Frequency analysis here:

Terminology: Decaying Instruments vs Non-decaying Instruments

This project’s methodologies assume that the sound inputs are recordings of the **decaying instruments**. Decaying instruments include families of chordophones (i.e. Guitar, harps, piano, etc.) and percussion (i.e. xylophone) instruments. These instruments have highest sound intensity at the time of flicking strings/striking keys of the instrument, and the sound intensity exponentially decays over time.

On the other hand, the common **non-decaying instruments** include woodwind instruments (i.e. Saxophone, flute, clarinet, etc.). When played with constant breath, these instruments’ sound intensities are somewhat equal over time.

The Figure 1 below shows a Matlab plot of the note A4 from a digital piano (Decaying instrument). As one can observe, it has the highest peaks at the beginning of the note and the peaks decrease exponentially over time.

For those interested, code for plotting the Figure 1 is as follow:

Throughout the post, I’ll try my best to include the code samples that can replicate these plots.

Now that we agree on the expected behavior of the input, let’s remind ourselves of the objective of the project: we are to detect note signatures of the sound recording of a decaying instrument. This means we expect the recording inputs to have more than one instance of note. To be able to detect individual notes, we must first separate each note from one another.

Let’s start from the base case: recording with single note.

Consider Figure 2, which shows magnitude plot of the sound sample shown before in Figure 1. As humans, we can easily detect the starting point of the note – Our eyes can easily detect the sudden amplitude jump at the marked location.

Can we use the same approach in the algorithm?

Detecting maximums of the

We can detect sudden amplitude jumps by finding local maximums in the plot. However, even though the plot in the Figure 2 might look like a bar graph, sound has a wave-like property and the amplitude oscillates very quickly over time. One can see this more clearly by zooming into the plot. Figure 3 shows the area where I have zoomed in, and Figure 4 shows the zoomed-in plot.

As it is shown, the plot is a high frequency sinusoid. If we were to find local maximums, we would be ending up with many false data points. In the figure, I have labelled few of the peaks that would falsely detected as starting point of the note.

It is apparent that we need to simplify the data we are given with. If we can somehow average the peaks of the sinusoid and find a smooth curve that ‘envelopes’ the original plot, it can be used to detect overall behavior of the original plot. In Figure 5, example of such envelope is drawn in red.

Finding envelope – Gaussian Filter

We can use a Gaussian Filter to find the envelope. By convolving a Gaussian Curve to our plot, we can filter out the high frequency component of the plot. Although details of convolution operation will not be covered in this post, following points should be enough to understand this operation:

* Convolution of two plots f(x) and g(x) in time-domain is equal to multiplication of the two in the frequency domain.
* Gaussian Curve, which has a bell-curve shape in time-domain, holds its shape in frequency domain. However, the standard deviation of bell-curve in frequency domain is inverse of standard deviation of the bell-curve in the time-domain

This means that if we want the envelope to keep overall shape of the plot, we must keep the low frequency components of the plot. And we can do so by convolving a Gaussian Curve with large standard deviation to the plot.

<Click to expand if you’re confused about why we must convolve a Gaussian curve with large standard deviation to the original plot>

Gaussian curve I used is shown in Figure 6. The size of the curve was determined from trial and error that forms the best envelope.

Red curve in Figure 7 shows the computed envelope when the Gaussian curve is convoluted to the signal. After down sampling process, we can now find the starting point of the note by locating local maximums.

The same procedure can be applied to recording input with multiple instances of note. Figure 8 to 10 shows the algorithm being applied to multiple instances of note.

Now that we’re able to find out the starting points of each notes, we will now find the ending points, in order to extract individual notes from the recording.

Finding ending point of the note

For ending points, we can simply take the midpoint of consecutive starting points. Figure 11 shows the overall process of finding duration of the first note.  
For the ending point of the last note, the algorithm will take the last

Sound, along with the other four senses of human, plays important role of transferring data from one to another. As a person, and as a musician who enjoys music, I consider the field of audio attractive.

Sound in engineering has many applications. By transcribing an audio recording to a computer-recognizable data, one can further process the data with various computer aided tools.

For example, Shazam, one of the main service provider of music identification tool, has announced that its monthly active user count of their service has surpassed over 100 million. Shazam also recently formed a partnership with Apple, and now offers their service through Siri. (Try asking Siri: “What song is this?”)

In this posting, I will be presenting the project I did in ECE462 – Sensory Communications course at University of Toronto. You can expect to learn the note separation (extracting individual instance of notes in the input recording) I used in the project, and how my team analyzed the extracted notes in frequency domain to determine their