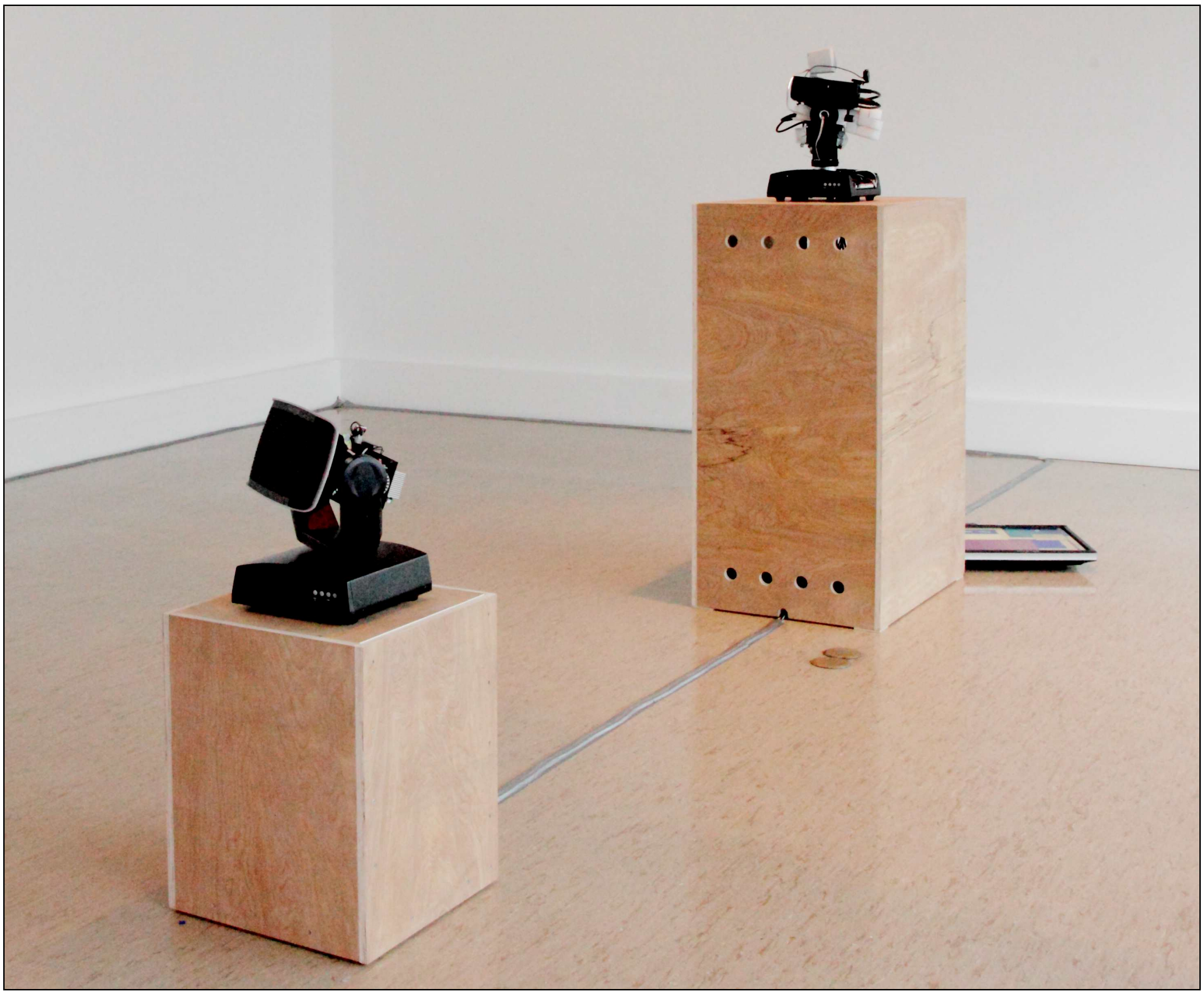

Figure 11 Main robot head, with secondary speaker in front

Conclusion

We have described the art installation and artificial intelligence [self.], and its biologically inspired models of listening, learning and responding. We have also looked more closely into the audio processing methods implemented as part of the inner workings of [self.]. The sound recognition and classification coupled with face recognition allows a multidimensional association web to be spun, creating weighted connections between sounds [self.] has heard and learned, and it utilize these associations to assemble its output statements. As no dictionary, grammar or other body of knowledge is given [self.] from its inception, it has to learn communication only via its sensory channels. The output statements of the AI thus also bear a strong imprint of whomever it has learned the different sounds and concepts from. [self.] as an artwork intends to reflect on the relation between technology and human beings. As one of several philosophical backdrops we've used Derrida's recollection of the Greek myth of Echo and Narcissus, providing an insight into what constitutes a self, and possibly in this context making an analogy between Echo and modern technology. As Echo, [self.] can only produce what it has heard, but also as Echo's clever manipulations and selections of phrases to be repeated, [self.] may be able to express *something else* with its collage-like recollections of impressions, something that may be the seed of the constitution of a true *self*.

With regards to the audio processing techniques involved in this artwork, we have given detailed insights into the recording and segmentation methods, as well as the analysis of the spatial position of incoming sound. In the classification and sonic analysis, we have used the CARFAC technique to create neural activation patterns for each sound segment. These are also used to find similarity measures between sound classes and to train audio-to-video associative neural networks.

