universität wien

MASTERARBEIT

Titel der Masterarbeit

”CATOS
(Computer Aided Training/Observing System) “

Verfasser

Jinook Oh, BSc

angestrebter akademischer Grad

Master of Science (MSc)

Wien, 2013

Studienkennzahl lt. Studienblatt:	A 066 013
Studienrichtung lt. Studienblatt:	MEi:CogSci - Middle European interdisciplinary master programme in Cognitive Science
Betreuerin / Betreuer:	Prof. Tecumseh Fitch

Acknowledgements

I would like to express my deepest gratitude to my supervisor, W. Tecumseh Fitch. He gave me lots of helpful comments and inspirations on many different aspects, encouraged and supported me mentally and substantially throughout this study.

Furthermore, I thank to Dr. Marek Nagy in faculty of Mathematics, Physics and Informatics of Comenius University. He led me though building and training Hidden Markov models for WAV file recognition in this study with instructions and useful comments. Also, I appreciate discussions and many insightful comments about AI and automatic systems from Dr. Paolo Petta in Austrian Research Institute For Artificial Intelligence.

This research was funded by the European Research Council Advanced Grant SOMACCA, Nr. 230604.

Contents

1 Introduction: Automatic animal training and testing

It is often the case in animal behavioral biology that a large amount of human resources, time, and data storage (such as video recordings) are required in animal observation and training. Some representative examples of these cases are:

- Observation of certain species continuously or monitoring for specific events, which occur irregularly, when behavior of certain species during any time period or specific time period, such as nocturnal behaviors, are investigated.
- Certain experiments require a prolonged training period, sometimes over a year. This type of experiment requires reliable responses, which may not correspond to usual behavior patterns, from animals in tasks. Therefore, training may require a long period of time until the subject is ready to be tested. Additionally, long periods of human supervised training can introduce unintended cues and biases for animals.

In the first case, an autonomous system for observing animals can save human resources and reduce the amount of data storage. The reduced amount of data can also conserve other types of human resources such as investigation and maintenance of large-scale data. There have been attempts to build autonomous observing or surveillance systems in the fields of biology, such as Kritzler et al. [1]'s work (Figure 1), and security systems, such as Belloto et al. [2], Vallejo et al. [3], for instance. There are also commercial products for surveillance systems with various degrees of automation, or incorporating artificial intelligence. However, the intelligence of each system is case-specific and it is difficult to apply these specific systems to novel situations without considerable adjustments.

In the second case, an autonomous system for prolonged, intensive training can also save human resources and eliminate potential cues and biases caused by humans. Training with an autonomous system is an extension of traditional operant conditioning chambers and many modern and elaborated versions have

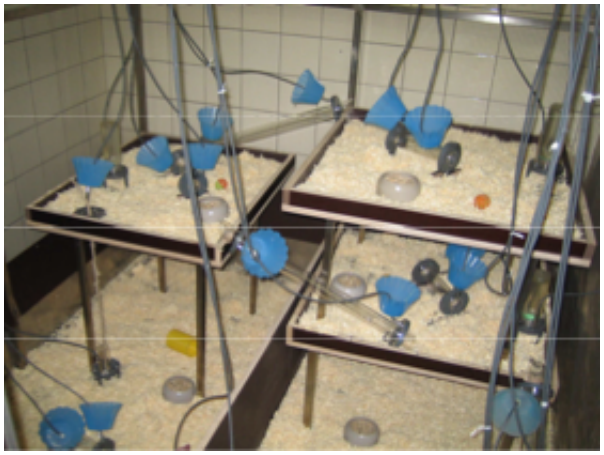

Figure 1: Photo of SNE device for mice, Kritzler et al. [1]

been developed and used, such as in Markham et al. [4], Takemoto et al. [5], Kangas et al. [6], Steurer et al. [7], and Fagot & Bonte [9]. However, many of the previous devices use commercial software. Also, they do not possess the observational features developed in the current project.

It would be useful to have an open-source, relatively low-budget, and modularized system which could be customized for the observation, training and the experimentation on animal subjects of various species. CATOS, the system built in the present study, fulfills these necessities.

The difference between the previous systems and CATOS (Computer Aided Training/Observing System) in the present work is that the animals do not have to be captured or transported to a separated space at a specific time in order to be trained. The disadvantages of separating animals (e.g., primates) are well-known, and include stress on animals separated from their group or moved from their usual confines, the risky catching procedure for both animal and human (cf. Fagot & Bonte [9]). Similar arguments apply to most animal species, especially when they are social.

The automatic learning device for monkeys (ALDM; Figure 2) described in Fagot & Bonte [9] is very similar to the trainer aspect of CATOS described in the present work, but CATOS is different in following features. First of all, it aimed

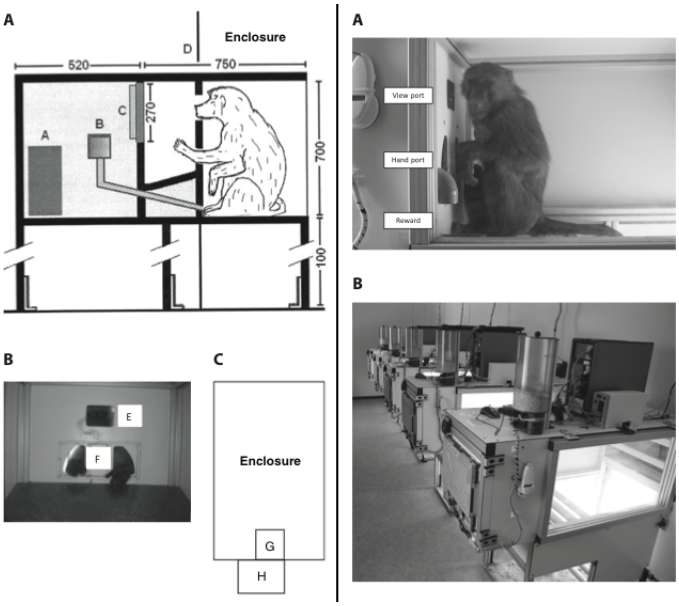

Figure 2: Schematic drawing of ALDM(left-side) and photos(right-side) of devices, Fagot & Paleressompoulle [8]/Fagot & Bonte [9]

to be open-source based and more modular so that it can be more easily adjusted and adopted to different species and experiments. Another feature is that CATOS is equipped with various observational features, including visual and auditory recording and recognition through video camera and microphone, which make the system able to interact with the subjects, such as reacting immediately to a subject with a motion detection from a camera or a sound recognition from a microphone.

CATOS should offer the following advantages.

- The system should be flexible in terms of its adjustability and the extendibility to various projectsand species. The software should be open-source, and both software and hardware components should be modularized as much as possible, so that they can easily be reassembled for various projects.

- The system should have various observational features applicable to a broad range of animal species and observational purposes.

- The system should perform continuous monitoring, and it should record video and/or sound only when a set of particular conditions is fulfilled. This would reduce the amount of data produced during the procedure.
- The system should have actuators to react in certain situations, which allows it to act as a trainer/experimenter. The human trainer/experimenter designs the procedure by adjusting parameters and modules, but the actual performance should be done by the system. In this way, the system could help reducing the amount of time required for training, and eliminating cues/biases which might be induced by the human interferences.
- With this system, the animal should not have to be transported to a certain space, or separated from its group, for training. The animals should be able to choose when to start a trial on their own.

Two CATOS prototypes have been built during this study. The first build of CATOS has 3 pushbuttons as a main input device for cats and the second build has a touch-screen as a main input device. The first build was an initial attempt to build and test such a system. The second build is the final product of the study. The basic structures of these two builds are more or less the same. The differences are that the second version has improved functions and it uses the touch-screen instead of pushbuttons.

The first build of CATOS was tested with domestic cats (*Felis catus*) to train them to press three different buttons differently depending on the auditory stimuli (three different human speech sounds). The final goal of this training is to investigate human speech perception in cats. There is no doubt in that many animal species can recognize some words in human speech. The examples of speech perception in dogs and chimpanzees can be found in the work of Kaminski et al. [10] and Heimbauer et al. [11] respectively. In some cases, animals can even properly produce words with specific purposes. An example of speech perception and production in a parrot can be found in the work of Pepperberg [12]. Despite these findings, there is ongoing debate about whether the same perceptual mechanisms are used in speech recognition by humans and animals (Fitch [13]). To

Software	(Tested) Version	Website
Operating system (Mac OSX)	10.7	apple.com
Python language	2.7	python.org
OpenCV library	2.3 & 2.4	opencv.org
ffmpg	1.1	ffmpeg.org
HTK	3.4	htk.eng.cam.ac.uk
pyaudio	0.2	people.csail.mit.edu/hubert/pyaudio
numpy	1.6	scipy.org
scipy	0.11	scipy.org
wxPython	2.9	wxpython.org
pyserial	2.6	pyserial.sourceforge.net
arduino	1.0	arduino.cc
matplotlib	1.2	matplotlib.org

Table 1: Summary of CATOS system dependencies

investigate this issue, animals have to be trained to show different and reliable responses to different human speech sounds. Then, we can test which features of human speech are necessary for different animal species to understand it. Thus, the final aim of the training in this study would be to obtain cats showing different responses to different human speech sounds with statistical significance (over 75%). Before reaching this final goal, several smaller steps and goals are required. These steps and goals will be described in later sections.

2 Base Technologies

At the heart of the CATOS system is a commercial computer (Apple miniMac, OS 10.7). The main computer language used is Python (Version 2.7.3, retrieved in November 2012, from http://www.python.org/), and several external packages. See the table 1.

The current sensory information processing of CATOS is built based on the computer vision algorithms (using OpenCV library; version 2.3.1a, retrieved on Feb. 12th, 2012, from http://www.opencv.org/), the Hidden Markov Model (using

HTK, Hidden-markov-model Tool Kit; Version 3.4.1, retrieved on Nov. 21st, 2012, from
http://htk.eng.cam.ac.uk/), and several sensors via a microcontroller. In the second build, a touch-screen with the Surface Acoustic Wave touch-technology was integrated.

2.1 Computer vision (OpenCV)

The current visual perception feature of CATOS is based on motion detection. This was accomplished using common web-cams and the open-source library called OpenCV, which is a library for real time computer vision and has over 500 functions [14]. The use of OpenCV means that many additional functions can be easily added depending on what the training and/or observing project requires.

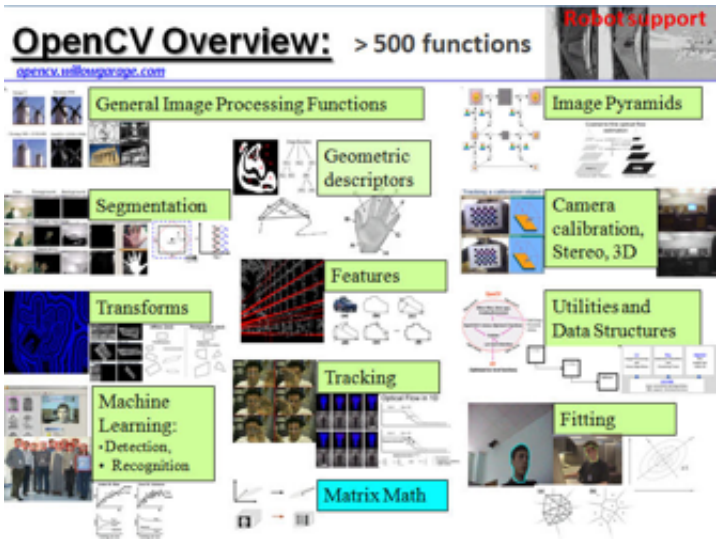

Figure 3: Categories of OpenCV functions;
from http://opencv.willowgarage.com/wiki/

2.2 Hidden Markov model toolkit (HTK)

The current auditory perception feature of CATOS is based on Hidden Markov Models. This was accomplished using a recording through a microphone and HTK, which is a toolkit with a set of library modules and tools for building Hidden Markov models. It is primarily designed for building HMM-based speech recognizers. (Young et al. [17]). The auditory perception of CATOS was not a required feature for the cat training program, which was performed with the first build of the system. But it was nevertheless implemented and tested as proof of concept.

2.3 Physical computing using a microcontroller

The interface between the computer and several sensors/actuators was implemented using a microcontroller called Arduino (see http://www.arduino.cc/). Arduino is a low-cost, extendable system intended for hobbyists, which has a wide variety of extension systems (”shields”) for different purposes.

3 Description of CATOS (Computer Aided Training/Observing System)

The overall system is composed of a combination of software and hardware components. (Figure 4).

The software components are mainly composed of the Python script named as ’AA.<version>.py’ and the program for the microcontroller. The ’AA’ runs all of the necessary processes and communicates with the microcontroller program. The microcontroller program operates sensors and actuators as it communicates with the ’AA’ program.

The hardware components are composed of various devices, some of which are directly connected to the computer via USB cables. Some other devices only have GPIO (General Purpose Input Output) pins; therefore they are connected

to the microcontroller. The microcontroller itself is connected to the computer via a USB cable. The hardware devices, which are directly connected via USB cables, can be accessed using various software modules, which are imported into the 'AA' program. The access to other devices only using GPIO pins is performed in the microcontroller and the 'AA' program simply communicates with the microcontroller program via a serial connection for sending commands to actuators and receiving values from sensors.

