486 – Research Project – Frankie Pike
Spring 2012 – CSC 486

Developing with the Emotiv Epoc

As part of a class on human-computer interaction, I will be working on the integration of the Emotiv Epoc headset with an interface for an iRobot Roomba which I developed previously. Though the SDK offers quite a few screens which highlight the capabilities of their product, the project will undoubtedly be more interesting if it can extend beyond just animations on the screen. This is a critical proof-of-concept for my thesis and should be quite useful toward that end. For anyone reading this after spring quarter 2012, I hope that this guide may offer you some insight into some development challenges, solutions I stumble across, and perhaps save you some time and frustration. This is my first real dive into developing with the product rather than attempting to use it via the Control Center.

If, like me, you intend to utilize the Cognitiv functions which map "brain states" to pre-trained output, you'll undoubtedly want to begin training a user profile in the SDK's control center. After installation, you can find the application in the Emotiv Research Edition's program folder. This program offers a good introduction to the device's capabilities and the types of methods you'll have access to in the SDK. Out-of-the-box, the Expressiv suite seems to have very good training data. It can match many facial expressions which is encouraging. Cognitiv needs training of both the neutral state and any other states you may want to map. Emotiv suggests this process be done iteratively, training 1 state with a good degree of reproducibility, and then adding another. You can train a maximum of 4, though in practice I had very limited success with 2. As part of this class, I will be conducting user studies which should shed some more light on how to more effectively train the models. Right now that's a serious concern of mine, but I'm encouraged by videos of users doing fairly impressive things, supposedly with the Emotiv Epoc:

Controlling Angry Birds: http://www.youtube.com/watch?v=WDgkZZYSVPo

Guiding a Robot: http://www.youtube.com/watch?v=0FPJvauXUJU (seems to be mostly EMG)

TED: http://www.ted.com/talks/tan_le_a_headset_that_reads_your_brainwaves.html

My Project

In the event you're interested, I'll take a little time to tell you a little about my project. I've run into quite a few people who have tinkered with this headset and have not spoken to anyone at CalPoly who has tried to apply it to robotics yet, which was a bit surprising. From the device's marketing material, it is pretty clearly targeted at gamers and most projects have attempted to use the device for on-screen character control. My goal in my thesis is to use this headset as a primary input method for a wheelchair control system. Though this is still a control application in concept, the level of user distraction stands to be significantly greater. There is also a greater time-sensitivity than may be expected in purpose-built games like those highlighted on the Emotiv website. For instance, if the wheelchair adopts a pacman-like behavior, proceeding along according to the last control signal until given another, it is imperative that the user is able to issue stop directives in a reasonable amount of time. It would be simply unacceptable to have them adopt Roomba-like behavior, stopping only after crashing. An on-board AI will attempt to ensure the safety of the user, and prototype versions will, of course, feature a kill switch. Unfortunately, the target audience doesn't have the use of their hands necessary to use safety buttons. If they could, there would really be no need to adopt something as intricate at EEG/EMG headsets. I am very excited by the prospects of this thesis and look forward to working on it more this quarter.

Seriously, Don't Use Tap Water

I have purchased a replacement set of sensors for the headset, and after a good deal of frustration with Emotiv, they have finally arrived. Part of the process of getting the headset set up is ensuring that the sensors are wet and making a good connection to the scalp. Unfortunately, one of the previous groups to use the headset didn't use saline solution as recommended, but instead used regular tap water. Though the sensors will still make a connection with regular water, green deposits begin to accumulate on the sensors and receptacles on the headset itself. It also appears to have corroded the point where the gold sensor affixes itself to the sensor assembly and consequently several sensors have fallen apart. There is also a strange green discoloration on the felt which is a little disconcerting, considering how long the device is on your head. In short, seriously don't use tap water with the new sensors. Saline solution is cheap and doesn't corrode them. Hopefully these new sensors last longer. ☺

Hello World

Bundled with the SDK is a collection of demo applications which can be loaded in Visual Studio. They were developed with an older version, but Visual Studio 2010 was able to convert the projects into a format it could understand. The DLLs bundled with the SDK appear to be compiled for x86 which is a little unfortunate since my development platform is x64. The demos work (for the most part), but the file you ultimately build seems to need to be Win32. Trying to create a build profile targeting the x64 platform was not successful and generated a considerable amount of errors. It also proved helpful to copy the converted project out of the Program Files directory and into the Documents/Visual Studio 2010/Projects folder. At least on this machine, Visual Studio needs to restart itself each time it is launched to get the appropriate permissions for that directory which is disruptive.

