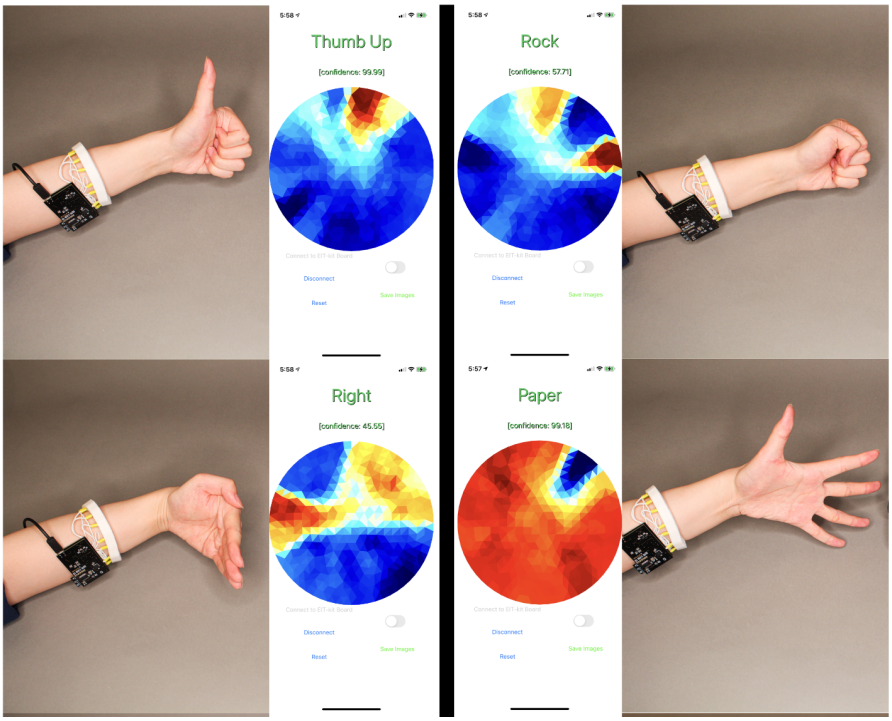
EIT Kit Mobile

Joshua Verdejo



Abstract

Traditionally, electronic devices have some type of physical input. Whether those are the buttons, triggers, and joysticks of a video game controller, or the flat, multi touch screens of mobile devices we have become used to, technology has nearly reached a boundary on these physical and visible inputs. While developments continue to be made, such as pressure sensitive displays and flexible touch surfaces, there is an entirely different aspect of technology that has not been as utilized: the invisible input. Electrical Impedance Tomography (EIT) devices measure internal muscular changes through conductivity changes measured by noninvasive electrodes worn on a wristband. In this way, users are able to imagine pressing a button, or simply make a gesture in open space, and have that gesture correspond to some command. However, these gestures do not have to be sophisticated at all -- the simple act of flexing and relaxing a muscle would be enough to generate a signal to respond to. Because of this, EIT technologies represent tremendous potential to create new interaction interfaces; both in terms of new interactions, and making older interactions more accessible as well, since the movements required to interact with an EIT based system would be much less involved. This project looked to create a novel method of interacting with EIT based devices, moving the interaction to a mobile medium in the form of a mobile API. By using a mobile device and focusing on human computer interaction, more specific features can be implemented and are explored in this paper.

Introduction and Overview

Electronics have made our lives easier since their inception. However, as the time went on, one of the primary motivations in technological design has become more emphasized: user intuition. Making experiences that are not intentionally new, but are more adaptive to user intuition, are what many would consider the mark of a great design. Being able to pick up a device, and understand how to use it before reading any documentation, tutorials, or manuals is the goal in many technological designs. From mobile devices to appliances, we have grown accustomed to interactions that make sense (touch screens, voice assistants who you speak to using common words, etc.). Technological experiences are made to be as intuitive as possible, so as to integrate seamlessly with everyday life through natural physical interactions. However, there are many technologies that have not taken full advantage of the physical interactions we can perform. One such technology is seen in Electrical Impedance Tomography. Electrical Impedance Tomography (EIT) measures electrical conductivity and impedance of a part of the body, and reflects its internal changes due to biochemical activities, like muscle movements. EIT devices are traditionally used for medical imaging. However, by transitioning to applications in human computer interaction, EIT can be reimagined. It can shift from being utilized as a tool primarily for imaging, to a form of "invisible input" where users do not have to click a button or flip a switch, but simply make a gesture or flex a muscle to trigger a command.

