Feature detection as an indication of attention

Collin Edington
Russell Lab
McGowan Institute for Regenerative Medicine
Department of Bioengineering
University of Pittsburgh
Email: cde15@pitt.edu

Scott Kennedy
Motor Lab
Center for the Neural Basis of Cognition
McGowan Institute for Regenerative Medicine
Department of Bioengineering
University of Pittsburgh
Email: sdk29@pitt.edu

***Abstract*—This paper presents a program which captures images from a web cam, detects human facial features, displays the results via a graphical user interface (GUI), and saves the data to a file. A case study highlights the program's specific application to non-human primate training in behavioral experiments and operates under the assumption that the animal is paying attention during positive feature detection. The user can choose to detect 1 of 6 different features and define the attention threshold per trial, which determines the percent positive frames constituting a trial in which the animal was inferred to have been paying attention. The GUI displays real-time data from each trial and the log file saves data from every frame.**

I. Introduction

Scientists often use non-human primates to study neural activity *in vivo* by training the animal to perform behavioral tasks and recording the extra-cellular potential of cortical neurons. In a typical experimental set-up, the researcher trains the animal to recognize a stimulus and acquire the skills necessary to perform the desired behavioral response. Before collecting experimental data, the animal must demonstrate a certain level of task proficiency, which is currently evaluated according to the total number of trials and success rate. However, proficiency is a function of both skill and attention, a distinction which this method does not address. The purpose of this particular application of feature detection is to provide an improved evaluation of task proficiency.

The most common method calculates success rate as a function of successfull and unsuccessful trials. A successful trial occurs when the animal recognizes the stimulus and correctly performs the behavioral response. However, unsuccessful trials can fall into one of two categories. On one hand, the animal may recognize the stimulus but fail due to an unskilled response. On the other hand, the animal may become distracted and fail by not recognizing the stimulus. Both responses decrease the success rate and increase the total trials. As a result, the researcher does not have enough information to determine whether the lack of proficiency is related to skill or attention. The implementation of feature detection distinguishes unskilled and distracted responses. Our goal is to provide the researcher with a qualitative analysis of the attention of the monkey to adjust training strategies within the session and quantitative data that can be analyzed after the session to determine strategies for future session.

II. Methods

The tracking program utilizes and extends numerous open source libraries, including OpenCV, Qt, and ITK to capture and process images through a graphical user interface. After VGA color images are captured, they are converted to grayscale and facial features are detected using Haar classifier cascades and the cvHaarDetectObjects() function included with OpenCV. This function utilizes Haar wavelets (similar to Fourier transforms) to detect features based on a precompiled set of data contained in an .xml file. In addition to .xml files for basic facial features, OpenCV also includes a Haar training utility that can create custom object classifiers using a large data set of images that are positive and negative for the desired feature. While this approach requires large numbers of image files and intensive processing power, it affords researchers the flexibility to track any definable feature. After tracking, a rectangle is superimposed to indicate the location of the selected feature. The program records the number of successfully tracked frames, as well as qualitatively determining whether the test animal was paying attention in a given trial (based on the slider value). This information is displayed graphically via trial counters and an attention progress bar. Variables are also saved to a comma-separated (.csv) log file for post-processing. The log filele includes the timestamp, trial number, feature being tracked, whether or not the feature was detected, and the time epoch.

III. Results

When the checkbox is selected, the current program successfully tracks all 6 features and displays a red rectangle superimposed around the feature on the captured image. After each frame, the program stores the time stamp in milliseconds, the current trial, the feature being detected, the detection result, and the current epoch in separate dynamic arrays indexed by frame number. The stored variables and the corresponding data types are summarized in table I. The features are assigned numbers, with -1 corresponding to no feature detection and positive numbers corresponding to the order on the GUI. The detection returns a 1 for a success, 0 for a failure, and -1 if the feature detection is turned off. Currently, 3 trial epoch are recorded, beginning with 0, but this parameter could be modified for different experiments.

TABLE I
STORED DATA TYPES

Time stamp	Current trial	Feature	Detect	Epoch
double	int	int	int	int

For testing purposes, the program calls a random number generator at the end of each frame to simulate cycling through three trial epochs. A trial ends at the conclusion of the third epoch. After each trial, the program calculates the number of frames in which the feature was detected during that trial and compares it to the attention threshold defined by the slider on the GUI in 1. If the number of positive frames is greater than the threshold, the animal was inferred to have been paying attention and the successful trial counter increases by 1. If lower than threshold, the unsuccessful trial counter increases by 1. The Save Log button generates a .csv log file with columns corresponding to the recorded variables and rows corresponding to the captured frames. The file is also automatically generated upon exit of the program. When the recorded variables are written to the file, the frame index resets. If the frame numbers reach 1000, the pre-allocated size of the arrays, the data will automatically be written to file and the frame index reset. The current GUI implimentation is displayed in figure 1.

Fig. 1. Current GUI display

IV. Discussion

The program accomplishes the goal of providing the user with both a qualitative and quantitative assessment of human attention. The GUI provides trial by trial data in real time to adjust training methods within the session. The log file provides frame by frame data off-line to adjust training methods for future trials. However, the feature detection algorithm does not detect non-human primate features. As mentioned in the methods, the feature detection algorithm relies on .xml files from a large data set of positive and negative features. The files used in this program are compiled from human data, and it is assumed that similar files compiled from non-human primate data would be sufficient to detect animal features.

When the algorithm successfully detects a feature, the captured frame rate is on average 3 Hz, as defined in the Qt code. However, if the algorithm is unable to detect a feature, the captured frame rate slows down significantly. This is a problem even when compiled in release mode. The current algorithm spends too much time attempting to find the desired feature instead of labeling it as a negative frame and capturing a new image. This is a bug that could be fixed in a future version.

The transition between trial epochs and thus from trial to trial is currently mediated by a random number generator. In order to receive the experimental signal to change trial epochs from the controller machine, the current code would need to be incorporated into the custom software connecting the three experimental machines. It was determined that this would not be feasible within the timeline of this project. However, the custom software does include a number of C++ modules and it would be possible to subscribe our code as an additional module.

V. Conclusion

Our program uses human feature tracking as a successful proof-of-concept to indicate attention. A future version will be optimized for non-human primate features and will provide a graphical history of the past 100 trials to compliment the counters on the current GUI. This will enable the user to determine the pattern of attention for recent history.

A final study will be needed to test for false positives and false negatives. However, in this particular application, our program is merely a training tool and exact statistical accuracy is not required. Instead, the final metric will be judged by the efficacy with which the researcher can train the animal.

Acknowledgment

The authors would like to thank Damion Shelton, for detailed lecture notes and providing the code from which this program was modified.