EECS 207: Digital Image Processing
Spring 2025
Project Part 1
Spectrogram Optimization Revisited
Due 11:59 pm Thursday, May 1 via CatCourses as .pdf file
(version 4/25/25 3pm)

Goal

Lab 2 had you investigate certain design choices to see what settings produce the best spectrograms for bird calls from five different species. That evaluation was done **qualitatively** by visualizing the resulting spectrograms as images. In Part 1 of your Project, you will revisit spectrogram optimization in the following context:

- You will evaluate the spectrograms **quantitatively** using a content-based image retrieval (CBIR) framework in which you will try maximize average precision.
- You will evaluate the performance using a larger dataset consisting of 218 calls from five bird species.

Similar to Lab 2, you will investigate spectrograms computed with different widow lengths, amounts of window overlap, and window shape. It will be interesting to see whether your qualitative findings from Lab 2 agree with your quantitative evaluation in this assignment.

You will additionally have the chance to investigate some other design choices when computing the spectrograms.

To Do:

1. Make sure you are able to run the Jupyter Notebooks from Project Part 0 to create the precision recall curves and compute average precision.

2. Fix the type of images features used to perform the CBIR. A reasonable choice would be Gabor texture features with 3 scales and 4 orientations. Fixing the image features allows you to focus on the effects of changing the spectrogram computation.

3. Produce precision recall curves and average precision values corresponding to spectrograms computed using:

    - Different window sizes.
    - Different amounts of window overlap.
    - Different types (shapes) of windows.

4. Investigate some other design choices available in `signal.spectrogram`. Here is the calling syntax for the function:

```
spectrogram(x, fs=1.0, window=('tukey', 0.25), nperseg=None,
noverlap=None, nfft=None, detrend='constant',
return_onesided=True, scaling='density', axis=-1, mode='psd')
```

In particular, investigate:

- Setting *scaling* to *'density'* or *'spectrum'.* The code in Project Part 0 uses the default which is .*'density'.*
- Setting *mode* to *'psd'* or *'magnitude'.* The code in Project Part 0 uses the default which is .*'psd'.*

5. It would take a lot of effort to investigate all combinations of the design choices above. So, instead you can just focus on varying one choice while keeping the others fixed. More on this below.

6. From the little experimentation I've done, I haven't noticed large variations in the average precision values based on these design choices. But, I think the variations are significant in terms of improving classifier performance in a bird call detection system.

To Submit:

A single PDF with a 1) precision recall curve, 2) a table of design choices and average precision values, and 3) a brief takeaway summarizing your findings for each of the five experiments below:

- Varying the window length keeping the **percentage** of overlap the same. Three different settings at a minimum.
- Varying the amount of window overlap. Three different settings at a minimum.
- Varying the window type (shape). Three different settings at a minimum.
- Setting *scaling* to *'density'* or *'spectrum'.*
- Setting *mode* to *'psd'* or *'magnitude'.*

That is, your PDF needs to at a minimum have five triplets of 1) precision recall curves, 2) tables of design choices and average precision values, and 3) brief takeaways summarizing your findings for a particular experiment.

See the file "EECS_207_Project_Part_1_sample_report_Spring_2025" as an example.

While the above is what you need to do/submit at a minimum, please feel free to experiment more. One final experiment of particular interest is to pick the best design choice from each of the five experiments and see if this results in the maximum performance. Or perhaps there's an interdependency between window size and window type (shape). That is, the best window size for a Tukey window might not be the best for a Hann window. Again, I'm having you perform this investigation since I would like to find out what the best spectrogram design choices are for bird calls.

Notes:

1. Make sure that the intermediary files in the CBIR framework are being produced for the correct spectrograms. That is, don't make the mistake, for example, of computing new spectrograms but then still using the image features from previously computed spectrograms. Make sure a set of these intermediary files are being computed for each variation of spectrogram and that these files are being read by the next Notebook in the processing pipeline.

    - The *.png spectrogram and *_info.txt spectrogram metadata files created by Compute_Spectrograms.ipynb in the folder \Spectrograms.

    - The *.png individual bird call images created by Compute_ROIs.ipynb in the folders \ROIs\<bird>\ROIs.

    - The *.txt feature files created by Compute_Gabor_texture_features.ipynb in the folder \features. Also, make sure that during feature extraction (running Compute_Gabor_texture_features.ipynb), the correct set of bird call*.png files are copied from \ROIs\<bird>\ROIs to the folder \images before the features are extracted.

    - The *.txt precision recall files created by Perform_query.ipynb in the folder \precision_recall.

    You are free to implement this however you want. One brute force method would be to create new versions of all the folders above for the intermediary files. Note that I might have missed some the intermediary files in the bulleted list above.

    If you change your spectrogram computation but end up with the same precision recall curve and same average precision value, this is potentially an indication that you are not using the correct intermediary files.