While this idea of an invisible input has the potential to change the way these EIT devices are used, there are very few people who would be able to take advantage of these advancements, due to the lack of support in regards to developing an EIT system. Some open source solutions have been attempted , however, finding enough supporting documentation to understand the software alone can be extremely difficult. Because of this, the project is designed to be easily compatible with an existing infrastructure that helps users model and plan out electrical components to build

and develop on their own EIT systems. By giving users flexibility in choosing constraints for the system, a variety of different EIT bands can be modeled and much more easily fabricated. EIT Kit Mobile removes some of the obscurities that comes with developing similar systems, and decreases time needed to prototype new systems.

Because the latest developments in open source EIT visualizations are not accessible for mobile devices, it was important to focus on a new medium to process, visualize, and interact with these signals. The API creates a framework for understanding and developing EIT based applications, and lowers the barrier of entry for future users. The API also creates new methods of connection EIT band interactions with different types of technology. The motivation behind this research is to make interactive technologies more intuitive, and make interactions between users and their digital environments more straightforward. Finally, an element of modularity was very important, so that these new methods of interacting with EIT systems could be further developed, improved, and refined over time.

System Description

EITViz is a tool, in the form of a Swift Package, that allows users to develop apps for iOS and PadOS that utilize EIT technology for human computer interaction, instead of the more traditional medical imaging purpose. By understanding what the system does and how EIT Kit works, users should be able to manipulate the visualizations to their content, and implement a variety of visualizations as needed.

Installing and using the EIT Kit Mobile API is a very straightforward process. Upon importing the package, users can view example view controllers to see how the process is implemented. If manual setup is required, users can simply set up the mesh by calling

mesh_setup, and defining parameters as needed, including shape of the mesh, number of electrodes, and number of terminals. After calling mesh_setup, users will have access to the three objects needed to get the mesh visualized: an EIT object which acts as a solver, reading the input conductivities and solving for the visualization output, and the objects that make up the mesh; one that defines the points in the mesh, and another that explains how to connect those points to form the triangular structure. By calling mapSingle, the solved EIT information is paired with the mesh information to "fill in" the mesh, and the API returns an object that can be displayed on the app.

At this point, the user has a 2D, single band visualization. However, with a few more built in functions, even more information can be read. Users can call mapMulti3D in order to visualize the information in a 3D manner, however, even here, there is choice. Users are returned a three dimensional object that is able to be visualized either entirely on the phone, or in the "real world" through AR. By doing this, users can have insight into the cross sectional volume behaviors of an object wrapped in two EIT bands, in real time. Finally. By looking at the example view controller, users can find a simple method for implementing a basic machine learning algorithm that can be trained to recognize specific types of processed data. Users can save images generated by the app (also included in the example), and use the saved images to train a machine learning model. In this project, training was performed through createML, an Xcode extension that allows for users to simply drag and drop images to train an image classifier. After training the model in createML, users can download the model, drag it into their project, and after appropriately naming it, should be able to see the category and confidence of predictions made by the machine learning model.

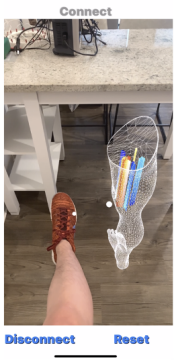
Technical Working Description

Muscle Visualization

This app uses the multi band approach to create a 3 dimensional, AR ready visualization. Using the EIT Kit Mobile Library, the mesh is generated on app open, and users have one interface to connect to the EIT Kit Device, or disconnect from it. Once connected, the EIT Kit Mobile Library multi band visualization tools are utilized, in order to create the 3D visualization of the bands, and overlay the model of choice for a user, in order to show activation areas within the model.

This app works with the EIT Kit Mobile API by utilizing the 3D visualization section of the API. Users wear two EIT bands, sending information from two cross sections of the thigh. The API's 3D visualization tool works to show relevant parts of the thigh, instead of the whole circle. It does that by calculating the average level of activity in the thigh, summing the collective change in conductivity, and dividing it by the number of zones of conductivity in the visualization. Once that is done, users can define a threshold over which information will be considered statistically relevant, and a low threshold based on the average as well, in order to gather information about the lowest and highest areas of activity.

