Acoustic Sensing with DiffLogic - Matt Davidson

BeagleBoard.org
Apr 03, 2024

Table of contents

Chapter 1

Introduction

This project aims to leverage acoustic sensing techniques to create a system that can enable rich tactile interaction with digital musical instruments designed on the Bela platform. A machine learning approach will be used, creating a pipeline to train and deploy differentiable logic gate networks to Bela. This framework will be utilised to perform classification of ultrasonic swept sine detection signals. Example code and a framework will be created to use and deploy this acoustic sensing technique within Bela projects, enabling detection of tactile elements such as hand position using piezo transducers. The project will be accompanied by an example digital musical instrument project that exhibits this novel technology in use.

1.1 Summary links

- **Contributor:** Matt Davison
- **Mentors:** Jack Armitage, Chris Kiefer

1.2 Status

This project is currently just a proposal.

1.3 Proposal

1.4 About

- **Forum:** u/mattd (Matt Davison)
- **OpenBeagle:** mattd (Matt Davison)
- **Github:** davisonaudio (Matt Davison)
- **School:** Imperial College London
- **Country:** United Kingdom
- **Primary language:** English
- **Typical work hours:** 9AM-6PM British Summertime (UTC+1)
- **Previous GSoC participation:** N/A

Chapter 2

Project

Project name: Acoustic Sensing with Differential Logic for Tactile Interaction with Digital Musical Instruments

2.1 Description

This project aims to leverage acoustic sensing techniques to create a system that can enable rich tactile interaction with digital musical instruments designed on the Bela platform.

Acoustic sensing is an established human-computer interaction (HCI) technique that measures the change in transfer function between an output transducer and a sensing transducer (piezoelectric disc transducers are often used for this purpose). When the transducer pair is placed upon an object, or group of multiple mechanically-coupled objects, changes such as the object's positioning, user grip strength, and grip position can be sensed. This is due to the changes in the structure-borne resonances. Sensing these changes in resonances is commonly achieved by outputting ultrasonic frequency sweeps, and performing a Fast Fourier Transform (FFT) on the input signal [6]. The FFT values can be fed to a machine learning classification algorithm, with training used to classify changes in the physicality of the object (including continuous changes in object orientation). By repeatedly performing this swept sine output and input sensor processing, near-realtime sensing can be achieved. A significant advantage to the approach of acoustic sensing, when compared to other techniques involving dedicated tactile sensors, is that touch detection can be added to existing objects in an unobtrusive manner.

Though previous research in the New Interfaces for Musical Expression (NIME) community has utilised similar sensing methods [1], continuous realtime methods presented in HCI literature [2,3] has not yet been explored in Digital Musical Instrument (DMI) designs. The creation of a platform enabling acoustic sensing on the Bela platform creates the opportunity for existing Bela users to utilise this approach in future tactile instrument designs. In addition to this, the use of differential logic gate networks will enable Bela users to utilise this particular approach to machine learning and classification within their C++ Bela projects.

The project will have the following deliverables that will benefit the Bela and Beaglebone communities: - A complete pipeline for training (in conjunction with a host machine) and deploying a differentiable logic gate network within a Bela C++ project - A framework and accompanying project for acoustic sensing using Bela, using sine sweep and FFT analysis to feed into a trained DiffLogic model - An example project consisting of a digital musicial instrument design that maps changes in the user's grip and hand positions (using acoustic sensing) to timbral characteristic of a synthesis algorithm.

2.2 Deep Differentiable Logic Gate Networks

Introduced by [5], differentiable logic gate networks provide a novel machine learning network architecture that requires lower computation times at inference time compared to equivalent traditional neural network architectures, whilst achieving similar performance. One significant tradeoff is that differentiable logic networks requires more processing power during the training stage. This tradeoff, however, makes it particularly suited to embedded systems such as the Beaglebone Black and the Bela platform, where the training phase can be achieved in conjuction with a more powerful machine. At inference time, the performance gains achieved by the differentiable logic network are particularly important for the resource-constrained environment of Bela. The use of logic gate networks, in comparision to alternative neural network approaches, leaves more CPU time free on the Bela for tasks such as audio synthesis and processing.