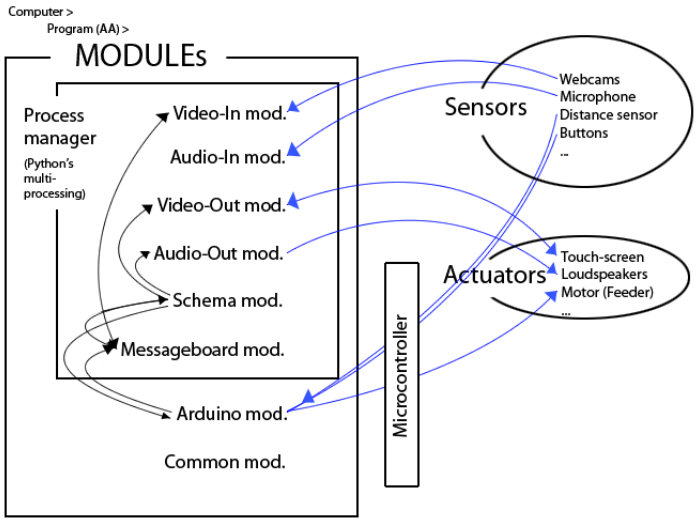

Figure 4: Schematic diagram of the overall system; Black lines denote the message exchange between the modules. Blue lines denote the control connection between the software and the hardware component

3.1 Device design

The design of the first build is depicted in Figure 5 (except 2 webcams attached on the walls). The design of the second build is depicted in Figure 6. Beside small changes in the design of the feeder, the big differences between the first and the second build are that the second build uses the touch-screen instead of pushbuttons, and many electronics, which should not be exposed to the subject,

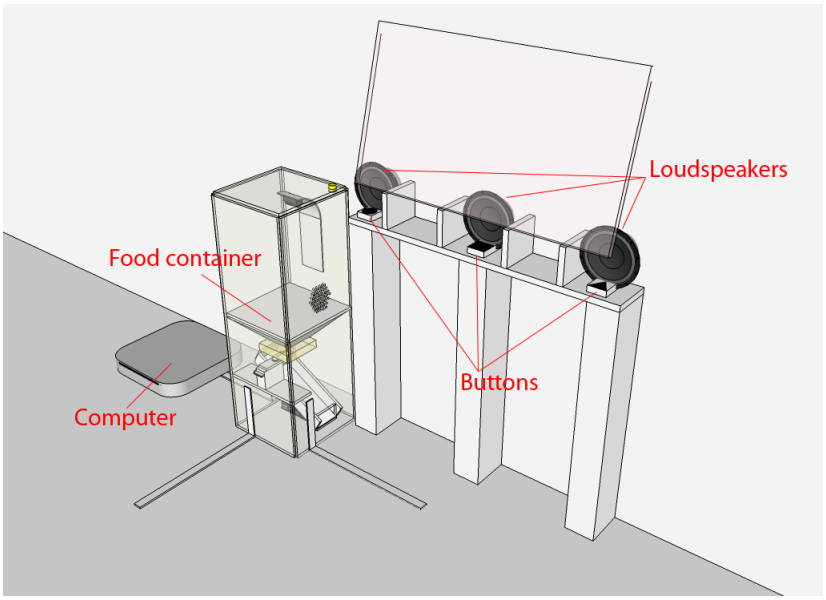

Figure 5: The design of the device of the first build

are embedded in a protective case. Three side panels of this protective case can be attached and detached easily via using magnets.

3.2 Software in the overall structure

The software for this system is called AA (Agent for Animals) as depicted in Figure 4. Once it starts, it constantly runs 7 processes in parallel until the user terminates the program. The number of processes can be changed as some of them can be turned on or off. These processes include a video-in process for each camera, a video-out process, an audio-in process, an audio-out process, a schema process, and a message-board process. Even though some of these processes have quite simple tasks, they were separated in order to prevent them from interfering with each other and/or becoming the bottleneck. The system has to process the visual, auditory, and other sensory and motor information simultaneously to recognize the change of the environment and respond to it properly.

The output data such as captured video input images, recorded WAV files,

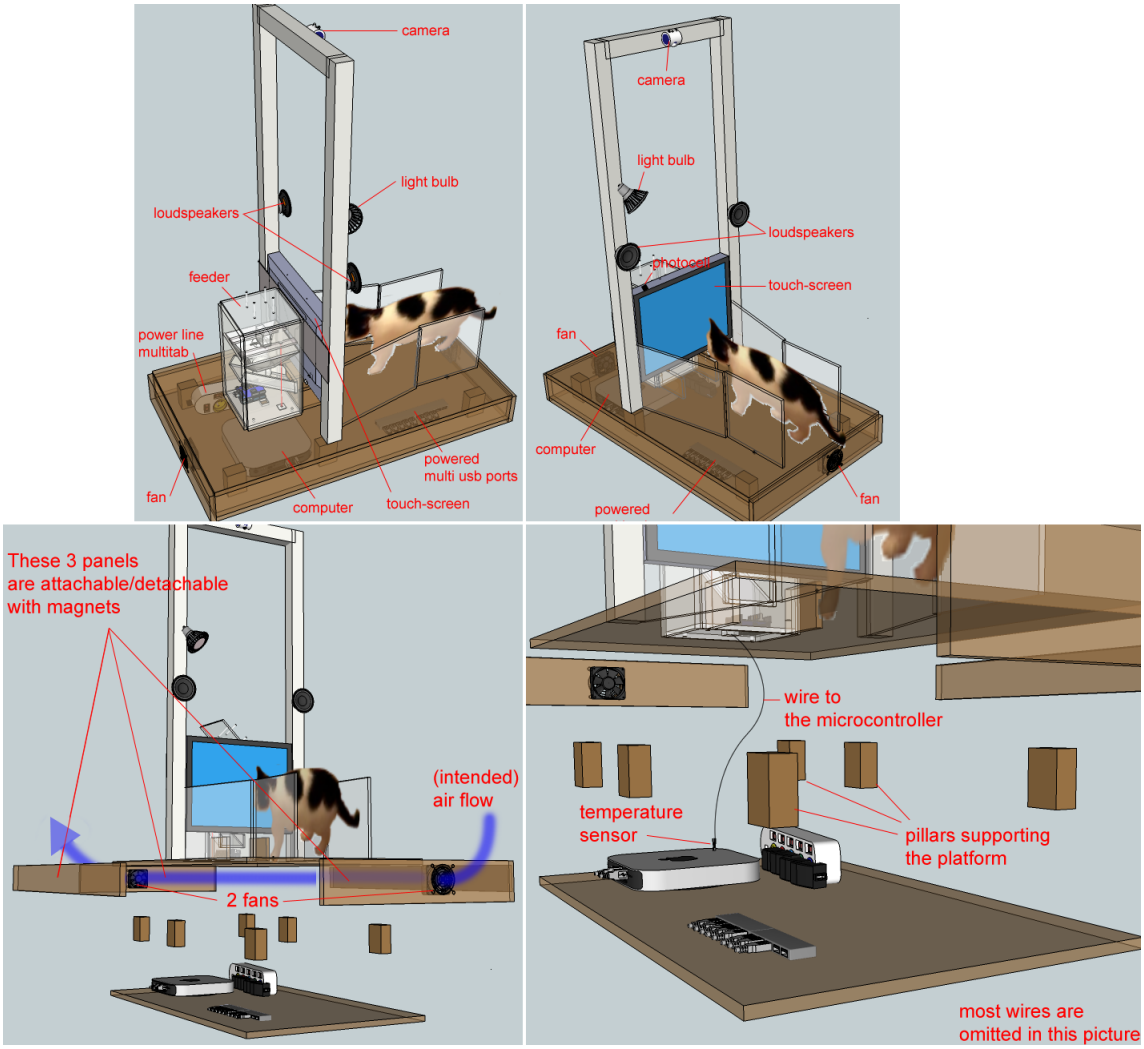

Figure 6: The design of the device of the second build

movement-records, CSV files for trial results, and the log file are temporarily stored in the 'output' folder. After the daily session is finished, all of these output files go through an archiving process which can include, but is not restricted to, generating movies, generating images with the movement analysis, labeling sound files, and moving different types of files into the categorized subfolders of an archiving folder named with a timestamp.

3.3 Message-board process

The message-board process facilitates the flow of messages among modules, and between the program and the microcontroller.

3.4 Schema process

The schema process mainly takes care of running trials, but it also handles other types of scheduled tasks such as sending emails periodically and logging system messages periodically.

A typical trial procedure, as used in this study and described in detail in later sections, proceeds as follow; one of the several feeding sounds is played with a certain time interval. The feeding sound for the trial is randomly determined and the time interval between trials is also randomly determined on each trial within a certain range. The process asks other modules for messages to check the cat's movements and the button-presses. It also sends messages to actuators to react appropriately to the cat's responses. This is a typical example of using this process, but the schema process is very flexible, and can be coded and parameterized differently for different training procedures and experiments (Figure 7).

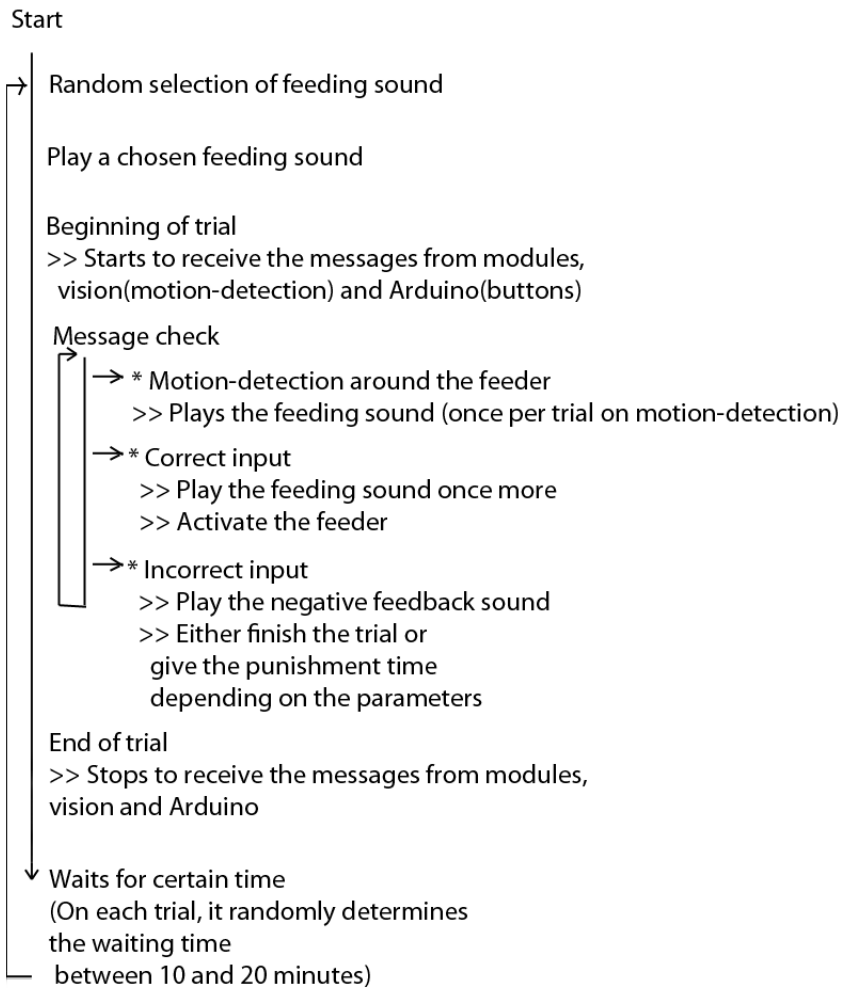

Figure 7: Flow of the current program's schema process

3.5 Video-In process

The video-in process is used for recognizing a movement, extracting moving objects (background subtraction) and recording it. The sequence of images is obtained constantly from 2 web-cams, positioned about 3 meters above on the wall of the experimental room. Once any motion is detected, the program starts to store the images (JPEG format) to a temporary folder until no motion is detected for 3 seconds. It also stores the records of movements such as the center point of movements and center points of foreground blobs. The stored series of JPEG images in a folder, from a motion-detection to 3 seconds of no-motion period, are compressed into one MP4 movie file using FFmpeg (http://www.ffmpeg.org) when the daily session is finished.

The two main algorithms in this process are described below.

3.5.1 Motion detection

This function searches for any change in recently obtained series of images and returns the center point of overall change. The key procedures to obtain this motion-detection information are as following (Bradski & Kaehler [14]):

1. The average of the recent images was obtained by the cv.RunningAvg function. The description of the cv.RunningAvg is :

   - image - Input image, 1- or 3-channel, 8-bit or 32-bit floating point (each channel of multi-channel image is processed independently)
   - acc - Accumulator with the same number of channels as input image, 32-bit or 64-bit floating-point
   - alpha - Weight of input image
   - mask - Optional operation mask

   The accumulator, acc, is given by the following formula:

$$acc(x,y) = (1-\alpha) \cdot acc(x,y) + \alpha \cdot image(x,y),\ if\ mask(x,y) \neq 0$$

- x - x coordinate of the image
- y - y coordinate of the image
- α - alpha parameter, which regulates the update speed (how fast the accumulator forgets about previous frames)

2. The absolute difference between the current frame image and the average image is calculated using the cv.AbsDiff function. The description of the cv.AbsDiff is :

   - src1 - The first input array
   - src2 - The second input array; Must be the same size and same type as src1
   - dst - The destination array; it will be the same size and same type as src1

3. The outline of the differences are detected by the Canny edge detector of Canny [15], using the cv.Canny function. The description of the cv.Canny is :

   - img - single-channel 8-bit input image
   - edges - output edge map; it has the same size and type as img
   - lowThresh - first threshold for the hysteresis procedure
   - highThresh - second threshold for the hysteresis procedure
   - apertureSize - aperture size for the Sobel() operator

4. The coordinates of fragments of the outline are obtained by the algorithm of Suzuki & Abe [16] using the cv.FindContours function. The description of the cv.FindContours is :

   - img - 8-bit single-channel input image
   - storage - indicating memory in which to record the contours

- mode -
  - CV_RETR_EXTERNAL: Retrieves only the extreme outer contours
  - CV_RETR_LIST: Retrieves all the contours and puts them in the list
  - CV_RETR_CCOMP: Retrieves all the contours and organizes them into a two-level hierarchy, where the top-level boundaries are external boundaries of the components and the second- level boundaries are boundaries of the holes
  - CV_RETR_TREE: Retrieves all the contours and reconstructs the full hierarchy of nested contours
- method -
  - CV_CHAIN_CODE: Outputs contours in the Freeman chain code; all other methods output polygons (sequences of vertices)
  - CV_CHAIN_APPROX_NONE: Translates all the points from the chain code into points
  - CV_CHAIN_APPROX_SIMPLE: Compresses horizontal, vertical, and diagonal segments, leaving only their ending points
  - CV_CHAIN_APPROX_TC89_L1 or CV_CHAIN_APPROX_TC89_KCOS: Applies one of the flavors of the Teh-Chin chain approximation algorithm

The images showing the example of motion-detection can be found in Figure 8

The motion detection described here is important for the system since it does not only triggers a recording, but also triggers certain responses of the system to the animal subjects.