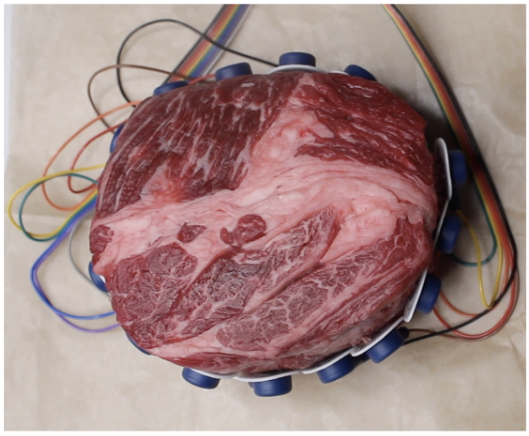
Finally, in this example, the visualization is contained within a wireframe model of a leg. This is done in order to simulate what the bands on the leg of the user are recording. Users can see the changes in conductivity in their leg, in real time. The steps for how to use this app, along with example photos, are shown below.

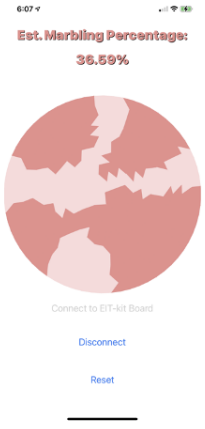


Meat Monitor

This app utilizes the EIT Kit Mobile Library in a very unconventional way. To begin, the mesh is set up as expected, primarily using the mesh setup function. Then, a modified version of one of the functions defined in the API (mapSingle) is used, where, instead of applying colors based on a gradient of values for a continuous appearance, a discrete version of the method is implemented. Numbers are snapped to one of two colors based on their conductivity. If the conductivity falls below a certain threshold, it is colored as fat (a light pink), otherwise, the meat is seen as lean (a darker pink).

Finally, while rendering each frame, each triangle in the mesh is counted, accounting for if it falls above or below the fat threshold. If the color range of a given mesh falls under the category of "fatty" the number of fat regions in the meat is incremented by one (since the triangles are all roughly the same area, this method is relatively unbiased with regard to orientation of the electrodes), until finally, the render is shown on the screen, along with an estimated marbling percentage, which is calculated based on the proportion of the area of the fatty sections of the meat compared to the entire cross sectional area of the meat. The steps that go into these interactions, as well as in app screenshots, are shown below.



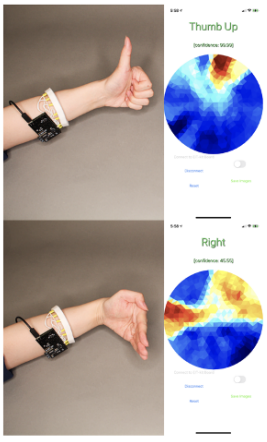


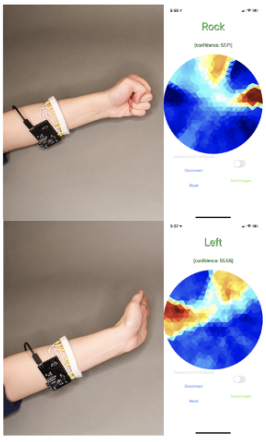
Gesture Recognizer

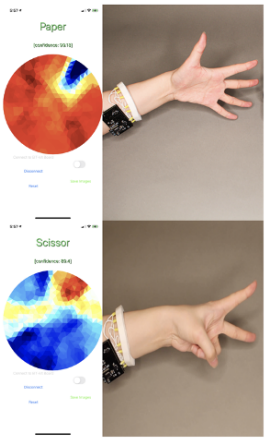
This app is a much more traditional use of the EIT Kit Mobile Library. Users have a single interface for training and running a machine learning option. Upon opening the app, the mesh is created, and on connecting to the EIT-Kit Device, the mesh fills with the information on conductivities read by the system.

Where this app differs is in how it uses these visualizations. While the previous examples focus on allowing the user to see the data, and drawing basic conclusions from it, or extrapolating the information in order to create a more advanced, three-dimensional version of the visualization, this app processes the data to a much more significant extent. This is done by allowing the user to save each frame as it is displayed, saving the images directly to the Photos app on the iPhone. From there, users can upload these images to the CreateML app, that comes built in on MacOS, and train their own machine learning model simply by using drag and drop commands.

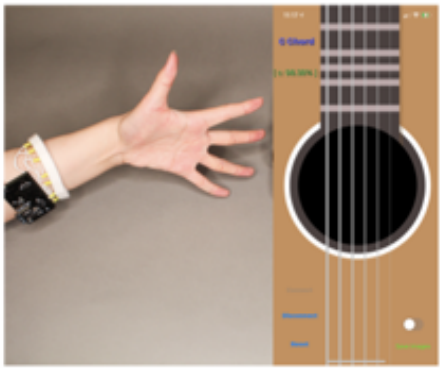
Once the model is done, all the user has to do is download it, drag it back into the Xcode project, and re-run the project. Once the user re-runs the project, the app will use the machine learning model in order to predict which gesture the user is making, based on the model the user just created. An example of a six-gesture classifier, along with in app screenshots, are shown in the following figures.







