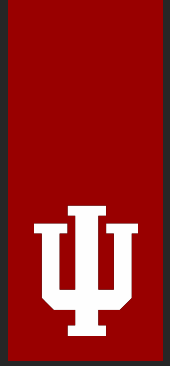


# Monophonic Audio Segmentation and Resynthesis

CSCI-B 557 Final Project

Stephen Karukas

INDIANA UNIVERSITY BLOOMINGTON

# Problem Description

Given:

- *F(X): A* recording of a monophonic instrument *F* playing a piece of music *X*.
- *Y*: A passage of music given as a “score” (symbolic representation)

Synthesize a recording *F’(Y)* that sounds convincingly like instrument *F* playing passage *Y*. Abstractly, the problem is to approximate an acoustic instrument *F* by a digital instrument *F’*.

Right now my solution works offline, so both the recording *F(X)* the score *Y* are provided at the beginning of the algorithm.

# Motivation

- For digitally synthesized music, instrument sound synthesis is usually done by professionally recording isolated sustained tones or using finely-tuned synthesis models. These libraries are often very expensive.
- Simple sinusoidal models like those we've seen may not do a very good job replicating the formants of instruments, how their timbres change across their range, or how timbre changes over time (vibrato).
- Using recordings from performances has the potential to include more "human" aspects, such as instrument-specific transition sounds.

# Specifics

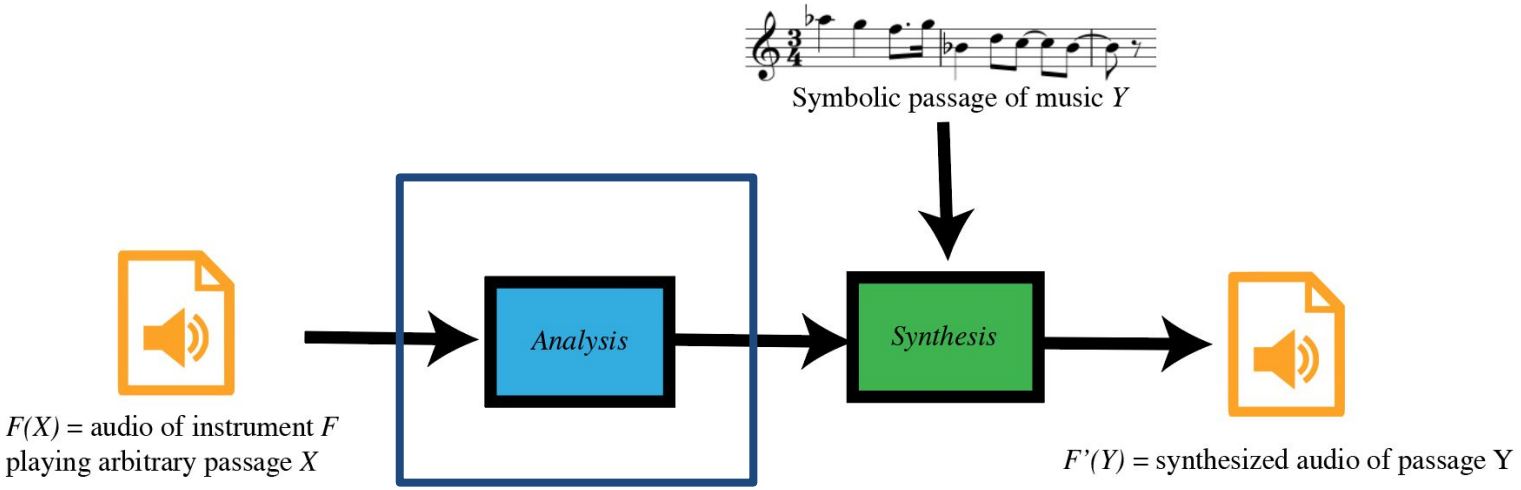
- For this project, I only used recordings that would be described as “sustained” (no short notes).
- The instruments include clarinet, violin, and flute, with varying levels of vibrato and expressivity in dynamic range.