3.5.2 Background subtraction

The background subtraction is activated only when motion is detected. The background subtraction is useful for reducing the time needed to analyze the video

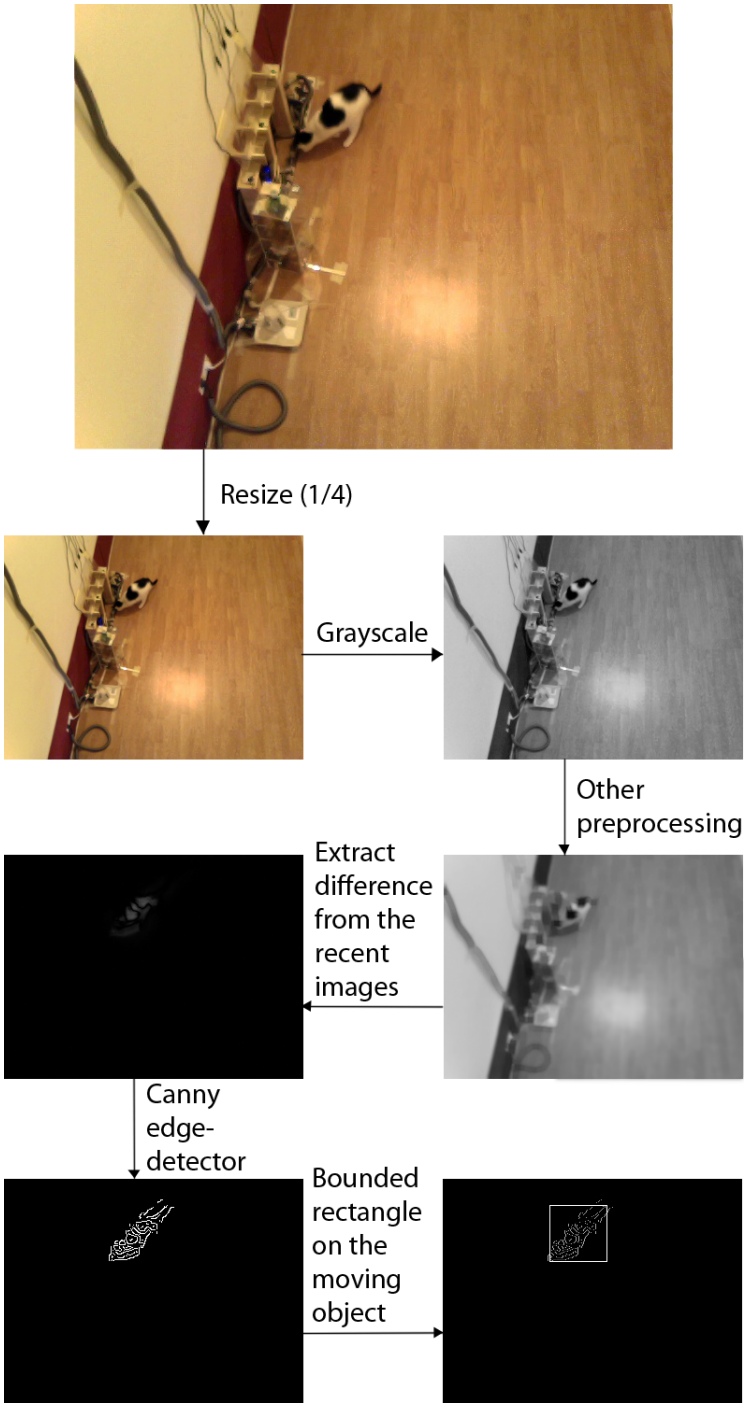

Figure 8: Example of motion-detection processing

data later. Obtaining a clear image of foreground blobs is not a critical step for the real-time process. Therefore, it was performed in a very simple way rather than using sophisticated algorithms. The program stores an image when a session starts and uses it as a background image. The program distinguishes the foreground blobs from the background image, mainly using the combination of the cv.AbsDiff and the cv.Threshold functions. The cv.AbsDiff function was already mentioned and the cv.Threshold functions generates a binary image depending on the threshold parameters. This function's description is :

- src - Source array(single-channel, 8-bit or 32-bit floating point)
- dst - Destination array; will have the same size and the same type as src
- threshold - Threshold value
- max_value - Maximum value to use with THRESH_BINARY and THRESH_BINARY_INV thresholding types
- threshold_type -
  - CV_THRESH_BINARY; dst_i = (src_i >T), otherwise 0
  - CV_THRESH_BINARY_INV; dst_i = (src_i >T), otherwise max_value
  - CV_THRESH_TRUNC; dst_i = (src_i >T), otherwise src
  - CV_THRESH_TOZERO_INV; dst_i = (src_i >T), otherwise src
  - CV_THRESH_TOZERO; dst_i = (src_i >T), otherwise 0

The background image can be updated if a foreground blob is persistent over a certain prolonged time. This helps to prevent recognizing certain changes in the background. An example case would be if an inanimate object was moved by the animal and it remains as a foreground blob continuously.

3.6 Video-Out process

The video-out process takes care of displaying certain images on the touch-screen and accepting touches from it for a trial. It uses the wxPython-2.9 graphical user interface library (retrieved from http://www.wxpython.org in March, 2013). This process was not implemented in the first build. In the second build, this process offers access to the touch-screen interface to the animal subject.

3.7 Audio-In process

The audio-in process is for recording any sound around the microphone. Similar to the video-in process, when the Root Mean Square amplitude goes above the threshold, it starts to record the sound as a WAV file until the amplitude drops under the threshold for 3 seconds. It also applies a high frequency pass filter before the measurement of RMS, to prevent the false recording of a door closing or other noises caused by humans in other spaces than the experimental space.

All the stored WAV files were automatically labeled with a model trained by HTK, Hidden-markov-model Tool Kit, program (Version 3.4.1, retrieved on Nov. 21st, 2012, from http://htk.eng.cam.ac.uk/), which is an open-source toolkit for building and manipulating HMM, Young et al [17]. A brief diagram of Figure 9 and Figure 10 depicts the general work-flow of HTK.

A Hidden Markov Model was trained using HTK to facilitate the human recognition of the WAV files by automatic pre-labeling of files. In the cat training/observation project reported in this study, the label was simply ”eating”(crunching sound made by a cat eating the dry food used as a reward) or ”etc”(all other sounds).

The HMM was not crucial for the training performed in this study, but it is a potentially important feature for future usages such as training/experimenting on any animal species whose vocal communication in a group is active or for training animals to vocalize. Namely, the HMM recognizer can act like the motion detection in the video-in process for responding toward animal subjects in real-time to their vocalizations.

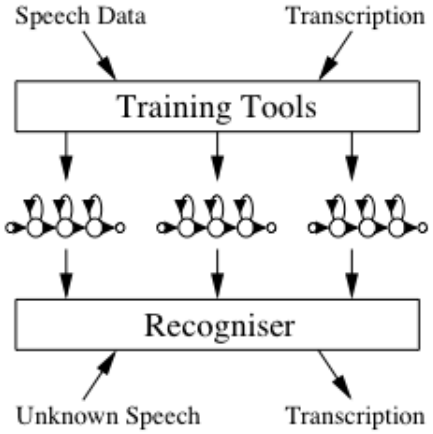

Figure 9: Simple diagram showing the general work-flow of HTK, Young et al [17]:p.2

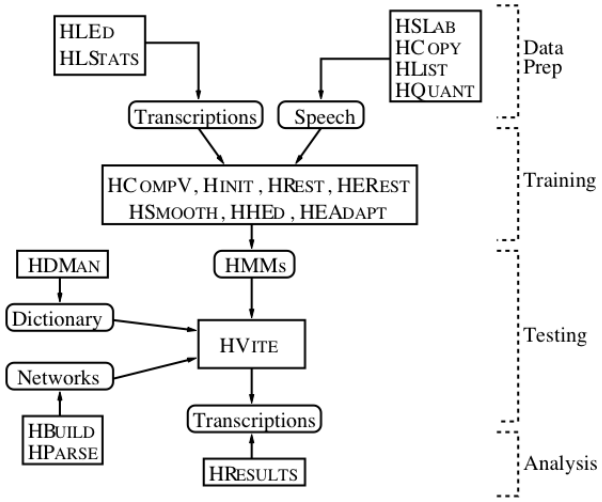

Figure 10: HTK processing stages, Young et al [17]:p.17

3.7.1 Preparing the sample WAV files

Several hundred WAV files (most of them were 3 to 5 seconds long), which were automatically recorded by the system as it was explained in 2.3, were used as input to train the HMM. Then, they were sliced into smaller WAV files containing segments only with ”eating” or ”etc” sound. This editing process was manually done using a program, Praat [18].

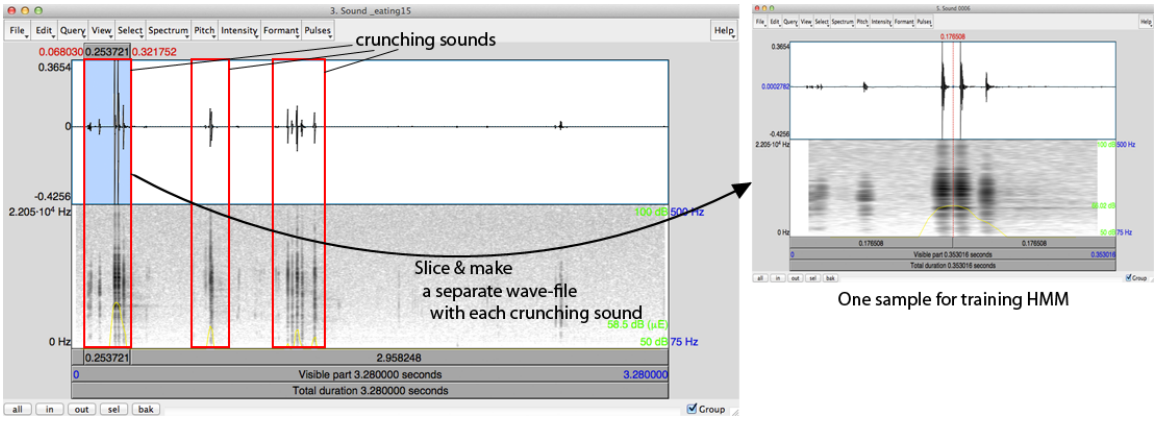

Figure 11: Making a sample WAV file

3.7.2 Training Hidden Markov Model

Several different Hidden Markov models were trained and tested using several tools in HTK such as ’HCopy’, ’HInit’, ’HRest’, ’HERest’, and ’HHEd’.

The common aspects for all the different trained models were as following:

- The ’targetkind’ of the configuration file was ’MFCC_E_D_A’, Mel-frequency cepstral coefficients.
- The grammar was ’(eating — etc)’ meaning the wave data is either ”eating” or ”etc”.
- The proto model files for both ”eating” and ”etc” were same.

The distinct aspects for the different trained models were as following:

- The first type of model is depicted in 1) of Figure 12. These models were trained with 'HRest' of HTK. In this first type, different values for parameters, 'NUMCHANS (Number of filter-bank channels to use in the analysis)' and 'NUMCEPS (Number of cepstral coefficients)' were tried, and also various numbers of Gaussian distributions using 'HHed' of HTK was tried.

- The second type of model is depicted in 2) of Figure 12. These models had one more state than the first type. Similarly to the first type, different parameterizations and different numbers of Gaussian distributions were tried.

- The third type of model is depicted in 3) of Figure 12. There was a tied state between "eating" and "etc" models. These models were trained with 'HERest' of HTK. Also, different parameterizations and different numbers of Gaussian distributions were tried. One of these models became the final models of the system.

Refer to Figure 12 for the different structures of the above 3 models.