A detailed explanation of the implementation of differentiable logic gate networks can be found in [5], however it is worth briefly discussing soem aspects here within the context of this proposal. While traditional neural networks are trained with varying weights between each neuron, the differentiable logic gate networks are instead trained by selecting the most suitable logic gate (out of a possible 16) for each neuron.

2.3 Cultural and Usage Implications of the Engineering Choices Taken

Though the main focus of this project is upon the software development output and the technical affordances achieved by furthering this technology on Bela, it is worth briefly considering the implications of the engineering approach upon any future creative output using these tools - such as new digital musical instruments. While constraints and affordances of the acoustic sensing approach will be discovered during (and after) the development process, some possibilities can be considered within this proposal. One consideration is that, while the acoustic sensing approach allows continuous sensing of tactile features, the latency involved in performing a sine sweep, Fast Fourier Transform, and DiffLogic classification is likely to be around 100ms.

It is generally accepted that digital instruments should aim to exhibit less than 10ms of action-sound latency [7] to achieve “embodied cognition” - where the instrument becomes a natural extension of the musician. While this figure is found to be higher, at 20-30ms, for continuous gestural interaction [7], it does still pose an immediate constraint on the use of acoustic sensing for DMI design. This, therefore, will also form part of the project evaluation - exploring the effect that this particular constraint has on both the initial design of new instruments and also the ultimate musical interaction with such instruments.

Acoustic sensing also has the interesting creative advantage in that it can impose fewer interaction constraints when compared to traditional forms of tactile sensing - such as force-sensing resistors. As the classification merely relies upon measuring the transfer function between actuator and sensor, additional creative uses that were not initially envisaged in the design process may be afforded - such as detecting objects placed on a surface or the tension of a spring used as an interaction device. Such affordances may give rise to secondary uses for this technology beyond sensing hand-based interactions, which could yield additional affordances of the technique.

To briefly consider these points, Milestone 9 will include a brief discussion and experimentation with a digital musical instrument designer, to discover the design and cultural implications of the system beyond its intended aesthetic.

2.4 Software

- C/C++
- Python
- Possibly PRU Assembly Code

2.5 Hardware

- Beaglebone Black
- Bela cape
- 2x Piezoelectric Transducer discs

- A small assortment of resistors, capacitors, and op-amps for anti-aliasing filtering and buffering of signals
- An audio amplifier for driving the actuation transducer

Chapter 3

Timeline

Provide a development timeline with 10 milestones, one for each week of development without an evaluation, and any pre-work. (A realistic, measurable timeline is critical to our selection process.)

Note: This timeline is based on the official GSoC timeline

3.1 Timeline summary

Date	Activity
February 26	Connect with possible mentors and request review on first draft
March 4	Complete prerequisites, verify value to community and request review on second draft
March 11	Finalized timeline and request review on final draft
March 21	Submit application
May 1	Start bonding
May 27	Start coding and introductory video
June 3	Release introductory video and complete milestone #1
June 10	Complete milestone #2
June 17	Complete milestone #3
June 24	Complete milestone #4
July 1	Complete milestone #5
July 8	Submit midterm evaluations
July 15	Complete milestone #6
July 22	Complete milestone #7
July 29	Complete milestone #8
August 5	Complete milestone #9
August 12	Complete milestone #10
August 19	Submit final project video, submit final work to GSoC site and complete final mentor evaluation

3.2 Timeline detailed

3.2.1 Community Bonding Period (May 1st - May 26th)

GSoC contributors get to know mentors, read documentation, get up to speed to begin working on their projects

- Implement and experiment with Jack Armitage's work on Bela DiffLogic.
- Familiarise myself with PyBela pipeline [4].

3.2.2 Coding begins (May 27th)

- Setup DiffLogic on host computer
- Setup PyBela pipeline, begin integrating this with DiffLogic model training process

3.2.3 Milestone #1, Introductory YouTube video (June 3rd)

- DiffLogic set up on host computer, PyBela pipeline setup, integration of PyBela pipeline into DiffLogic network training in progress

3.2.4 Milestone #2 (June 10th)

- Complete the setup of a training environment whereby data can be recorded on Bela and used to train DiffLogic network on host computer

3.2.5 Milestone #3 (June 17th)

- Create infrastructure within Bela IDE for loading a pretrained DiffLogic model into a Bela C++ project
- Produce a well-documented Bela project that has code in place for both the training phase (using PyBela to transfer data to host computer) and inference, running the model within the project.