Flute - Bach partita in A minor, Sarabande

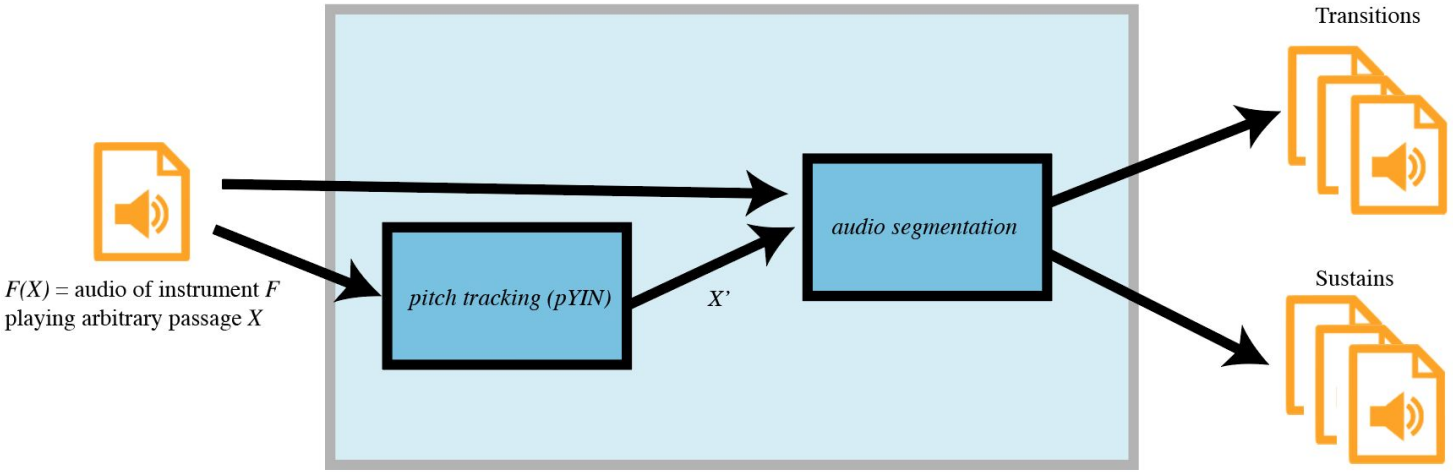
Violin - John Williams, *Jewish Town (Krakow Ghetto, Winter '41)* from Schindler’s List

The audio recordings were taken from public domain sites and personal files.

# Overall Problem Workflow

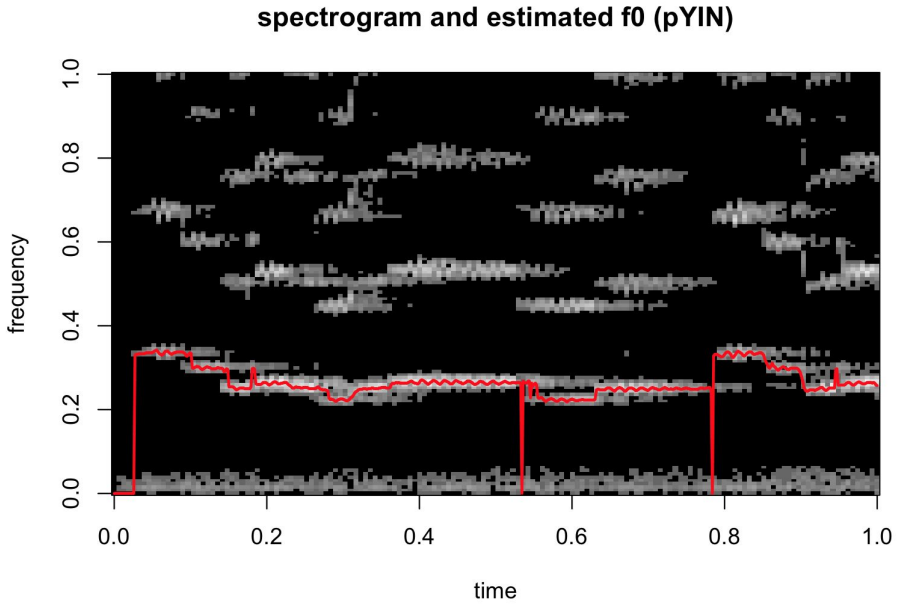


# Analysis Workflow



# pYIN [1]

- A popular algorithm for calculating fundamental frequency (f0) of audio.
- I used the implementation from the Python librosa package [2].
- It proved very reliable for pitch tracking in this project