3.7.3 Testing HMM

The first and the second type of models were not very successful. 5 "eating" sounds and 5 "etc" sounds were randomly picked up out of the WAV files recorded by the system. These 10 files were tested using the trained model with 'HVite' and 'HResults' of HTK. Most of them, if not all, were labeled as "etc". For the third model, various parameters and the number of Gaussian distributions were tried. Several preliminary tests on the third model with various parameters were performed in two sample groups. One group had all the sample WAV files, which were used for training the HMM models. Another group had 10 to 20 WAV files. Half of them were "eating" and another half of them were "etc". They were randomly chosen from all the WAV files recorded by CATOS during sessions. These models were trained twice more later due to the application of the high-frequency

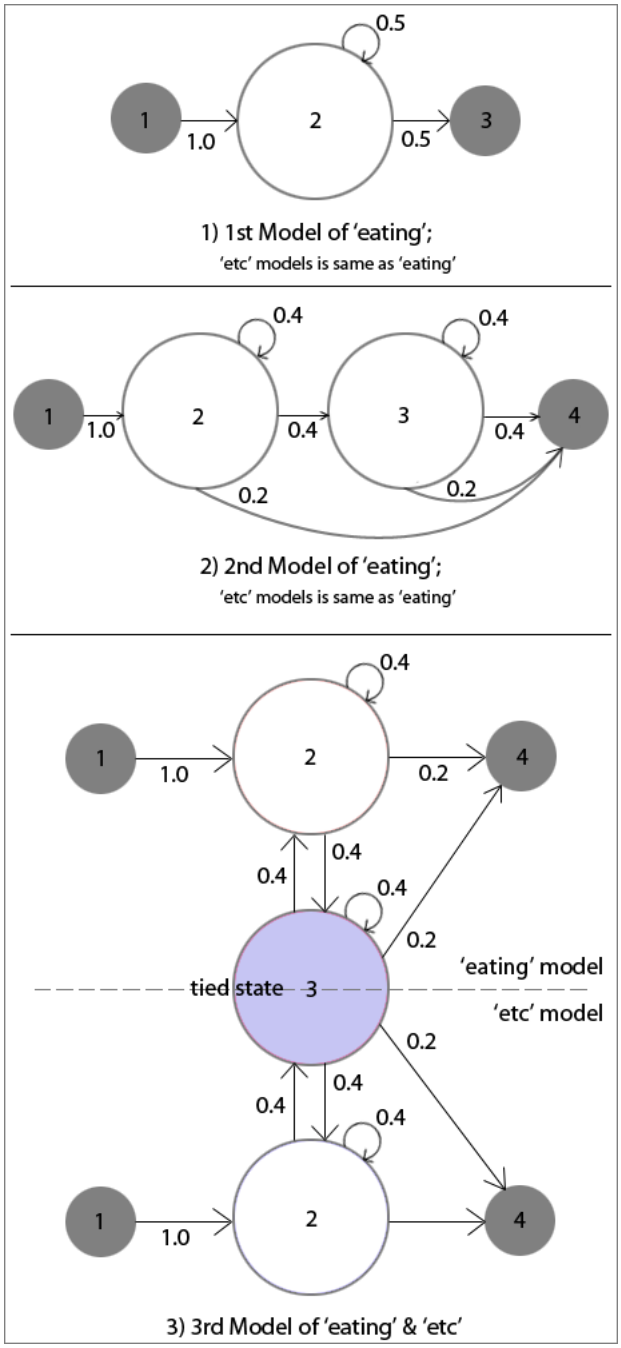

Figure 12: HMMs' states transition probabilities. * The numbers besides the arrows are the transition probabilities. ** The first and the last state(grey color states) are not emitting states. *** The state #3 of the third model is a tied state between "eating" and "etc" accounting for short silent moments between sounds.

	Parameters (NUMCHANS, NUMCEPS, Number of Gaussian dist.)	Correctness rate			
		Raw WAV files		Sliced WAV files used for training HMM	
		eating	etc	eating	etc
First training	6,6,2	100.0	100.0	99.6	84.6
	6,6,4	100.0	100.0	99.8	90.9
Second training: Training after adding the high frequency pass filter	6, 12, 1	0.0	100.0	-	-
	6,12,2	0.0	100.0	-	-
	6,12,4	0.0	100.0	-	-
	6,12,8	90.0	100.0	100.0	99.1
The second training; Training after adding more samples	6, 12, 10	90.5	100.0	-	-
Third training: Training after adding more samples, which have both sounds	26, 24, 8	82.1	72.4	100.0	92.7

Table 2: Test results of the third HMM models; The first training: Number of raw WAV files were only 5 for each label, The second training: Number of raw WAV files were 10 for each label, The third training: Number of raw WAV files were 57. Forty of them were from the previous training, and an additional eight ”eating” and nine ”etc” sound files contained both ”eating” and ”etc” sounds in one file

pass filter and adding more samples, which had both sounds, ”eating” and ”etc”, in the same WAV file. The results after testing with the different versions of the third model are depicted in the Table 2.

The last trained model (the colored row in Table 2) from the second training was applied for the system because the third training did not improve the model significantly. When the AA program finished each daily session, it labeled each WAV file using this model. After running this procedure for a few weeks, all the WAV files recorded in three randomly chosen days were manually evaluated. The total number of evaluated WAV files were 287. The correctness-rate was about

95%.

3.8 Audio-Out process

The audio-out process plays auditory stimuli (WAV files) through the loudspeaker(s).

3.9 Feeder

The two feeders (Figure 13) used in this study is a device mainly comprising the Arduino microcontroller; (refer http://www.arduino.cc/), a motor-shield for the microcontroller, a servomotor, and a frame encasing the whole feeder. Both Feeder variants work in a similar way, by rotating the servomotor by a certain number of degrees, although the second feeder shows better performance in terms of consistent amount of food released, due to the usage of an Archimedes' screw.

Initially, an estimate of the amount of food left in the food container was obtained using an IR distance sensor, but this feature was discarded in the second build since the distance information from the IR sensor was not accurate enough for this application.

The second feeder confirms the emission of a food reward via the piezoelectric sensor, which is positioned right below the Archimedes' screw.

3.10 Physical computing using a microcontroller

Communication between the Arduino chip and the main computer was accomplished by using the Arduino module of the 'AA' program.

The circuit for the first build is depicted in Figure 14.

- The three mechanical buttons provided the interface used to obtain a response from the cats.
- The IR distance sensor was used for measuring the amount of food left.

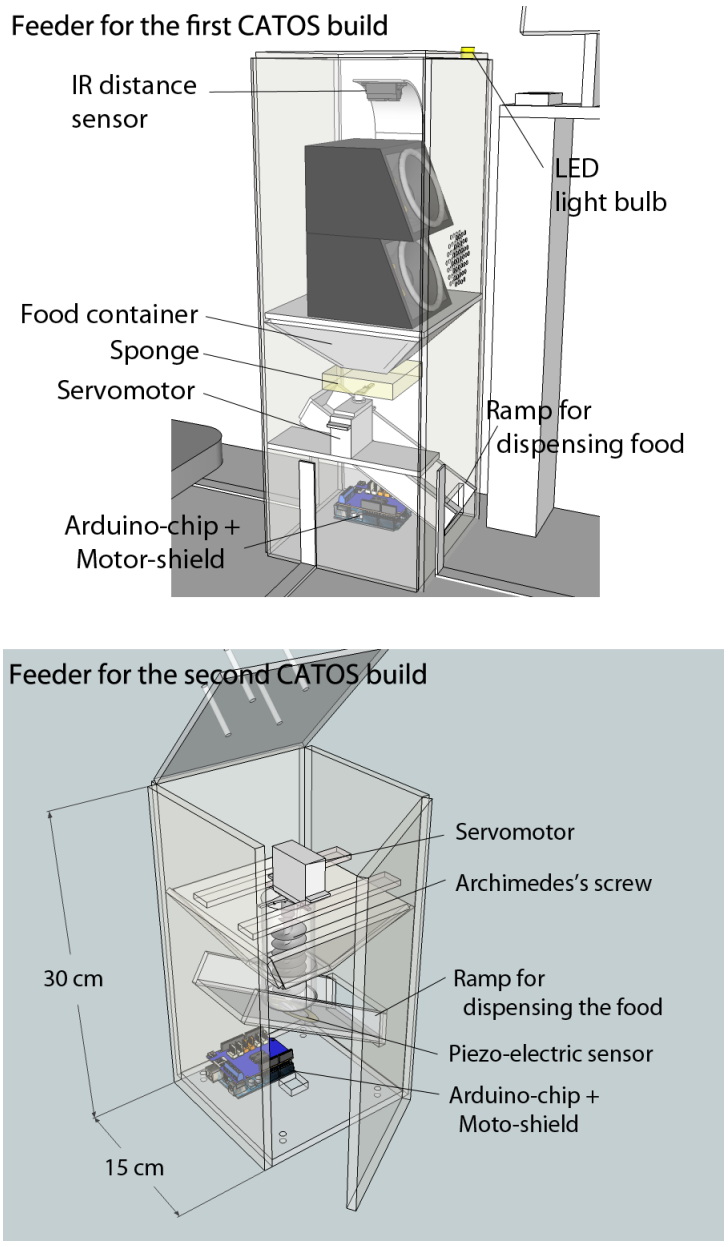

Figure 13: Feeders for the first and the second CATOS builds

- The servomotor dispense the food via a brief opening of the food container at the bottom.

The circuit for the second build is depicted in Figure 15.

- The temperature sensor measures the temperature inside of the protective wooden platform.
- The photocell sensor measures the ambient light level.
- The light bulb can be turned on when the photocell sensor indicates the ambient light level is below a user-defined threshold.
- Two fans are turned on when the temperature sensor indicates the temperature is too high in the platform.
- The piezoelectric sensor is read while the servomotor is actuating, in order to confirm the occurrence of the food reward. This sensor reading is required because occasionally the food dispensing fails due to the combination of the short motor activation time ($<$0.5 seconds) and the shape of the dry food pieces (which can fit into other pieces easily and then fail to emrege).
- The servomotor is responsible for the food dispense by turning the Archimedes' screw back and forth.

3.11 Utility program; AA_Data viewer

Besides combining all the above modules and implementing some common functions, one more Python program was implemented to facilitate the process of analyzing the recorded data. The program is called "AA_DataViewer". It loads the log file, the result CSV (comma separated values) file containing the results of the trial, the movement-record CSV files, the MP4 movie files, and the WAV files from one folder containing all data collected for one session (day). In the Figure 16,

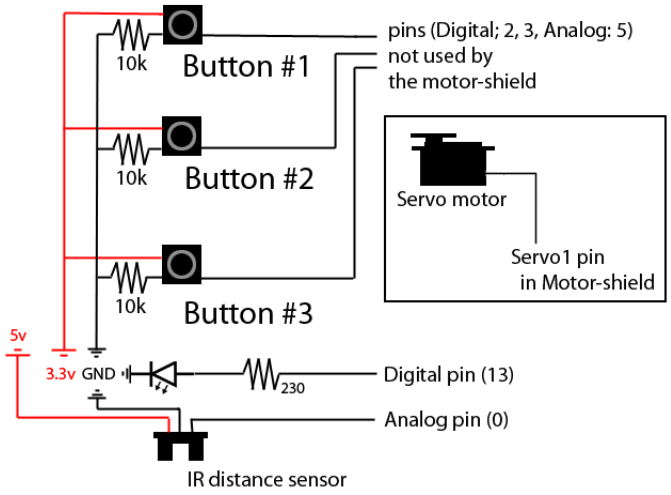

Figure 14: Circuit diagram of the microcontroller in the first build

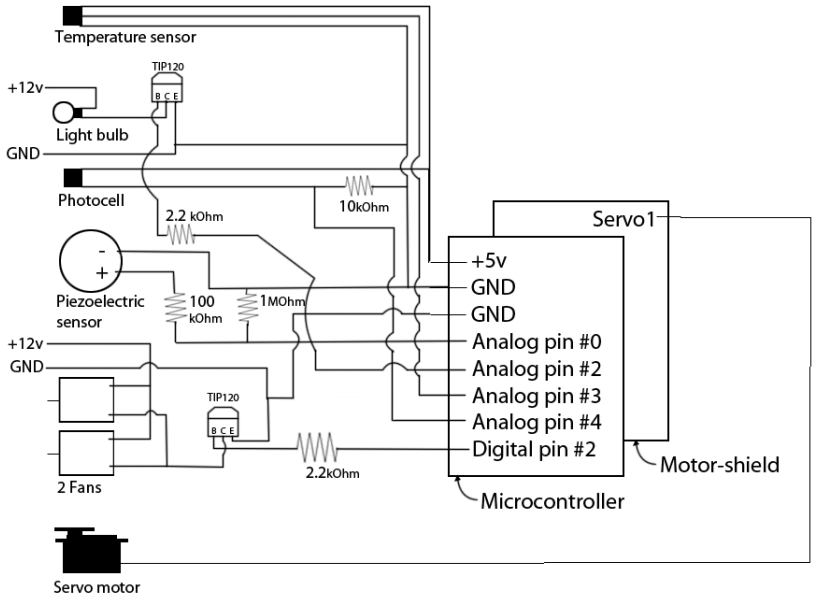

Figure 15: Circuit diagram of the microcontroller in the second build

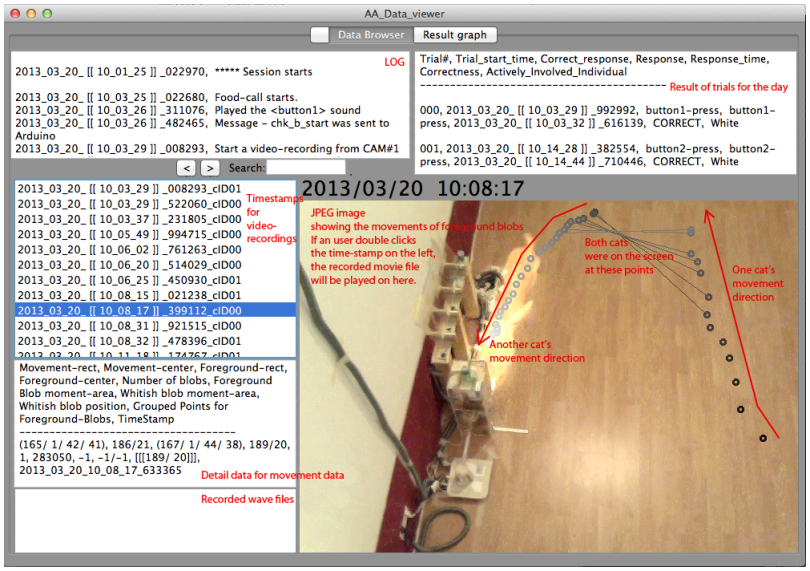

Figure 16: Screenshot of AA_DataViewer; Browsing various data from a session

there is a JPEG image showing the movements of the blobs. The circles represent the positions of the blobs and their color represents the time-flow, with the black corresponding to the beginning of the movie, and the white to the end of the movie. A line connecting multiple circles means that those blobs occurred at the same time.