3.2.6 Milestone #4 (June 24th)

- Decide upon, and implement, a method of ultrasonic signal generation and capture on Bela (e.g. using the analog at 88.2kHz sampling rate)
- Implement acoustic actuation signal synthesis (sine sweeps), experimenting with variables such as linear or logarithmic sweeps and minimum sweep time required. Some existing work on creating sine sweeps within a Bela project can be found here: Sine sweep generation on Bela

3.2.7 Milestone #5 (July 1st)

- Implement the pre-processing of captured sensor data (Fast Fourier Transform, level thresholding, scaling, etc.) for the generation of training data
- Generate example training data, using acoustic sensing to sense different hand grip positions as a basic test scenario, and test the classification accuracy of the trained model on the host machine

3.2.8 Submit midterm evaluations (July 8th)

Important: July 12 - 18:00 UTC: Midterm evaluation deadline (standard coding period)

3.2.9 Milestone #6 (July 15th)

- Experiment with DiffLogic configurations - network depth, randomly initialised connections etc. - to determine the appropriate tradeoff between computational complexity and network effectiveness for the task of acoustic sensing
- Implement acoustic sensing Bela project where the acoustic actuation, sensing, processing and classification process is done repeatedly, enabling near-realtime sensing.

3.2.10 Milestone #7 (July 22nd)

- Implement continuously variable classifications - where training using data including multiple intermediate object configurations can enable a model with a continuously variable output between two classifications

3.2.11 Milestone #8 (July 29th)

- Hardware design of example instrument - using piezo transducers and Bela integrated into design, along with vibrotactile haptic feedback provided by a voice coil transducer.

3.2.12 Milestone #9 (Aug 5th)

- Implement full Bela project for the instrument design - building upon the acoustic sensing Bela project created earlier, with the continuous classification network output providing mapping parameter for the timbre of the sound synthesiser.
- Conduct a short evaluation of the affordances of the technology by presenting the system to an instrument designer to experiment with, receiving feedback on the constraints and possibilities of the system.

3.2.13 Milestone #10 (Aug 12th)

- Complete documentation and demo videos of the instrument design example
- Document the acoustic sensing C++ code.

3.2.14 Final YouTube video (Aug 19th)

Submit final project video, submit final work to GSoC site and complete final mentor evaluation

3.2.15 Final Submission (Aug 24nd)

Important: August 19 - 26 - 18:00 UTC: Final week: GSoC contributors submit their final work product and their final mentor evaluation (standard coding period)

August 26 - September 2 - 18:00 UTC: Mentors submit final GSoC contributor evaluations (standard coding period)

3.2.16 Initial results (September 3)

Important: September 3 - November 4: GSoC contributors with extended timelines continue coding

November 4 - 18:00 UTC: Final date for all GSoC contributors to submit their final work product and final evaluation

November 11 - 18:00 UTC: Final date for mentors to submit evaluations for GSoC contributor projects with extended deadline

Chapter 4

Experience and approach

I have previous experience with the Bela platform through using it in my research, which is focussed upon bidirectional haptic interaction with digital musical instrument designs. Alongside this, I have previous written embedded software for commercial audio products using C/C++ for baremetal, FreeRTOS, and embedded linux systems. This software, for professional audio mixing consoles, involved low level control of FPGAs, interfacing with and controlling external components, and audio processing. This included collaborating with others on larger projects, using source control and project management tools to allow efficient collaboration. Though I have not contributed to larger open source projects – that is experience that I would like to gain through undertaking the GSoC project – I have several personal open source projects on my GitHub profile including audio processing and MIDI control projects.

While I have less formal experience with implementing embedded machine learning algorithms, I believe with my existing skills in embedded software development combined with the existing documentation and knowledge base for PyBela and DiffLogic will enable me to fill any gaps in my knowledge and successfully complete the project.

I believe that the scope of the project, and the included timeline, is an ambitious yet manageable amount of work to achieve within the scope of a medium (175hr) GSoC project. While, like any development project, there are unknowns and areas of additional risk, I believe that the proposal outlines sufficient contingency plans and alternative solutions to enable a successful outcome with benefits to the open source community.