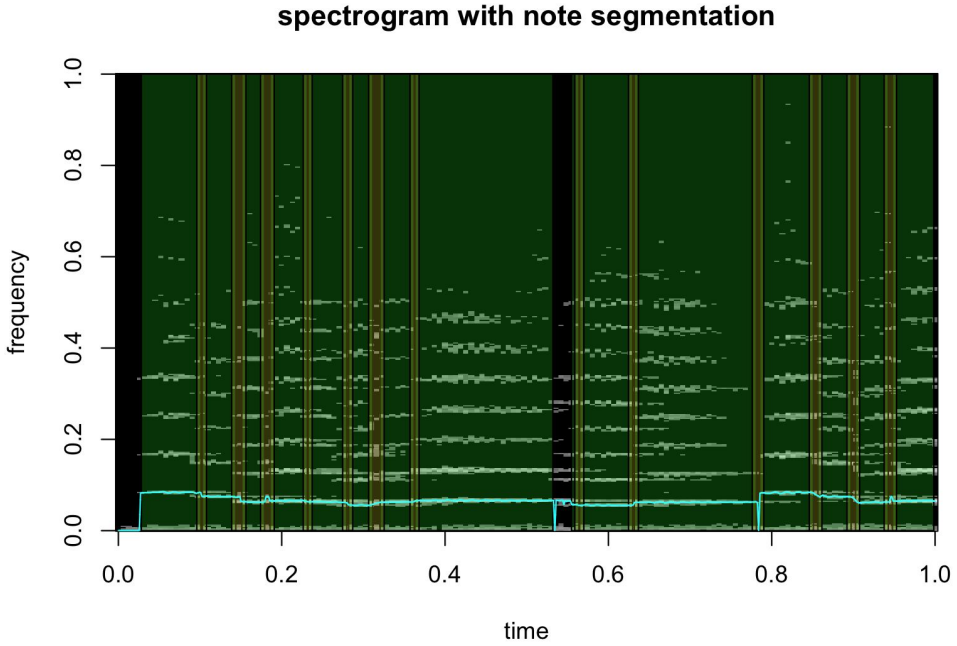



Recording: *Schindler's List* on violin

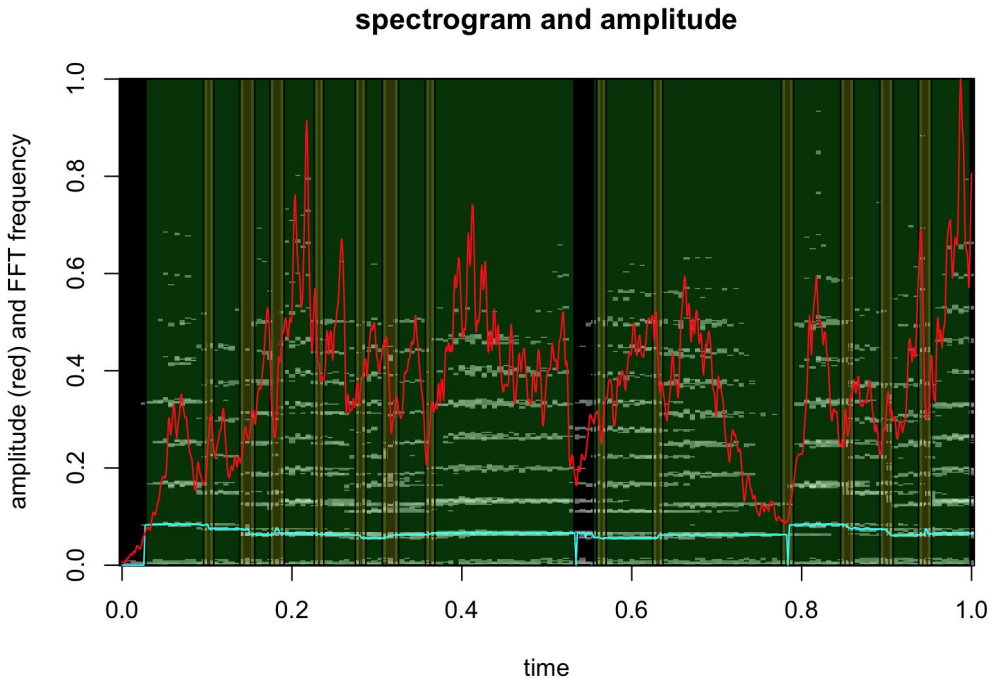
[1] "PYIN: A fundamental frequency estimator using probabilistic threshold distributions" Mauch, Dixon 2014
[2] https://librosa.org/doc/main/index.html

