Mini-Projects Guidelines

Tuesday, September 17, 2019

The mini-projects have an emphasis on individual acquisition of new knowledge and practical aspects related to audio signal processing. The projects should be done in groups of two. Students may choose a topic from the list below or propose their own project. The topic does not have to be part of the lecture; it only has to be in the broader area of audio signal processing.

The evaluation of the mini-projects will be based on a report, the accompanying code and documentation, and a short demonstration of the practical results. More on the deliverables can be read below.

Deadlines

- ***Friday, October 4, 2019:*** Project preferences (top 3) sent by email to mihailo.kolundzija@epfl.ch. If proposing a project different from the list below, submit a title and single paragraph description.
- ***October – November 2019:*** Meet at least once a month with the assistant to report progress. The Tuesday exercise sessions (12h–14h) will mainly be devoted to theoretical exercises, but projects can also be discussed with the assistant and lecturers. The Wednesday exercise session (9h–10h) is devoted to projects.
- ***Tuesday, December 17, 2019:*** Submit project and demo day!

Deliverables

- ***Report:*** this can take the form of a 5-10 page PDF document (excluding source code and figures), a webpage, or Jupyter Notebook. *All figures taken from a book or from the web must be clearly referenced.* Below is a recommended structure:

  1. Title
  2. Abstract
  3. Introduction
  4. Review of existing techniques
  5. Algorithm description
  6. Results and discussion
  7. Conclusion
  8. Bibliography

- ***Source code:*** ZIP file submitted on Moodle OR link to repository on c4science / GitHub.[1]

  Be sure to include all necessary audio files and a `README` file on how to run your algorithm. Ideally it's a single call, with well-explained arguments. If we cannot easily run your algorithm, we will not evaluate your implementation.

[1] You should be able to make an account with free private repositories on c4science with your EPFL credentials. Alternatively, you can sign up (for free) for the GitHub student pack for free private repositories on GitHub and other tools.

Project proposals

Project 1. Modeling of Room Impulse Responses
In this project, you are asked to model Room Impulse Responses (RIR). RIRs depend on the position of the sources and microphones, and on the characteristics of the room (e.g. size, absorption coefficients of the walls). Compare your implementation to that in `pyroomacoustics` and/or other free RIR generators you may find online. The simulator on `pyroomacoustics` relies only on the image source method. We would suggest investigating another method, such as ray tracing, and/or techniques to add naturally sounding reverberation. You can read about reverberation generation here and here.

Project 2. Binaural Room Impulse Responses Simulating human hearing sys.
Implement an algorithm that calculates binaural room impulse responses, given the size of the room, the positions of the source and of the head. You should use head-related transfer functions (HRTFs) to simulate how a plane wave arriving at the head is sensed by the two ears. You can either adapt a room acoustics approach, simulating many reflections, or simply calculate the main early reflections and add a stochastic reverb. Fine-tune your algorithm to obtain results that are as good as possible. You can find an open dataset of HRTFs here and use `pyroomacoustics` for simulating room impulse responses.

Project 3. Directinal sources and receivers for `pyroomacoustics`
One typically models sound sources and microphones in room acoustics simulation softwares as omnidirectional. This however is not very realistic and flexible. The goal of this project is to allow the source or microphone to be described by their directional response, be it in the spherical harmonic domain or in terms of spherical distribution of attenuation. Extra points for providing support for frequency-dependent directional characteristics. We highly encourage adding this feature to the existing library `pyroomacoustics`.

Project 4. Dual-Microphone Noise Removal for Keyword Spotting
The goal of this project is to apply an adaptive noise cancellation filter to a pair of microphones such that the a wake-word uttered by the user, such as "Alexa" or "Hey Google", appears crisper and more intelligible. The noise source could be any background noise, but also music and dialog coming from an audio system, radio or TV. You can use the this paper as a good reference. You can get a wake-word dataset from here.

Project 5. Pitch Detection
An important parameter of a music signal is its pitch or fundamental frequency. In this project, you will investigate methods to determine the pitch of incoming signals, mostly music. The analysis will be applied on monophonic signals. Try using your approach to implement a "lock" that can only be opened if you whistle the correct *relative* combination of frequencies.

Project 6. Beat detection
Implement an algorithm which detects sound pressure maxima (beats) occurring at regular intervals in time. Analyze the audio signal in a causal manner (looking only at present and past samples) and predict the time instants when beats are expected to occur. Experiment with the parameters of the algorithm to make it work with as wide a range of music signals as possible.

Project 7. Noise reduction / speech enhancement
The goal of this project is to remove the noise from a noisy sound sample. The sample should be analysed and modified in short-time Fourier domain (and possibly also in time-domain) in order to remove as much of the signal components which are noise as possible while keeping as much of the desired signal as possible. Here are some resources to a few denoising approaches: spectral subtraction (already in `pyroomacoustics`), all-pole modeling, subspace approach, and neural network approach. Your results could serve as a *Pull Request* for this sub-module in `pyroomacoustics`!