After you convert the project, loading EmoTutorials will give you the following samples:

EX1 EmoState logger: Provides an introduction on how to acquire EmoStates from the API. My project does not use this functionality extensively, but from these functions you can supposedly determine certain emotional states like instantaneous happiness, attention, etc.

EX2 ExpressivDemo: Provides an introduction on how to acquire Expressiv data from the API. This is the term the SDK assigns to most of the EMG functions and facilitates tasks like blink detection, winking, etc. Though there is a pre-trained model for any user (which appears to perform relatively well), this model is also trainable on a per-user basis which may output higher quality readings.

EX3 ProfileManager: As noted, most functions in the SDK leverage models which are trained over time. Since this process is highly specific to an individual user, the SDK helps by providing a mechanism for bundling data into user profiles which can be reloaded at a later time. As I expand the system, I intend to look into this functionality more, but a simple system can forego this step.

EX4 CognitivDemo: The Cognitiv functionality is of the greatest relevance to my project, and perhaps the most interesting for developers and users alike. It promises the ability to map "brain states" to output. This demo shows how to both train and recall the models necessary to make that happen.

EX5 EEGLogger: Since CalPoly was kind enough to purchase the "research upgrade" back for the SDK, raw EEG from the headset is exposed. With this data is may be possible to create enhanced processing techniques and not be limited strictly to the black-box methods provided by the Cognitiv suite. Those methods are both a blessing and a curse since they provide a good deal of functionality but are not extensible. The methods in this demo could provide a square 1 start to baking your own functionality.

EX6 GyroData: Conveniently, the Epoc also has a gyroscope which can be accessed through the API. Though there are certainly cheaper gyroscopes, this could be a convenient thing to use in conjunction with the other methods, or even as a verification mechanism (nodding after a command during training). It also had intuitive application in my project, but the hope is to rely mostly on EEG/EMG.

EX7 MultiDongleConnection: The SDK appears to be setup to handle multiple headsets and dongles simultaneously. This is clearly targeted at research users, and probably won't be particularly relevant for most student projects.

EX8 MultiDongleEEGLogger: A more elaborate extension of EX7.

EX9 EmoStateAndEEGLogger: A more elaborate extension of EX1 and EX5.

Together, these provide a good collection of code from which you can get started, assuming you're developing in C++. These demos are also available in Java using the JNI. Since my existing codebase is already in C++, I intend to continue down the DLL route.

Keeping it Classy

While the examples provide a good foundation for understanding the relevant methods, it was actually a bit surprising that they don't bother to use a more class-centered approach to interfacing with the headset. I, for one, prefer organized interfaces for programs and have started to develop a re-usable class for this project. Below is the first version of the source code for my interface with the headset.

```
#ifndef EMOTIV_H
#define EMOTIV_H

#include <iostream>
#include "edk.h"
#include "edkErrorCode.h"
#include "EmoStateDLL.h"

#pragma comment(lib, "../lib/edk.lib")

using namespace std;

class Emotiv
{
private:
	EmoStateHandle eState;
	EmoEngineEventHandle eEvent;

public:
	Emotiv();
	~Emotiv();
	void parseExpression(EmoStateHandle eState);
	void checkForNewExpression();
};

#endif // EMOTIV_H
```

```
#include "Emotiv.h"

// Constructor
Emotiv::Emotiv()
{
	cout << "[Emotiv] Preparing headset interface..." << endl;
	eEvent = EE_EmoEngineEventCreate();
	eState = EE_EmoStateCreate();
	EE_EngineConnect();
	cout << "[Emotiv] Headset ready." << endl;
}

// Destructor
Emotiv::~Emotiv()
{
	EE_EngineDisconnect();
	EE_EmoStateFree(eState);
	EE_EmoEngineEventFree(eEvent);
}

void Emotiv::parseExpression(EmoStateHandle eState)
{
	if (ES_ExpressivIsBlink(eState))
		cout << "[BLINK DETECTED]" << endl;

	if (ES_ExpressivIsLeftWink(eState))
		cout << "[LEFT WINK DETECTED]" << endl;

	if (ES_ExpressivIsRightWink(eState))
		cout << "[RIGHT WINK DETECTED]" << endl;

	if (ES_ExpressivIsLookingRight(eState))
		cout << "[LOOK RIGHT DETECTED]" << endl;

	if (ES_ExpressivIsLookingLeft(eState))
		cout << "[LOOK LEFT DETECTED]" << endl;
}

void Emotiv::checkForNewExpression()
{
	try
	{
		int state = EE_EngineGetNextEvent(eEvent);
		unsigned int userID = 0;

		// New event needs to be handled
		if (state == 0)
		{
			// Get event properties
			EE_Event_t eventType = EE_EmoEngineEventGetType(eEvent);
			EE_EmoEngineEventGetUserId(eEvent, &userID);

			if (EE_EmoEngineEventGetType(eEvent) == EE_EmoStateUpdated)
			{
				EE_EmoEngineEventGetEmoState(eEvent, eState);
				parseExpression(eState);
			}
		}
	}
	catch (const std::exception& e)
	{
		cerr << e.what() << endl;
	}
}
```