Another feature of this program is its ability to generate a graph with selected sessions as in Figure 17. In the 'archive' folder, there are sub-folders, each of which contains all the data for a session. When the 'select sessions' button is clicked, a pop-up window appears for selecting multiple folders. The result data from these selected sub-folders of 'archive' folder is drawn as a graph using an external Python library called Matplotlib. By visualizing the data for certain period, it helps the trainer or experimenter quickly assess the current status of the training procedure.

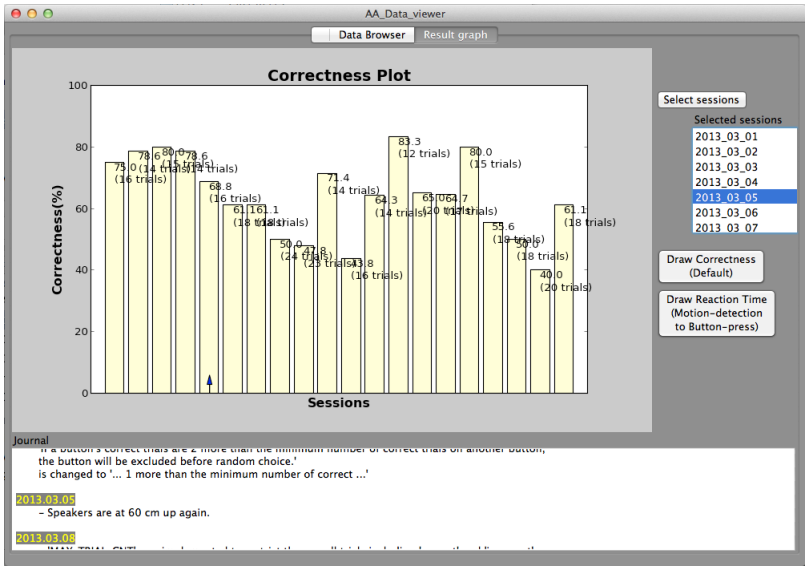

Figure 17: Screenshot of AA_DataViewer; Graphical representation of performances across sessions

4 Testing the system; Procedures for training the cat

The two pet cats in this study were originally obtained from an animal shelter in June of 2012, and had not been exposed to any training since June of 2012 until the beginning of the training for this study in October, 2012. Both were approximately 2.5 years old. One was a neutered male and another was an intact female.

This study was performed in the cats' residence. The spaces were as depicted in Figure 18. The room #1 (blue colored area; 5 meters long and 2 meters wide room) was used for this study, but they could access any of the spaces depicted in Figure 18.

The setup in Room #1 was as Figure 19.

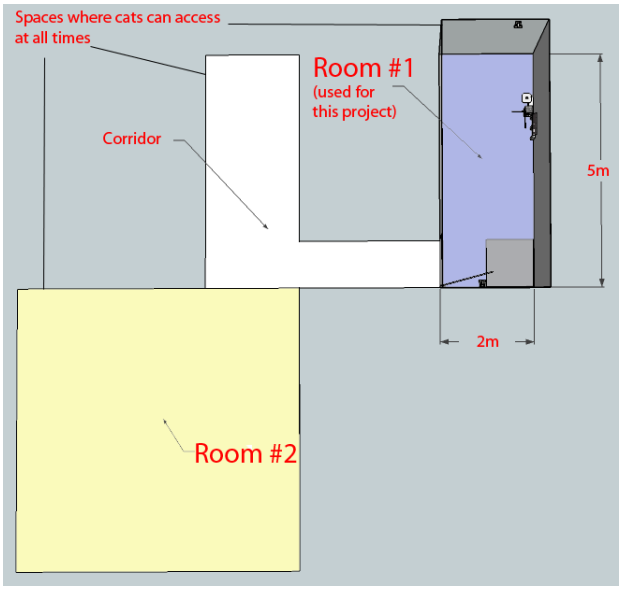

Figure 18: Spaces in which cats can access. Room #1 was the experimental space.

4.1 Session and trials

Each daily session lasted between 8-12 hours depending on the training state. A sound from the possible trial sounds was randomly chosen to be played with a random inter-trial interval. After this playback, if there was no reaction such as a motion-detection or button-press within the duration of the trial, which could be between 20 and 60 seconds depending on the training state, then it was not recorded as a trial in the result CSV file. If there was a reaction within the duration of the trial, then the trial officially started, which means logging and writing a trial-entry in the result CSV file.

At the beginning, there was no limit on the number of trials per day. In this case, the cats approached the feeder on the playback of the sound for the trial only when they wanted to do so, and they ignored the sound otherwise. Later, when the difficulty level was increased, a limit of the number of trials per day was implemented in order to restrict the cats from randomly pressing buttons repeatedly to obtain food rewards.

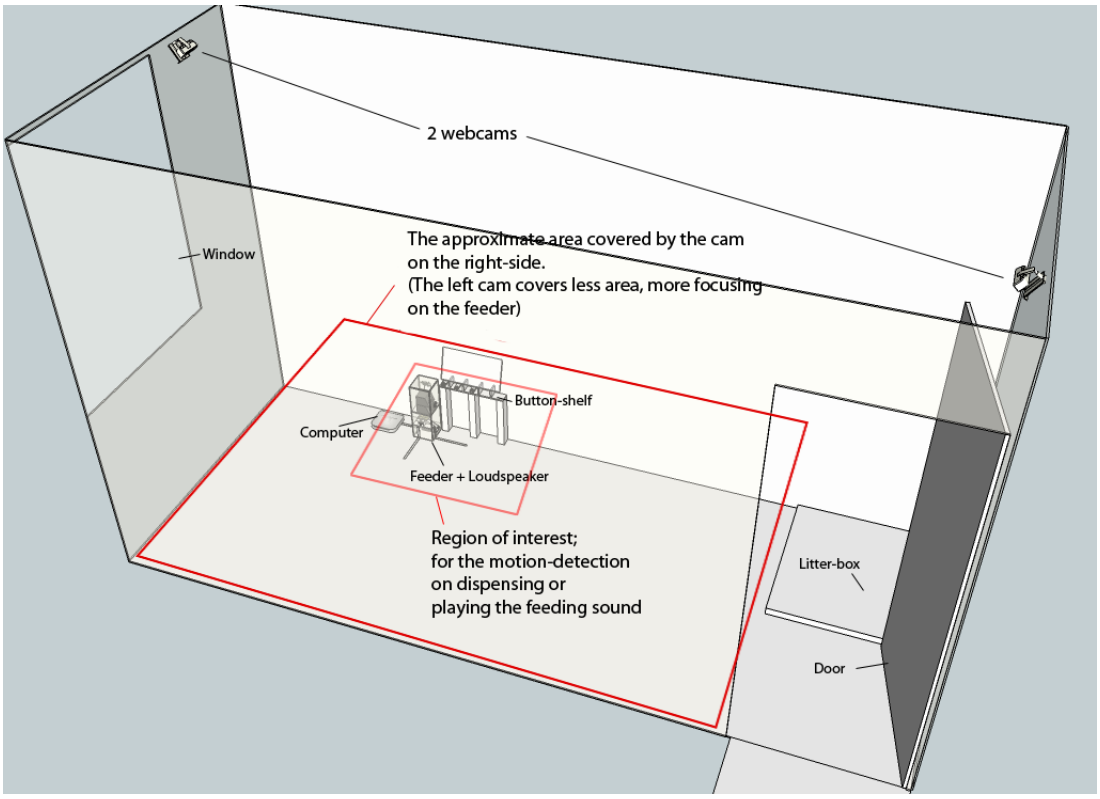

Figure 19: Setup of the experiment room

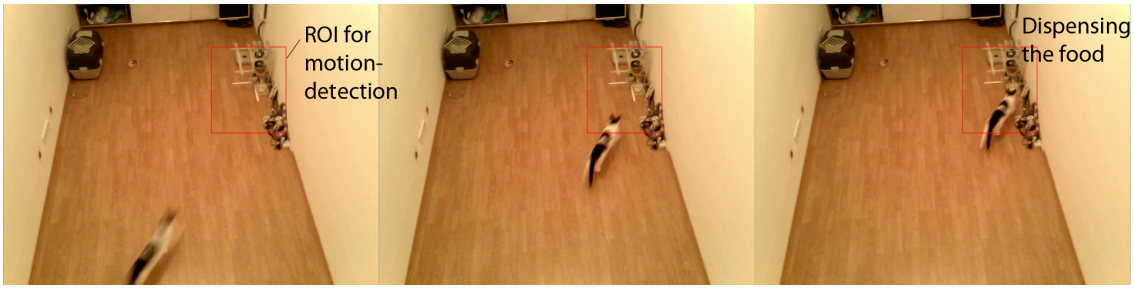

Figure 20: Female cat approaching the feeder right after CATOS played the feeding sound

4.2 Training phase #1 Approaching the feeder in response to sound playback

The goal of this phase was to train the cats to learn to associate a sound playback through the loudspeaker with a potential food reward. The sound stimulus was the experimenter's recorded speech calling the cats' names. If any movement was detected around the feeder, a small amount of food was dispensed. Figure 20. If there was no movement around the feeder for 20 seconds after playing the feeding sound, the trial ended without dispensing any food.

4.3 Training phase #2 Pressing a button in response to sound playback

The goal of this phase was to train the cats to press a button after the sound stimulus playback in order to obtain the food reward. The trial length, or in other words, the time period during which a cat can press a button to obtain the food reward, was 30 seconds long.

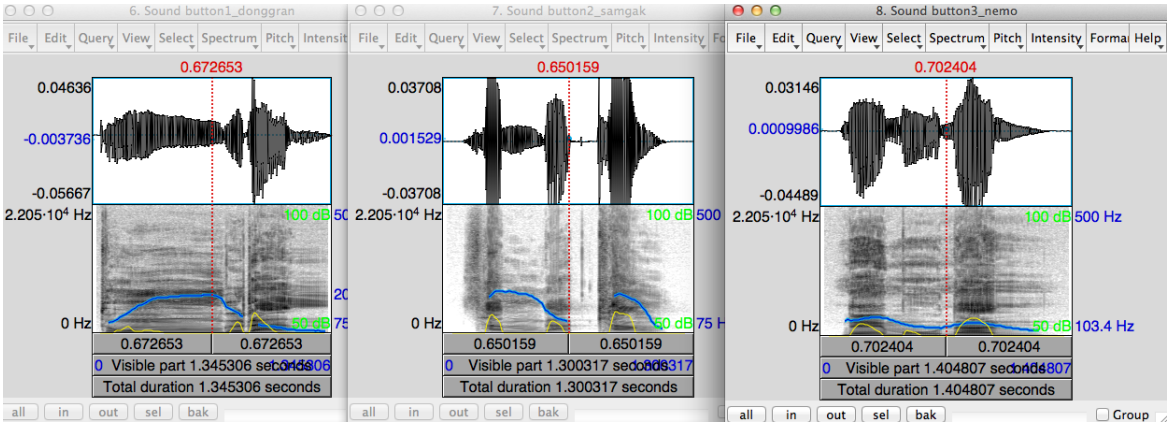

Figure 21: spectrogram of each feeding sound for 3 buttons

-

4.4 Training phase #3 Pressing a specific button corresponding to a specific sound stimulus

The goal of this phase was to train the cats to associate a specific feeding sound with a specific button when there are multiple buttons. The trial length was 60 seconds long. Through this phase, new buttons were added one by one. In total, three buttons were implemented, and the 3 different feeding sounds, each of which corresponding to a button. The sound stimuli are the experimenter's recorded speech, 'Dong-g-ran', 'Sam-gak-hyeong', and 'Ne-mo-nan' respectively. The spectrograms of these sounds are shown in Figure 21. They have similar duration and mean intensity. The durations are 1.35, 1.30, 1.40 seconds respectively, and the mean intensities are all 64.5 dB. However, these sounds are quite different in terms of the prosody. The first syllables of each of them use different vowels, 'o', 'a', and 'e', to make the differences between them more noticeable.

This phase was the longest of all three training phases. A number of parameters and assisstance features were implemented or deactivated. Some of them were as follows: various trial lengths, negative feedback sound, sound playback once more on the correct button response, repetition of the same trial on incorrect response, punishment time (no-reaction of the device on any button-press

with the LED light on), and so on. None of these variations yielded significant improvement. Then, three speakers were implemented right behind each of the three buttons to help the cats to differentiate the sounds, at least by localization. These three speakers were at the height of the buttons for the initial implementation, but were later elevated to 30cm, 45 cm, 60cm, and 120cm above the buttons to prevent the cats from relying only on localization cues to choose the correct button (Figure 22). After they were elevated 120 cm above the buttons, the three speakers were also gathered together around the middle speaker, meaning that they were positioned approximately in the same location.

5 Results

5.1 Results from system design and manufacture

5.1.1 Reduced data storage

The two web-cams observed the experimental area (room #1 of Figure 18) for 8 to 12 hours per day for about 5 months (from the middle of October 2012 to the middle of March 2013). The movement records, MP4 movie files, JPEG image files, and WAV sound files generated during this period took 37.35 Giga bytes of storage.