Project 8. Blind Source Separation / Cocktail Party Problem
Blind source separation (BSS) attemps to separate a set of source signals from a single microphone recording

or a set of mixed signals, with limited information about the source signals or the mixing process. Currently, there are two implementations in `pyroomacoustics` that use multiple recordings: AuxIVA and Trinicon. You can find a general overview BSS here and here and some reference implementations at `scikit-learn` and `pyroomacoustics`. Your results could serve as a *Pull Request* for the BSS sub-module in `pyroomacoustics`!

Project 9. Pitch Shifting Voice Effects
The goal of this project is to apply voice effects in real-time. This can be done either on a Nucleo STM32 board in C (as in these exercises) or with your laptop in Python using the `sounddevice` library. In this Jupyter Notebook, you can see various implementations of voice transformations in a non-real time fashion. We ask you to implement the following effects from the above notebook in real-time:

- Robot voice.
- Pitch shift via Granular Synthesis.
- DFT-based pitch shift.

Project 10. Keyword Classifier
Use machine learning to train a keyword classifier. You can read about a simple classifier in this tutorial. Use the Speech Commands Dataset and augment it with `pyroomacoustics` to train a more robust classifier. You can also find external datasets / noise files from the web, such as at `https://freesound.org/`. We recommend using the `keras` library if you are new to TensorFlow. Try to turn your classifier into a real-time keyword detector using `sounddevice`.

Project 11. Acoustic Echo Cancellation
Implement a basic acoustic echo canceler and compare different algorithms (e.g. LMS, NLMS, and RLS). What happens during double-talk? Also listen to the resulting signals. You can find some reference implementations on pyroomacoustics.

Project 12. Chorus effect
Implement a chorus effect that takes an input signal and outputs the sum of several copies of the input signal, each with a differently varying pitch. The main challenge with this project is to do the interpolation of the sound signals correctly so that no annoying artifacts are perceived.

Useful development tools

- ***MATLAB:*** EPFL provides all students with a license of MATLAB. See here for more information. Octave is a free alternative to MATLAB with a similar syntax and IDE.

  Don't forget to add appropriate comments to your code, in particular describing the input and outputs of functions that you create.

- ***Python:*** A great, free alternative that will make your project accessible to a wider audience! Some useful tools for Python are Anaconda (for creating environments and installing/managing packages), `pip` (for installing packages), and PyCharm (as an IDE).[2] If you use Python, we recommend sticking to Python 3.

  For "clean" code, you can find a recommended *Python style guide* here.

  We also recommend using doctrings to document functions and modules. For example:

```
    def function(x, y):
    """
    This function does something.

    Parameters
    ----------
    x: data type
    Description.
    x: data type
    Description.

    Returns
    -------
    data type
    Description.
    """

    return x+y

```

  Useful packages in Python: `numpy`, `matplotlib`, `scipy`, `sounddevice`, `pyroomacoustics`, `keras`, `tensorflow`, `scikit-learn`, `pandas`, `python_speech_features`, `argparse`

- ***Git:*** Version control system to collaborate on a project / code. We've already mentioned two hosting services: c4science and GitHub. You can a desktop tool, such as GitHub Desktop or GitKraken, or the Terminal. Below are frequent commands in one's workflow:

```
$ git status                    # List all new or modified files
$ git add <file>                # Prepare file for versioning
$ git commit -m "<message>"     # Record added files to version history
$ git push                      # Upload all local branch commits
$ git pull                      # Download and incorporate changes
$ git checkout -b <branch_name> # create new branch called "branch_name" and switch to it
$ git checkout <branch_name>    # switch to "branch_name"
```

  You can find a nice overview of Git here.

- ***Documentation:*** LaTeX is a popular choice for documents in the research community. You are free to use your desired software but be sure to export the final report as a PDF!

  As any practical experience, it is worth the extra effort to showcase your work so that you can build a strong portfolio! Do consider a simple webpage (for example, with GitHub Pages) or a Jupyter Notebook for presenting your work.[3]

Additional resources / links for project ideas

- `pyroomacoustics`: demo notebook, documentation, GitHub repo (for up-to-date code and changes)
- Kaldi: open-source speech recognition toolkit.

[2] With your EPFL account you can get the Pro version for free.
[3] When published, a Jupyter Notebook can be nicely rendered on nbviewer.

- Keras: high-level API for quick experimentation / prototyping with neural networks.
- Sources for audio files: 48-channel anechoic audio recordings for 3D sources (good for testing array processing algorithms like direction-of-arrival, beamforming, source separation), Speech Commands Dataset (recordings of common commands like “yes”, “no”, “stop”, “go”), miscellaneous audio files (good for noise files), Alexa dataset (for training a Wake Word detector), CMU Arctic (clean recordings of male/female speech with different accents), additional datasets, and search engine for dataset.