It became clear in the examples that EmoStateHandle and EmoEngineEventHandle are of particular importance. I was initially concerned that the API might operate in a polling fashion, where calls to methods like EE_EmoEngineEventGetEmoState might be tightly bound to the time and lost if the call doesn't happen to fall within some limit. There does seem to be a degree of buffering, though an initial look through documentation wasn't particularly enlightening. My concern stems from trying to detect high-frequency blinking as a possible input method, which the Control Center seems to struggle with, even after calibration. The device is capable of 128 samples per second according to specs, but whether or not this is fully realized by the Expressiv functions is unclear. The TestBench does seem to detect the spikes indicative of blinking with a greater degree of success than the Expressiv functions suggesting that a better method may need to be rolled with raw EEG access.

Envisioning Energy

My early attempts to harness the headset proved quite difficult and did not exhibit anywhere near the degree of reproducibility that would be necessary to exert active control over something like a moving wheelchair. Due to the nature of calibration routines, it was an open question of whether my roles as both data collector and provider were a potential source of interference during training sessions.

Another key difficulty was trying to identify which types of “thoughts” lent themselves best to mappings. The documentation for the Epoc suggests that thoughts model something kinetic, like pushing or pulling an object. Another inherent difficulty is that you are asked to hold the thought for a substantial period of time (5+ seconds) which is difficult for more natural mappings (like envisioning a stop sign). In this experiment, I avoided suggesting any particular thought model to participants, allowing them instead to discover it themselves. I was also hopeful that by avoiding priming them, we might uncover something accidently which maps quite well. My favorite of the suggestions from a participant was to envision a ball of energy travelling down your arm. The state was triggered reproducibly once it reached his hand. This process is very clearly a per-user ordeal. I’ll examine what worked and what didn’t work later.

Gyro Drift

After 30 seconds of running an application which uses the gyroscope onboard the Epoc it will become clear that you have to account for drift in the readings. The first version of my program allowed users to press a button on the keyboard to recalibrate the sensor’s position, but from a human-computer interaction perspective this is a bit strange. Below is some code for consideration:

```
void decayFunction()
{
        static int entry = 0;

        // Is this a valid time to reduce?
        if (x && entry++ % DECAY_INTERVAL)
                return;

        // Decay X coordinates
        if (x > DECAY_INTERVAL || x < -DECAY_INTERVAL)
                x -= DECAY_INTERVAL;
        else if (x < DECAY_INTERVAL && x > -DECAY_INTERVAL)
                x = 0;
```

```
            // Decay y coordinates
            if (y > DECAY_INTERVAL || y < -DECAY_INTERVAL)
                   y -= DECAY_INTERVAL;
            else if (y < DECAY_INTERVAL && y > -DECAY_INTERVAL)
                   y = 0;
      }
```

The basic idea of this approach is to have the headset's "neutral" position ultimately converge upon where the user's head is pointing. This relies on the assumption that turns of the user's head are anomalous things which we are trying to detect, and that, as much as possible, when the user's head is facing forward, X and Y should be reading 0.

This model seems to work well for me. I'll be experimenting with this configuration later this week.

State Training

There are a good many threads on the Emotiv support forums dedicated to *what* you should be thinking about or envisioning while trying to train the Cognitiv states. In the end, I couldn't find a truly helpful thread. Most basically concluded that this process is highly user-specific and simply posted what worked for them. I found many of these to be good starting points, but did find some general principles that were consistent across my test subjects.