To obtain a rough idea of the degree of reduction in data storage that was achieved using the system, the number of recorded frames in the video recording was assessed. Data for 15 days were taken to calculate it. The total observation period was 406138 seconds, corresponding to 112.8 hours. The number of frames recorded was 206024 and the average FPS(Frame Per Second) was 7.5, therefore, approximately, the video recordings were stored for 27470 seconds (=7.6 hours), which is about 6.7% of entire observation period.

These specific numbers are not very meaningful since they can fluctuate with the increase or decrease of the subject's movements, but the point is that the most of the meaningless recordings were successfully filtered out by CATOS.

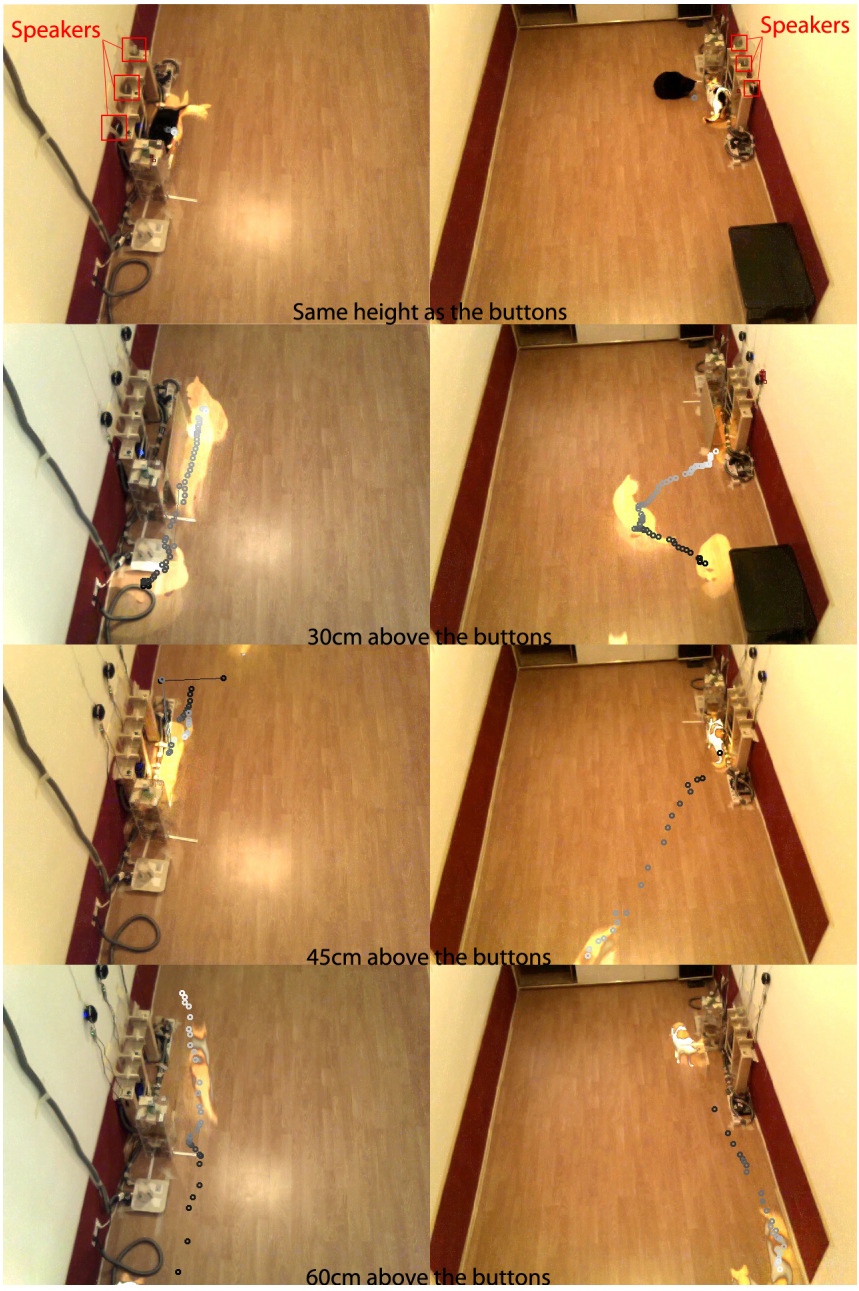

Figure 22: Three speakers at different heights; views from two webcams

Trial#	Trial_start_time	Correct_response	Response	Response_time	Correctness	Actively _involved _individual
000	2013_03_17_10_54_13_965895	button2-press	button3-press	2013_03_17_10_54_15_388055	INCORRECT	White
001	2013_03_17_11_12_10_886611	button3-press	button3-press	2013_03_17_11_12_15_867220	CORRECT	White
002	2013_03_17_11_29_12_784309	button2-press	button2-press	2013_03_17_11_29_16_663150	CORRECT	White
003	2013_03_17_11_44_10_316059	button1-press	button3-press	2013_03_17_11_44_25_102358	INCORRECT	White
004	2013_03_17_12_04_11_212958	button1-press	button1-press	2013_03_17_12_04_15_334470	CORRECT	White
005	2013_03_17_12_15_11_635546	button2-press	button2-press	2013_03_17_12_15_20_902902	CORRECT	White
006	2013_03_17_12_35_12_568855	button3-press	button3-press	2013_03_17_12_35_18_499058	CORRECT	White
007	2013_03_17_12_48_24_338224	button1-press	button3-press	2013_03_17_12_48_37_354654	INCORRECT	White
008	2013_03_17_14_53_20_438590	button1-press	button3-press	2013_03_17_14_53_34_863212	INCORRECT	White
009	2013_03_17_15_06_13_043957	button1-press	button1-press	2013_03_17_15_06_25_565992	CORRECT	White
010	2013_03_17_15_22_17_607862	button2-press	button2-press	2013_03_17_15_22_25_693059	CORRECT	White
011	2013_03_17_15_49_16_225613	button3-press	button3-press	2013_03_17_15_49_21_977289	CORRECT	White
012	2013_03_17_16_18_13_924841	button1-press	button3-press	2013_03_17_16_18_19_795965	INCORRECT	White
013	2013_03_17_16_29_14_719158	button1-press	button1-press	2013_03_17_16_29_21_202002	CORRECT	White
014	2013_03_17_16_40_15_896583	button1-press	button3-press	2013_03_17_16_40_21_633403	INCORRECT	White
015	2013_03_17_16_52_17_723957	button1-press	button3-press	2013_03_17_16_52_31_705759	INCORRECT	White
016	2013_03_17_17_04_15_963706	button2-press	button2-press	2013_03_17_17_04_22_467897	CORRECT	White
017	2013_03_17_17_22_17_300355	button3-press	button2-press	2013_03_17_17_22_22_520951	INCORRECT	White
018	2013_03_17_19_37_46_166803	button1-press	TIMEOUT	2013_03_17_19_38_15_833813	TIMEOUT	Black

Table 3: Example of the result CSV file

5.1.2 Reduced time & effort of trainer/experimenter

First of all, human presence is not necessary. Data transfer from one computer to another, maintenance, or modification of the system requires human interaction, but no time and effort is required concerning the training and testing sessions.

Because no one attends the sessions, a periodic analysis of the animal's performance with the system is required. A simple assessment of how much food the animals took, or more specifically, how many correct and incorrect trials occurred, can be done quickly since this information is already stored in result CSV file displaying the number of correct and incorrect trials generated with timestamps at the end of each session. Table 3 is an example of this result file. A more de-

tailed assessment, such as watching recorded videos, takes some more time, but the total amount of data is already greatly reduced as described previously. Also, the aforementioned data-viewer utility program displays all the timestamps and its JPEG image, which presents a brief report on the movement detected in the recorded video-clip as shown in Figure 16 . Thus, simply browsing the JPEG images is often enough to assess the session. If it is not enough, then one can obtain a more detailed assessment by playing the video-clips recorded around the trial times.

5.2 Results of training phase #1

This phase lasted for about three weeks. The task was simple and the food was clearly visible and could be smelled. Both cats learned that they could obtain food reward after the sound playback. The female cat showed reliable responses in a few days as depicted in Figure 23

5.3 Results of training phase #2

This phase lasted for about four weeks. Both cats avoided pressing the button. Replacements of the button in various ways were tested for a few weeks, in order to entice the cats to press the button. The final position of the button is shown in the Figure 24. The male cat's training was not considered any more at this point because he never succeeded in learning to press the button more than a few times during this period.

After the female cat succeeded in learning to press the button, the feeding algorithm was re-programmed so that only when the feeding sound for the button1 was played, the button could be pressed in the following 60 seconds. Otherwise, the button-press was ignored. The cat did not have much difficulty with this. The cat pressed the button 12 times with the food reward on the first session as shown in Figure 25. This success is due to her many attempts at obtaining food reward with a number of button presses. These incorrect attempts with a number of button

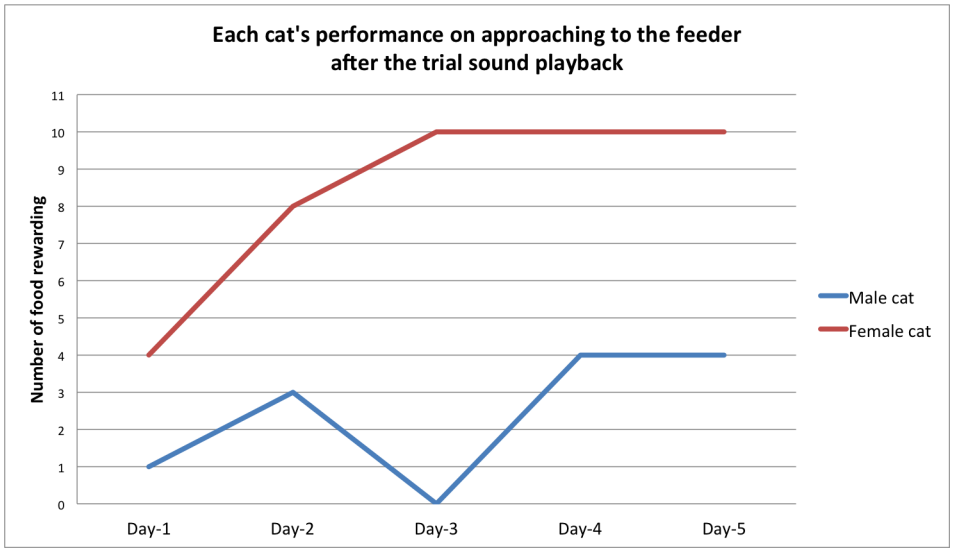

Figure 23: Performance in the training phase #1; * Approximately 10 instances of food rewarding were enough amount of food for a day for a cat. ** The male cat's performance on the day-3 was 0, because only the first cat approaching the feeder was counted. However, he approached the feeder right after the female cat several times.

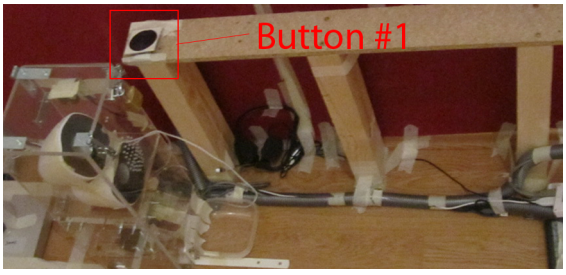

Figure 24: The position of the first button in the phase #2

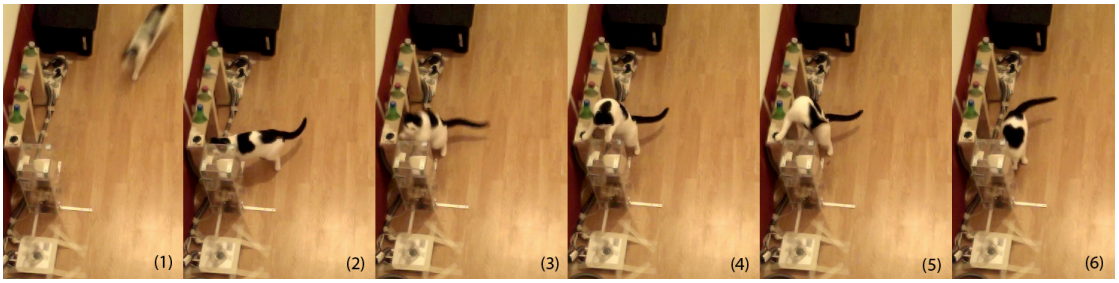

Figure 25: The cat obtaining the food reward in the training phase #2; (1) The cat is approaching to the feeder after hearing the feeding sound (2) When she was moving in ROI for motion-detection, the feeding sound was played once more (3)&(4) The button was pressed. (5) Food was dispensed

presses decreased quickly over the next three days as shown in Figure 26.

5.4 Results of training phase #3

This training phase has already lasted for about four months and its goals have still not been achieved so far. In this phase, the number of buttons was gradually increased to three. The female cat could press any of the three buttons to obtain the food (Figure 27). Over a month the female cat associated the feeding sound, a button-press, and the possible food-reward. Although she pressed any of the three buttons, she failed to associate the three different feeding sounds with three different buttons. Various parameters and methods were tested to facilitate the association of each feeding sound with each button, but these failed to guide the cat.

After a period of about two months, three speakers were implemented behind each button. The result data obtained after this modification was implemented, along with other important setup changes, are depicted in the Figure 28.