Air Guitar



This app uses a similar premise as the gesture recognizer, with a bit of a fun twist. Users are able to run the methodology, train the ML model, and upload it in the same way as they do with the Gesture Recognizer app. However, when the machine learning algorithm is uploaded, users now have another way to interact with the app: through sound files. In this iteration, users can match sounds to the gestures they make.

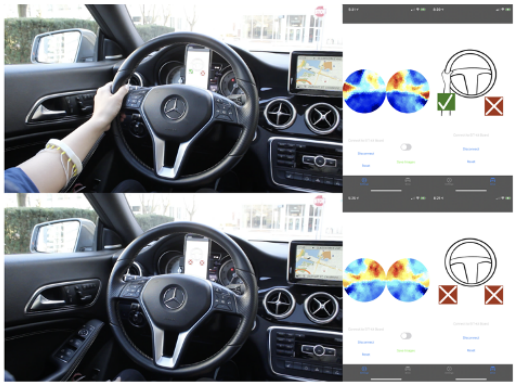
Once users train the ML model and upload it, the only steps the users need to follow in order to play the chord are as follows: The user makes the gesture, and "strums" the string on the guitar, by swiping their thumb across the screen. The intention for the app is recreational, but showcases the various ways that the EIT Kit band can be used for accessibility. The ease of use of the system, combined with the fact that multiple gestures can be loaded into the system and triggered with a simple screen motion, highlight the most promising part of the potential of the library.

The ability to quickly and effortlessly change commands as long as users are able to make large, discernable differences in hand motions, takes away from the smaller, more discrete taps and swipes that users usually have to perform, which can be extremely difficult for certain users, especially those who struggle with fine motor control.

Drive Monitor

This app implements the multiband EIT approach in an unconventional way: by having the user wear the bands on different arms. Users can train a machine learning model for each arm, or an individual machine learning model for both, in the “Settings” section of the app, and apply that model to the “Drive” section of the app. Once the visualization is trained and the user is in the “Drive” section of the app, users receive visual feedback on multiple safety aspects of their drive, such as drive time, and portion of time either hand was off the wheel.

Because both arms are being read at the same time, the multi band visualization functions of the EIT Kit Mobile Library are being utilized for the training visualization, and the gesture recognition as well. However, the multiband hardware currently alternates the sending of data (starting with the left at runtime), so the app does the same, only updating one side of a reading at a time. By increasing the sampling rate, it may be possible to get the visualization to a point where it seems as though the updates are happening at nearly the same time. Alternatively, the user could modify the code such that it connects to two separate EIT Kits, receiving, parsing, and display the data for both at the same time. An example using the single EIT Kit approach is shown below.



Performance and Modifications

One of the initial ideas I had for the 3D visualization was to change the representation of the model to look more like the actual muscular system. There were moments where several methods of implementing these systems, especially computing 3D models, would crash phones simply because of the difficulty in processing that the phones were not able to handle. The processing power of the phone was just not able to handle what was being asked of it. Another interesting performance issue came from the framerate of data being sent. There were many modifications in communication protocols that changed how information could be sent from the band to maximize framerates, such as getting rid of all extraneous characters and ensuring exactly 4 significant figures on every measurement.

Initially, working on this project, I hoped to build in some functionality with Apple Watch and WatchOS. I thought it would be straightforward because I was working on the Apple ecosystem. It turned out to be one of the more interesting failures of the project, as developing for watchOS is a totally separate beast, and would require a much more significant amount of work than anticipated. What would make the work for watchOS much more difficult is not simply the fact that it is a different application, it is also in the fact that many of the functions built into the phone app surprisingly pushed the upper boundaries of the CPU. Due to this, working on the watch would be even more constrained, and would take a considerable amount of code optimization that may have been out of scope given time and level of Swift experience.

Tools Used

The only tools used for development were a Mac and an iPhone. However, to test with the hardware, a custom EIT band is required. Because of this, I also wrote a script that should be able to run on an ESP32 to simulate a band sending information, so the three tools needed would be a Mac to run Xcode and CreateML, an iPhone capable of running AR (I tested with an iPhone XR and it seemed to work well) and an ESP32.