4.1 Contingency

Though I am confident that my software programming skills will enable me to tackle many of the issues faced in this project, there are also many resources to turn to as and when I become stuck on a particular problem. Specifically for this project, the following resources are of particular significance:

- The Bela Forum for issues relating to the Bela cape's hardware, the Bela IDE and Bela's Xenomai realtime environment.
- The BeagleBoard Forum for issues that apply to the Beaglebone Black more generally – such as PRU programming.
- Peers within the Augmented Instruments Lab for additional help with Bela, along with additional machine learning expertise.
- Texas Instruments documentation such as the AM335X Technical Reference Manual for architecture-specific information e.g. PRU communication.

Though the overall order of the project timeline is important, the order of inidividual aspects of the timeline can be flexible if there are no, or few, dependencies upon prior work. In these cases, and where other sources of help have not led to a breakthrough, having some flexbility to swap a smaller task order around until I am able to discuss the blocking issue with my mentor would be a possible approach. This would enable me to continue with work that contributes to the overall project, without sinking too much time into a problem without at least having a discussion about it first.

There are also several areas of the project that warrant more specific contingency details, to de-risk these particular elements further. One such example is enabling ultrasonic audio I/O on the Bela platform. To achieve *ultrasonic* acoustic sensing, a higher sampling rate than Bela's default 44.1kHz is required. Though it may be possible to modify Bela's PRU code to increase the sample rate possible for the audio I/O (this would also likely require hardware modifications to adjust the anti-aliasing filter), an alternative would be to utilise the aanalog I/O instead, which can be run at 88.2kHz.

4.2 Benefit

While acoustic sensing has been explored, and shown to be effective, in previous literature [2, 3], it is a technique that has not been utilised within the New Interfaces for Musical Expression (NIME) community. Additionally, previous implementations of acoustic sensing have relied upon laptops or mobile phones to process the sensing data. Using the Bela plaform with the Beaglebone Black will introduce an embeddable method of acoustic sensing.

The project output of a new open source DMI using acoustic sensing will provide a concrete example (and open source, modifiable code) of these technologies for other Bela users wanting to explore similar interaction techniques, however the most significant contribution to the open source community will be the pipelines and software infrastrcuture created during this project. This will include the following:

- A pipeline for creating, training, and deploying differentiable logic gate networks will be created, building upon work from [4].
- C++ code for swept sine acoustic sensing on Bela, incorporating a configurable sine sweep generator, input processing (FFT, etc.), and a differentiable logic gate network instance, along with a simple to use training setup.
- Documentation and setup guides for both elements, along with the example DMI project serving as a reference implementation of both.

4.3 Misc

The link to the pull request for cross compiling is here.

4.4 References

1. J. Armitage, T. Magnusson, and A. McPherson. Studying Subtle and Detailed Digital Lutherie: Motivational Contexts and Technical Needs. In NIME'23, Mexico City, Mexico, May 2023.
2. G. Laput, E. Brockmeyer, S. E. Hudson, and C. Harrison. Acoustruments: Passive, Acoustically-Driven, Interactive Controls for Handheld Devices. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pages 2161–2170, Seoul Republic of Korea, Apr. 2015. ACM.
3. M. Ono, B. Shizuki, and J. Tanaka. Touch & activate: Adding interactivity to existing objects using active acoustic sensing. In Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology, pages 31–40, St. Andrews Scotland, United Kingdom, Oct. 2013. ACM.
4. T. Pelinski, R. Diaz, A. L. B. Temprano, and A. McPherson. Pipeline for recording datasets and running neural networks on the Bela embedded hardware platform. In NIME'23, Mexico City, Mexico, June 2023.
5. F. Petersen, C. Borgelt, H. Kuehne, and O. Deussen. Deep Differentiable Logic Gate Networks. In Advances in Neural Information Processing Systems 35, New Orleans, LA, USA, 2022.
6. C. Cai, R. Zheng, and J. Luo. Ubiquitous Acoustic Sensing on Commodity IoT Devices: A Survey. IEEE Communications Surveys & Tutorials, 24(1):432–454, 2022.

7. A. P. McPherson, R. H. Jack, and G. Moro. Action-Sound Latency: Are Our Tools Fast Enough? In NIME'16, Brisbane, Australia, July 2016.