The design of the voices of [self.] has been given attention in terms of the signal processing and synthesis techniques used. Two kinds of voice function have been implemented: One main voice, creating a close-to-natural reproduction of recorded sound but with a slight twist to add [self.]'s own character to the output. The secondary voice is slower in pace and spatialized in the room, reflecting the thought processes underlying or paralleling the main voice's output. An ambient texture is created with granular techniques on a selection of recent sounds [self.] has experienced. This texture adds a timbral and spatial counterpoint to the more straightforward conversation-type sounds, and it acts as another layer of sonic interaction through the slow-paced updating of grain source waves from the recent sonic activity.

The art installation [self.] has a mode of interaction and a way of learning that is in some ways similar to how a small child interacts with the world. In this manner it reflects its own status as a relatively newborn entity, with an abundance of room for improvement. The authors intend to follow up the work started here, and build more elaborate and potentially more powerful intelligence for the next [self.]. The audio analysis techniques can be expanded and completed by adding perceptual feature analysis based on spectral statistics, and most possibly also features extracted from the stabilized auditory images used by Lyon [7]. A finer segmentation of perceived sounds can also aid [self.] in grasping a common *stem* or *root* for sounds pointing to the same semantic entity, but used in different contexts. The currently implemented intelligence does allow for learning and interaction but not so much for internal reflection. Also it does not have an inbuilt *desire* to develop further, outside the limits currently set for it. Systems for positive and negative feedback on different kinds of behavior are examples of basic building blocks needed for the entity to generate a *self-drive* towards changing its own behavior or its own conditions for interacting with the world.

Acknowledgements

The authors would like to thank the Arts Council Norway, Trondheim Muncipality and NTNU for financial support. Special thanks to Stefan Gemzell of Ljusdesign[15] for giving us for free the first moving head to experiment on, and to Robin Støckert of Soundscape Studios[16] for providing us with the rest of the hardware for the robot. We would also like to thank curator Anne Gro Erikstad for inviting us to create the artwork [self.] for its first exhibition at Trøndelag Senter For Samtidskunst[17] in November 2014.

[15] www.ljusdesign.se/en/
[16] www.soundscape-studios.no/english/products/products.htm
[17] www.samtidskunst.no/english/

References

[1] DeArmitt, P., Resonances of Echo: A Derridean Allegory. Mosaic (Winnipeg), 2009. 42(2): p. 89.

[2] Heidegger, M., The question concerning technology. Technology and values: Essential readings, 1954: p. 99-113.

[3] Benjamin, W., The work of art in the age of mechanical reproduction. 2008: Penguin UK.

[4] Rutsky, R.L., High techne: Art and technology from the machine aesthetic to the posthuman. Vol. 2. 1999: U of Minnesota Press.

[5] Stiegler, B., Transindividuation, I. Rogoff, Editor. 2010, e-flux, 311 East Broadway, New York: e-flux journal #14.

[6] Stiegler, B., Technics and Time: The fault of Epimetheus. 1998: Stanford University Press.

[7] Lyon, R.F., Using a Cascade of Asymmetric Resonators with Fast-Acting Compression as a Cochlear Model for Machine-Hearing Applications, in Autumn Meeting of the Acoustical Society of Japan. 2011. p. 509-512.

[8] Tidemann, A. and Ø. Brandtsegg. [self.]: an Interactive Art Installation that Embodies Artifcial Intelligence and Creativity. in ACM Cognition + Creativity. 2015.

[9] Tidemann, A. and Ø. Brandtsegg. [self.]: Realization / Art Installation / Artificial Intelligence. in International Conference on Entertainment Computing 2015.

[10] Peeters, G., et al., The timbre toolbox: Extracting acoustic descriptors from musical signals. Journal of The Acoustical Society Of America, 2011. 130: p. 2902-2916.

[11] Yegnanarayana, B. and S. Gangashetty, Epoch-based analysis of speech signals. Sadhana, 2011. 36(5): p. 651-697.

[12] Hamming, R.W., Error detecting and error correcting codes. Bell System Technical Journal, The, 1950. 29(2): p. 147-160.

[13] Cortes, C. and V. Vapnik, Support-vector networks. Machine Learning, 1995. 20(3): p. 273-297.

[14] Jaeger, H. and H. Haas, Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. Science, 2004. 304(5667): p. 78-80.

[15] Brandtsegg, Ø., S. Saue, and T. JOHANSEN, Particle synthesis–a unified model for granular synthesis. Accepted paper at Linux Audio Conference, 2011.

[16] Roads, C., Microsound. 2001: Mit Press.

[17] Poli, G.D. and A. Piccialli, Pitch-synchronous granular synthesis, in Representations of musical signals, P. Giovanni De, P. Aldo, and R. Curtis, Editors. 1991, MIT Press. p. 187-219.

[18] Ervik, K. and Ø. Brandtsegg, Creating reverb effects using granular synthesis, in Ways Ahead: Proceedings of the First International Csound Conference, J. Heintz, A. Hofmann, and I. McCurdy, Editors. 2013, Cambridge Scholars Publishing. p. 181-187.