The performance did not reach above 75% over a few days in a row since the loudspeakers were elevated to 60 cm on March 5th, which implied that the cat could successfully distinguish the correct button by the localization of the sound, but did not associate each of the three sounds to a specific button yet. A foil stim-

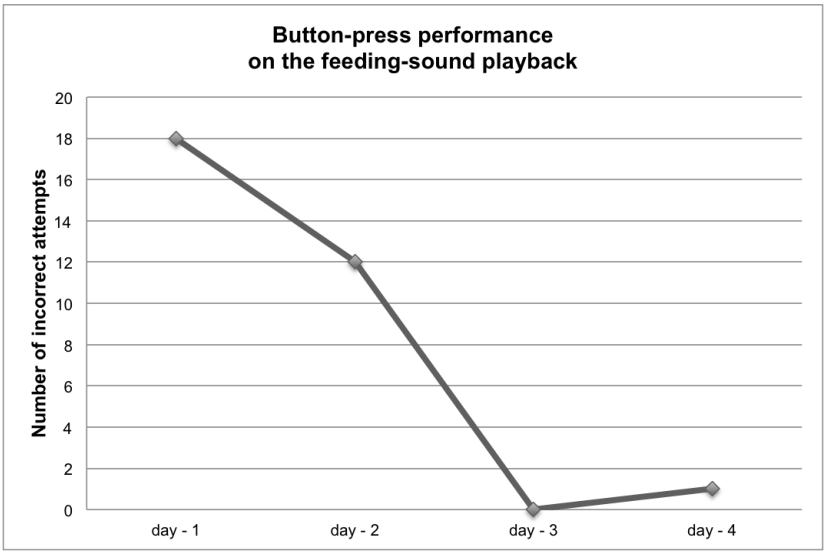

Figure 26: Button-press performance after re-programming of the feeding algorithm; One incorrect attempt means that the cat pressed the button outside of a trial period. (The trial period lasted for 60 seconds after the start of the trial.)

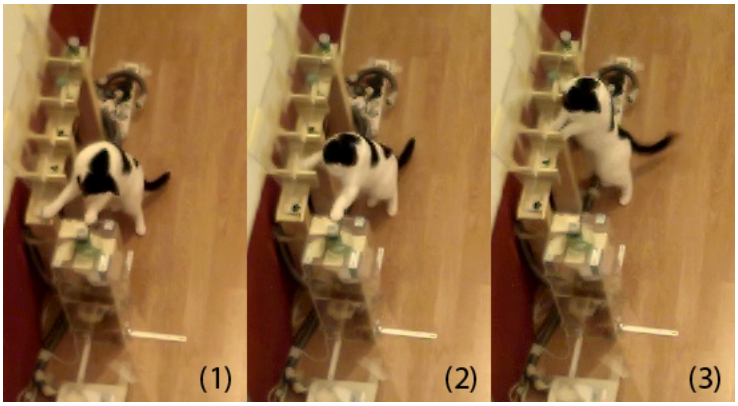

Figure 27: The cat pressing each button to obtain the food reward

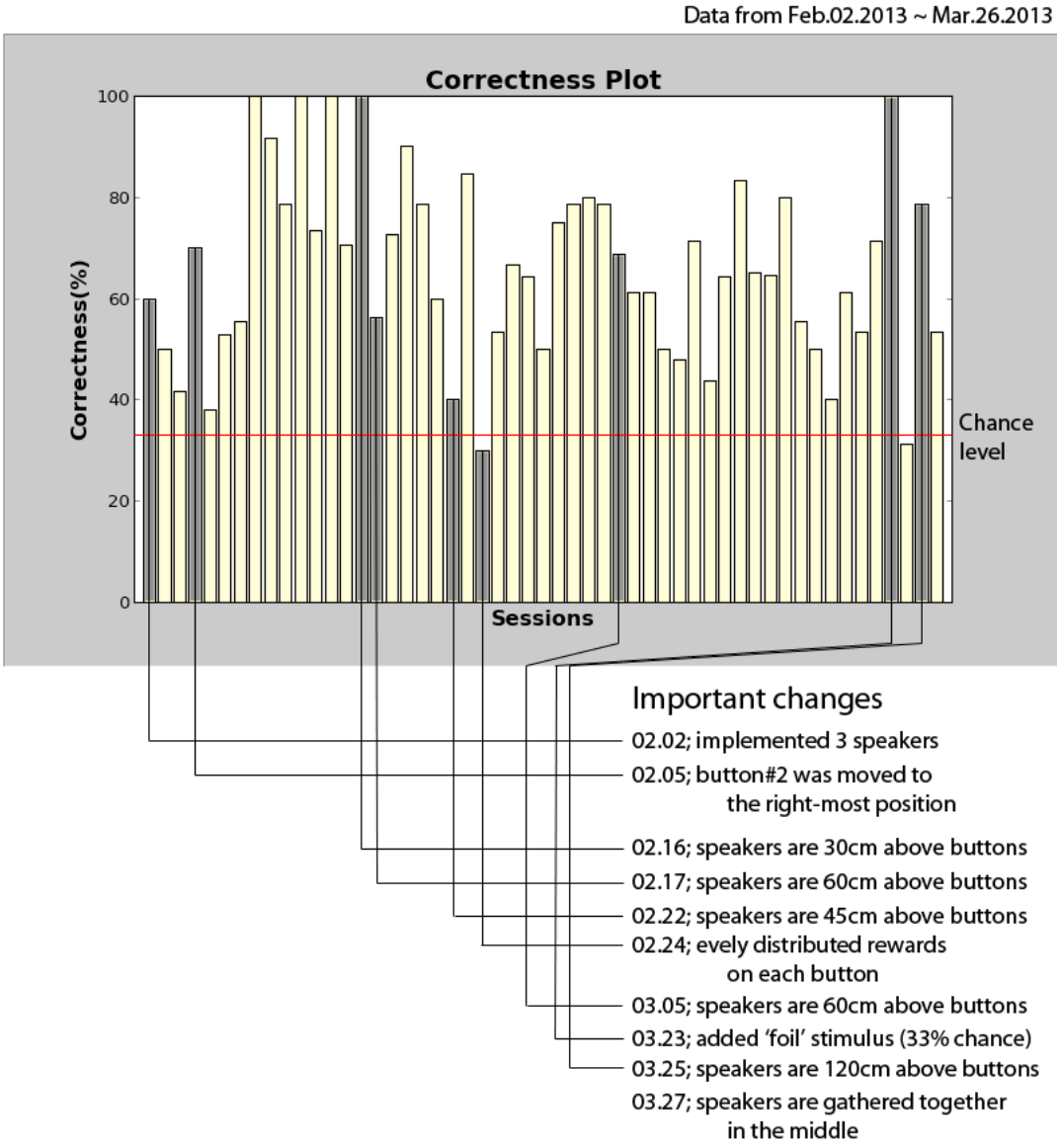

Figure 28: The result data of the three buttons & three speakers with some important changes; * '02.24; evenly distributed rewards on each button' means that the system counted the correct-responses(food rewards), and if the reward was released more with a certain button-press, that button-trial would be removed from the next random trial selection. ** '03.23; added foil stimulus (33% chance)' meaning that, at the random stimulus selection, the system played the 'foil' stimulus with 33% of chance. As soon as any motion was detected during this trial, the negative feedback sound was played.

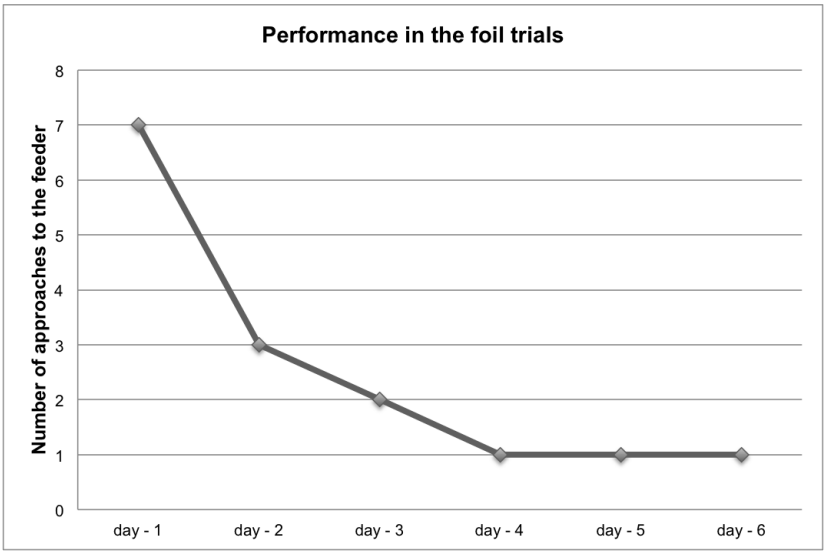

Figure 29: Performance in foil trials: the subject quickly learned that no food was available when the foil sound was played, and stopped approaching.

ulus was implemented on March 23rd to test whether the cat could pay attention and distinguish the sound whose prosody is different from the other three sounds. This foil stimulus was also the experimenter's recorded speech, 'Yuk-gak-hyeong', which is similar in duration and mean intensity with other speeches of the three buttons, however, different in prosody. The female cat showed a fast learning curve in discriminating this foil stimulus as in Figure 29.

6 Conclusions & Future directions

In this project, I successfully designed and built two versions of an automatic animal observation and training system called CATOS, and tested this system in a pilot project with domestic cats, implementing an auditory discrimination task.

6.1 Building CATOS

- As discussed in section 5.1, the large amount of data storage and human resources required to train animals and analyze the stored data were reduced by the use of CATOS.
- During the testing period, the system successfully acted as a trainer starting a trial, waiting for a given time duration to accept the response, dispensing the food reward for the correct response or giving negative feedback(sound/light) for the incorrect response.
- It was not necessary to take an animal to a separated place or remove it from its group and/or its own living space for training sessions.
- Modularity has been the main goal of the system from the beginning. The software and hardware of the system are functionally divided for easy adjustments, extensions or modifications in the future.

6.2 Testing the system: Training cats

During this training on cats for testing the system, both cats learned that approaching the feeder on a playback sound could lead to a food reward. Then the female cat further learned that pressing one out of three buttons could lead to a food reward. She also showed a different behavior to a foil sound stimulus as shown in figure 29. The training of the association between each sound stimuli and each button is an ongoing process as depicted in the figure 28. Although her performance on discriminating 3 sound stimuli fluctuates to a great extent, her performance is often significantly high ($>$75%). Perhaps with enough time, this individual will fully learn the task.

7 Further work & Discussion

I now discuss some changes and additions that would be worthwhile in future implementations.

7.1 Access restriction to ROI zone

Although the animal should not be separated from its own residence or group, the system would benefit by having only one subject at a time in its ROI (region of interest) zone. The benefits of this would include individual identification, individually different tasks and rewards, and so forth. The procedure allowing only one individual in the ROI should be performed by animal subjects on their own in order to minimize their stress level. Also, the individual in the ROI should be able to leave the zone at any time. This access restriction system could be introduced in the following order.

1. CATOS is first implemented without the access restriction system.
2. Once the subjects are used to obtaining food from the feeder, a transparent tunnel-unit for access restriction would be implemented.
3. A few more transparent tunnel-units are attached in a row if necessary.

An example of the proposed access restriction tunnels is depicted in Figure 30.

7.2 Individual identification

In this study, when it was necessary for the analysis to recognize which animal was responsible for a specific trial, this was determined from video files manually after the session. With the help of the aforementioned program, AA_DataViewer, these determination did not require much effort. However, if the goal is to have the system respond differently to each individual, it will be necessary for the system to recognize individuals automatically. RFID (Radio frequency identification)

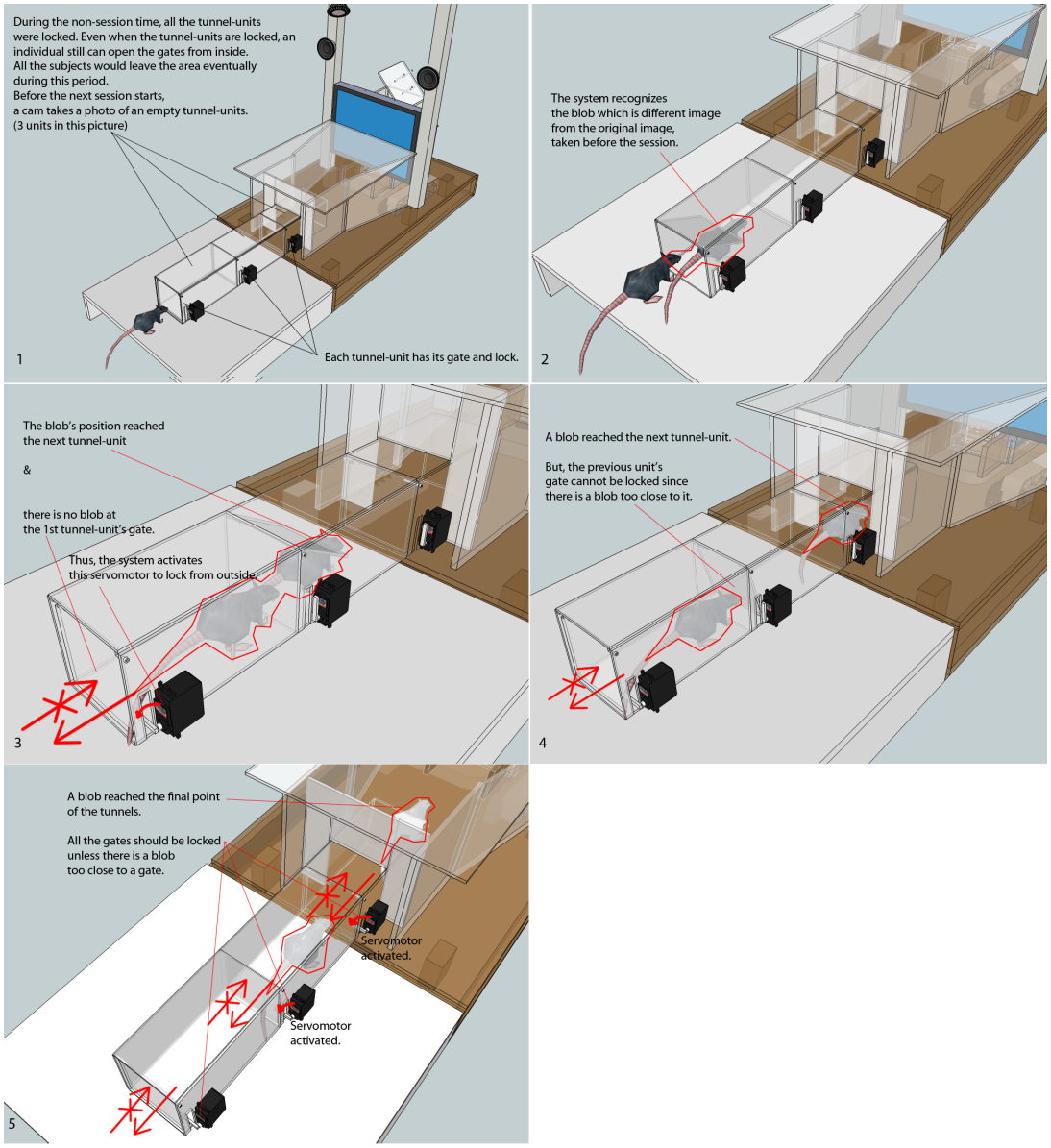

Figure 30: Example of using tunnels for restricting access; * In this example, 3 tunnel-units are assembled. ** A gate of each tunnel unit can block animals entering into the tunnel-unit from outside (distal side from the device). An animal inside a tunnel would be allowed to leave the tunnel-unit (away from the device) at any time.

technology or a computer vision technology exploiting a specific color or shape are both being considered. Either technology, or both, might be used in the future depending on the type of animals.