<u>What doesn't work</u>:

Words: this is unfortunate, partly because if it did work this way, the interface would be incredibly natural. Promises of mind-control carry with it the assumption that the device will accept simply *what* you are *thinking*. In reality, the Epoc is more adept at *how* you are *feeling*. No amount of training while simply envisioning or internally saying words seemed to work for any participant.

Abstract Concepts: Though these seem relatively close to words, it is often cited that colors can also carry a charged emotional context, and I had hope that envisioning colors or similar abstract concepts like "friendship" might carry enough emotion to build a reliable model. In practice, this was not the case.

Concrete Objects: To get away from the other categories, subjects also frequently fixated on concrete objects, like a stop sign (intuitively mapped to stopping the device), or even more specific instances like the stop sign on their walk to class. This is probably still too high a level of abstraction to be reasonably detected via EEG signals (or at least those acquired by devices like the Epoc).

<u>What works (sometimes):</u>

Emotions: Perhaps comically, I happened across this while getting quite frustrated training the device myself. I had spent hours trying to get a reasonable model working and the device seemed bent on staying put. I gave up, acknowledged my frustration, and the device started to move. Apparently what I thought I was training was not, in fact, the most salient thing in the signature… it was my frustration. This was a bit surprising to me at first, but does make anecdotal sense. It also advises some caution about ensuring that participants are in a relatively calm state during training, lest it interfere with the model you're working so diligently to build.

Kinesthetic actions: It comes as little surprise that the SDK's training suite is so insistent on using kinetic metaphors for training actions, rather than simply user defined fields. In practice, participants noted that it was substantially easier to hold onto a single thought longer when it involved some mental "exertion". Envisioning themselves pushing or pulling something worked well and is likely a good place to start with future participants.

Meditative Activities: If you've had any exposure to meditation or yoga, clearing the mind is a common goal. As part of this, breathing activities and focusing on individual body parts are common. During testing, some participants used these practices with a surprising degree of success. It also likely helps to avoid the emotional contamination that was noted previously.

In the end though, I do think it is important to qualify "what works" with "sometimes". There really wasn't a universal trigger that worked well across participants, but once they ultimately began mapping things to motion or meditation, all participants were ultimately able to exert some degree of control over the system.

The Viability of EEG Control with Epoc

I recently concluded a usability study with several participants over the course of many hours trying to build reliable EEG models for control. The following were the results of that evaluation:

The results of this evaluation were disheartening to say the least. Even dismissing the time necessary for initial setup, the time required is nowhere near the 5 minutes Emotiv claims is needed before the user can already exhibit a degree of control over the system. In reality, the time necessary to build the model should be less of a concern for end users than the fact that within an hour the sensors dry out, causing degradation in signal quality. While it is easy for an experimenter to rewet the sensors, the target audience is supposedly using the device because they do not have effective use of their hands. To date, Emotiv does not offer a dry sensor and competing products (within Emotiv's price point) have significantly fewer sensors and do not even attempt to offer the state mapping Emotiv does.

I am slightly validated in one of my hypotheses: subjects should be able to exhibit a greater degree of control with the headset than I should. I do not claim this as a matter of self-deprecation, but likely as a result of the effects of trying to be both the subject and the tester. Wielding the control center application is, in some senses, non-trivial and there is considerable visual stimulation on-screen, including timers. As one attempts to train the system, you cannot help but notice elapsing timers, filling progress bars, and other dynamic UI elements. The effects of these distractions are likely to be nonnegligible, especially as you attempt to center focus on a single thought. Though you can force yourself to look away from the screen, close your eyes, etc., these are all still themselves active processes which detract from the focus the application demands. The need to context switch between controlling the computer and your own thoughts must happen within 2 seconds (the amount of data which the trainer discards from the time training starts) and it is something I believe I have difficulty with, even after many hours of logged time. Seeing the other subjects developing a higher degree of proficiency at a much faster pace was reassuring, and I believe largely due to the ability to focus solely on their own thoughts.