7.3 Slave units through a wireless network

The observation feature of the current system can cover a small area such as a room used in this study. But if multiple slave units with microcontrollers or single-board computers, which can send information to a master computer, are implemented in different places, the system can expand its scope to a larger and more complex area. This multi-unit system could prove particularly useful for animal observation tasks.

7.4 Testing with other animals including other species

In this study, CATOS was tested only with cats, but the system is not designed for any specific species. Therefore it will be helpful to improve the system to test it with several completely different species. Currently, rats and marmoset monkeys are being considered. Figure 31.

7.5 Artificial intelligence

7.5.1 Advanced artificial intelligence implementation

Currently, CATOS is a simple reflex agent in terms of AI(Artificial Intelligence), according to the definition of Russell & Norvig [19]. As a long-term goal, CATOS could adopt more advanced AI technologies. The improvements in AI could include :

- Coping with uncertainties using probability theory; Many aspects of an animal's behavior have uncertainties which are not easily manageable with simple set of production rules.

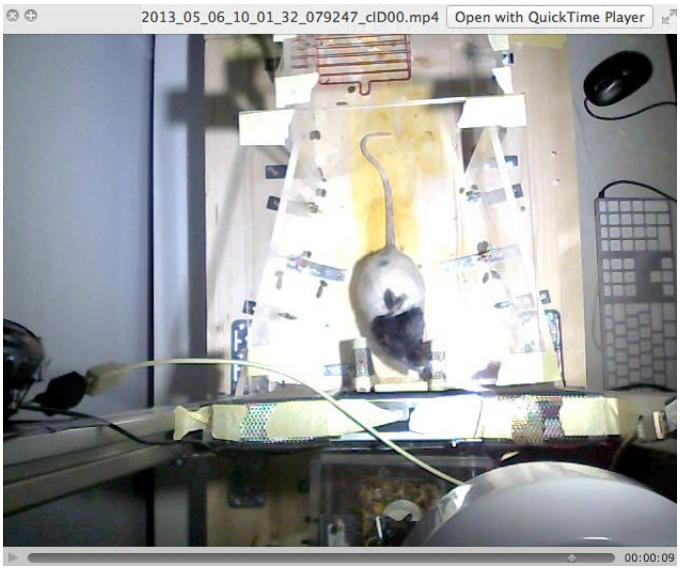

Figure 31: Screenshot of a video-clip generated by CATOS during a pilot test with a rat.

- Decision making and/or planning as an utility-based agent; A utility-based agent could suggest possible next approaches for a successful training based on the analysis of previous result data.

7.5.2 Developing an intelligent agent for interaction with living organisms

The main purpose of CATOS is training animals and/or investigating animal cognition. But, there is another possible side effect of the system considering the development of Artificial intelligence. As Brooks [20] emphasized, testing an agent in the real world is very important in developing an intelligent agent. CATOS system might face more realistic or natural problems than other systems working with humans. When a system is working with humans, the users typically understand the purpose of the system and tend to cooperate to achieve the overall goal of the system (or sometimes intentionally try to break the rules of the system for debugging). However, CATOS will work on animal subjects without human interference and the animal subjects will be blind to the goal of the system. This may result

in valuable challenges for artificial intelligence design and development, beyond those encountered with human users and those not considered in current theory.

References

[1] Mareike Kritzler, Stephanie Jabs, Philipp Kegel and Antonio Krger. (2008). Indoor tracking of laboratory mice via an RFID-tracking framework. MELT '08 Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments.

[2] Nicola Bellotto, Eric Sommerlade, Ben Benfold, Charles Bibby, Ian Reid, Daniel Roth, Carles Fernandez, Luc Van Gool and Jordi Gonzalez. (2009). A distributed camera system for multi-resolution surveillance. Proc. of the third ACM/IEEE Int. Conf. on Distributed Smart Cameras (ICDSC).

[3] David Vallejo, Javier Albusac, Luis Jimenez, Carlos Gonzalez and Juan Moreno. (2009). A cognitive surveillance system for detecting incorrect traffic behaviors. Expert Systems with Applications.

[4] Michael R. Markham, Allen E. Butt and Michael J. Dougher. (1996). A computer touch-screen apparatus for training visual discriminations in rats. Journal of the Experimental Analysis of Behavior, 65, 173-182.

[5] Atsushi Takemoto, Akihiro Izumi, Miki Miwa and Katsuki Nakamura. (2011). Development of a compact and general-purpose experimental apparatus with a touch-sensitive screen for use in evaluating cognitive functions in common marmosets. Journal of Neuroscience Methods, 199, 82-86.

[6] Brian D. Kangas and Jack Bergman. (2012). A novel touch-sensitive apparatus for behavioral studies in unrestrained squirrel monkeys. Journal of Neuroscience Methods, 209, 331-336.

[7] Michael Morten Steurer, Ulrike Aust and Ludwig Huber. (2012). The Vienna comparative cognition technology(VCCT): An innovative operant conditioning system for various species and experimental procedures. Behavior Research Methods, 44:909-918.

[8] Joel Fagot and Dany Paleressompoulle. (2009). Automatic testing of cognitive performance in baboons maintained in social groups. Behavior Research Methods, 41(2), 396-404

[9] Joel Fagot and Elodie Bonte. (2010). Automated testing of cognitive performance in monkeys: Use of a battery of computerized test systems by a troop of semi-free-ranging baboons (Papio papio). Behavior Research Methods 2010, 42(2), 507-516

[10] Juliane Kaminski, Josep Call and Julia Fischer. (2004). Word learning in a domestic dog: evidence for 'fast mapping'. Science, 304.

[11] Lisa A. Heimbauer, Michael J. Beran and Michael J. Owren. (2011). A chimpanzee recognizes synthetic speech with significantly reduced acoustic cues to phonetic content. Current Biology, 21.

[12] Irene M. Pepperberg. (1987). Evidence for conceptual quantitative abilities in the African grey parrot: labeling of cardinal sets. Ethology, 75.

[13] W. Tecumseh Fitch. (2011). Speech perception: a language-trained chimpanzee weighs in. Current Biology, 21.

[14] Gary Bradski and Adrian Kaehler. (2008). Learning OpenCV: Computer vision with the OpenCV library. ISBN: 978-0-596-51613-0

[15] John Canny. (1986). A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 8: 679714.

[16] Satoshi Suzuki and Keiichi Abe. (1985). Topological structural analysis of digital binary images by border following. Computer Vision, Graphics and Image Processing 30: 3246.

[17] Steve Young, Dan Kershaw, Julian Odell, Dave Ollason, Valtcho Valtchev and Phil Woodland. (2006). The HTK Book. Cambridge University Engineering Department. http://htk.eng.cam.ac.uk.

[18] Paul Boersma and David Weenink. (2009). Praat: doing phonetics by computer (Version 5.3.05) [Computer program]. Retrieved Feb. 21st, 2012, from http://www.praat.org/

[19] Stuart J. Russell and Peter Norvig. (2010). Artificial intelligence. A modern approach. Third edition. Pearson Education.

[20] Rodney A. Brooks. (1991). Intelligence without representation. Artificial Intelligence 47: 139-159

Appendices

A Abstract

A.1 Abstract in German

In der Ethologie sind mehrere Fälle denkbar, in denen eine autonome Beobachtung von Tieren/ein Trainingssystem nützlich sein würde. 1) Zur kontinuierlichen Beobachtung bestimmter Arten, oder zur Dokumentation spezifischer unregelmäßig auftretender Ereignisse; 2) Zum Langzeit Training von Tieren in der Vorbereitung von Verhaltensexperimenten; und 3) Zum Trainieren und Testen von Tieren ohne menschliche Einflussnahme um potenzielle, unbeabsichtigte Hinweise und daraus folgende Datenverfälschungen durch den Experimentator zu vermeiden. Das primäre Ziel dieser Studie ist es, ein System namens CATOS zu erstellen (Computer Aided Training / Observing System) das in den oben genannten Situationen verwendet werden kann. CATOS wurde als ”Proof of Concept” gebaut und in einem Pilotversuch getestet, in dem Katzen geschult wurden drei Tasten zu drücken als Antwort auf drei verschiedene Töne (menschliche Sprache), um Futter als Belohnungen zu erhalten. Nach der Erstellung des Systems wurde es im Laufe von ca. 6 Monaten kontinuierlich verbessert und zwei Katzen konnte erfolgreich trainiert werden.

A.2 Abstract in English

In animal behavioral biology, there are several cases in which an autonomous observing/training system would be useful. 1) Observation of certain species continuously, or for documenting specific events, which happen irregularly; 2) Long-term intensive training of animals in preparation for behavioral experiments; and 3) Training and testing of animals without human interference, to eliminate potential cues and biases induced by humans. The primary goal of this study is to

build a system named CATOS (Computer Aided Training/Observing System) that could be used in the above situations. As a proof of concept, the system was built and tested in a pilot experiment, in which cats were trained to press three buttons differently in response to three different sounds (human speech) to receive food rewards. The system was built in use for about 6 months, successfully training two cats. One cat learned to press a particular button, out of three buttons, to obtain the food reward.

B Curriculum vitae

B.1 Profile

Enthusiastic, dedicated, diligent, organized, problem solver with good written and verbal communication skills and work well both independently and in a team environment. I have had a great interest and enthusiasm in animal's cognition and various types of intelligence since 2005 when I transferred to the biology department of the Andong National University. My ultimate goal has been a researcher focusing on studying cognition since then.

B.2 Education

- University of Vienna/ Cognitive Science (MEi:CogSci)/ Master's degree/ Sep. 2011 - Present
- Andong National University/ General Biology/ Bachelor's degree/ Mar. 2005 - Feb. 2011
- Shinheung College/ Computer Science - Multimedia/ Associate Degree/ Mar. 1998 - Feb. 2001

B.3 Scholarships

- Andong National University/ High GPA Scholarship (Partial)/ 2nd semester in 2010
- Andong National University/ High GPA Scholarship (Full)/ 1st semester in 2010
- Andong National University/ High GPA Scholarship (Full)/ 2nd semester in 2005

- Shinheung College/ High GPA Scholarship (Full)/ 2nd semester in 2000
- Shinheung College/ High GPA Scholarship (Partial)/ 1st semester in 2000

B.4 Work experience

- University of Vienna/ Vienna (Austria)/ Python programmer/ Jun. 2011 - Present: While I have been working in University of Vienna, I have been programming for various experiments on human and animal species. Additionally, I worked on some minor engineering and electronic engineering tasks because these were closely related problems in many animal experiment cases.
- Seoul National University/ Seoul (S.Korea)/ Research intern/ Jul. 2010: In Seoul National University internship period, I worked on video coding and simple scripting using Excel macro function. The video coding was conducted on experiment video clips for Mexican Jay (*Aphelocoma wollweberi*).
- Ridgewood veterinary hospital/ New Jersey (U.S.A.)/ Vet. assistant/ Sep. 2007 - Oct. 2009: In Ridgewood veterinary hospital, I assisted in various medical practices on different animal species. These include restraining, care-taking, EKG, taking and developing X-ray, drawing blood, preparing blood, urine, and stool samples, running blood tests such as CBC, Blood chemistry, and Electrolytes, injections via SQ/IM/IV, administering various medications, surgery preparation and assisting including vital sign monitoring, handling various surgical instruments, and sterilizing surgical instruments.
- Andong National University/ Andong (S.Korea)/ Lab assistant/ May. 2005 - Dec. 2005: In Andong National University, I helped graduate students for collecting and identifying sample insects.
- Taxworld/ Seoul (S.Korea)/ Web programmer/ Jun. 2000 - Oct. 2002: In Taxworld, I developed several web-sites using either PHP or ASP and MySQL or MS-SQL database.