Nonetheless, even for subjects freed of the need to observe and control the computer, the degree of success in reproducing more than 2 Cognitiv states was essentially zero. Repeated training was of no assistance and the more we focused on training the later states, the more difficult it became to reproduce the first two states. Fatigue and frustration may well have been a contributor to the declining abilities of the subjects, though everyone claimed they were having a good time. With the inability to reproduce the Cognitiv states during training, it did not make sense to proceed forward with attempting to test active control. The few times participants were prompted in a fashion similar to what would be expected of them in the active testing phase, excessive periods of time (greater than 30 seconds) were spent waiting for any state to be triggered. Nonetheless, it is possible that testing on a different day, with subjects who are already familiar with the process might prove more fruitful. The documentation does suggest that training success rapidly diminishes after an hour.

In the end, the outcome is troubling to the prospects of precise wheelchair control with low cost EEG headsets. Many of the research papers on similar projects use substantially more expensive hardware which requires conductive gel. Commentary on collaborative EEG projects does note that the Emotiv product has notably poor signal impedance relative to “medical/research-grade” EEG devices (interesting further reading: http://www.physiologicalcomputing.net/?p=1191). In spite of the claimed limitations, there are still plenty of videos showing people exhibiting an impressive degree of control over the device and it may just be a matter which requires an even greater time investment. As a practical matter, however, I am most concerned with the ability to issue directives reliably as there are clear safety implications when a user’s motion is dictated by such a system. Even extending the greatest observed control across all 4 actions, active control 34% of the time is still clearly unacceptable for a primary input device.

If you’d like to learn more about that project, the full report is available at:
https://wiki.csc.calpoly.edu/CSC-486-S12-06/wiki/Assignments

Not Everyone’s a Gamer

While demoing the gyro component of my project today, I had several subjects note that the inverted forward, backward mapping was strange to them. Most users were able to adapt to this mapping, but one in particular was very adamant that it was completely nonsensical. I agree that when you say it aloud (tilt your head backward to drive forward, and tilt your head forward to drive in reverse) it does sound counter –intuitive, but for those familiar with video game controls, the mapping was as expected. While it would not have been difficult to incorporate a user-selectable mapping in the prototype, the thought hadn't dawned on me because the device operated in a manner which felt intuitive. In reality, this was simply a matter of conditioning which is worth addressing for the comfort of future users. I believe that this is a good thing to keep in mind for future development.

Visual Evoked Potentials

In my research following the relative failure of simple state mapping, I happened across a process which seems to be showing very positive results: visual evoked potentials. Commercial products already exist which implement this technique, and more are expected to be released in the next several months (Summer 2012). I've outlined two of the common methods below.

Visual Evoked Potential (VEP/P300): Rather than attempting to map states with actions, VEP attempts to distinguish between available actions based on a single metric: user attention. Available options are arranged in a grid, with rows and columns being highlighted in intervals. The underlying methodology is that users will fixate on a single action and when it gets highlighted, ~300ms (hence P300) thereafter, a measurable peak will occur in their EEG. Subsequent highlights then attempt to reduce the number of possibilities in the set until the action is identified. Encouragingly, there is already a commercial product implementing such a system (http://www.intendix.com/). Their website suggests that new users are able to enter between 5 and 10 characters per minute on the system. However, their product fully emulates a keyboard, and a system with substantially fewer keys could exhibit higher throughput. Open source packages centered on EEG discuss implementation details for such a system and suggest that such a system may be implementable on the Epoc. Since it is only relies on "perked attention" vs. "non-attention", it may also be possible to implement it with less elaborate, dry sensor EEG devices like those offered by NeuroSky, thereby avoiding the time limitations of wet sensors.

Steady-State Visual Evoked Potential (SSVEP): This method shares the same principle as P300, but attempts to find matches between a display item's refresh rate and frequency responses in the brain. The same company noted in the P300 case, intendiX, also has a product offering set to arrive this year based on this method. They note "up to" a 98% accuracy and have apparently demonstrated it in controlling avatars in the popular World of Warcraft game. Such motion control has direct applicability to my intended project, though I do have some doubts about the ability to bring such a technology outside. Being so heavily tied to refresh rate, it is unclear what the effects of a display that is more washed out might be. P300 lends itself to high-contrast UIs which should still be discernible assuming an outdoor screen is used. Nonetheless, it might still be an interesting area of research to explore.

This will likely be the direction I pursue in my thesis, whether or not this is realizable on the Emotiv is another story. Some P300 solutions have been demonstrated on the Epoc which is encouraging.