# Audio Segmentation

- Find times in the audio where notes begin, end, and transition.
- A few possible metrics to segment by:
  - amplitude
  - spectral flux
  - rate of change of pitch
  - deviation from a set pitch

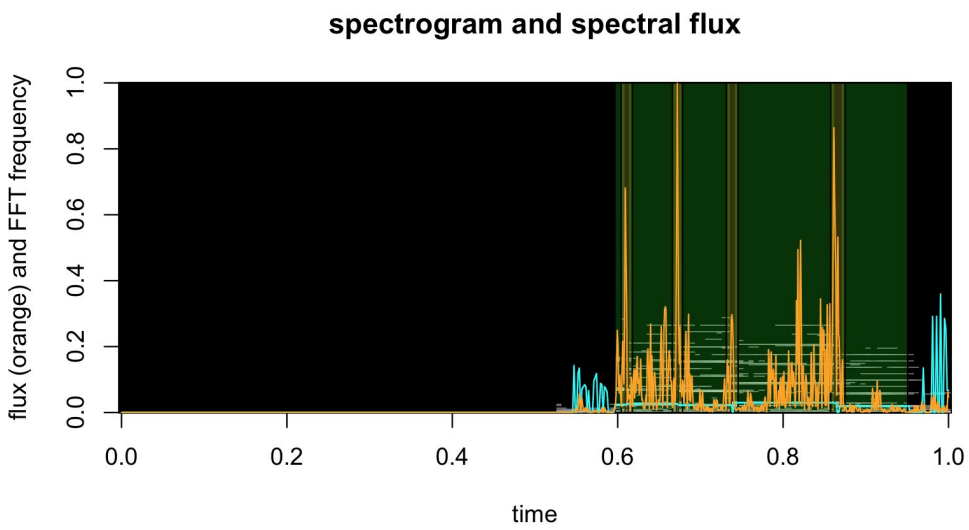


# Amplitude

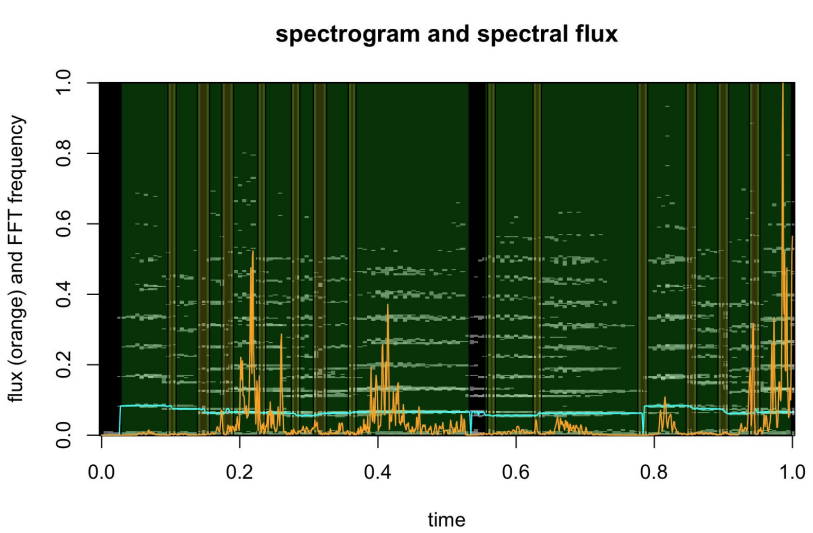

spectrogram and amplitude
amplitude (red) and FFT frequency
0.0
0.2
0.4
0.6
0.8
1.0
time

# Spectral Flux (distance measure between consecutive STFT frames)

clarinet

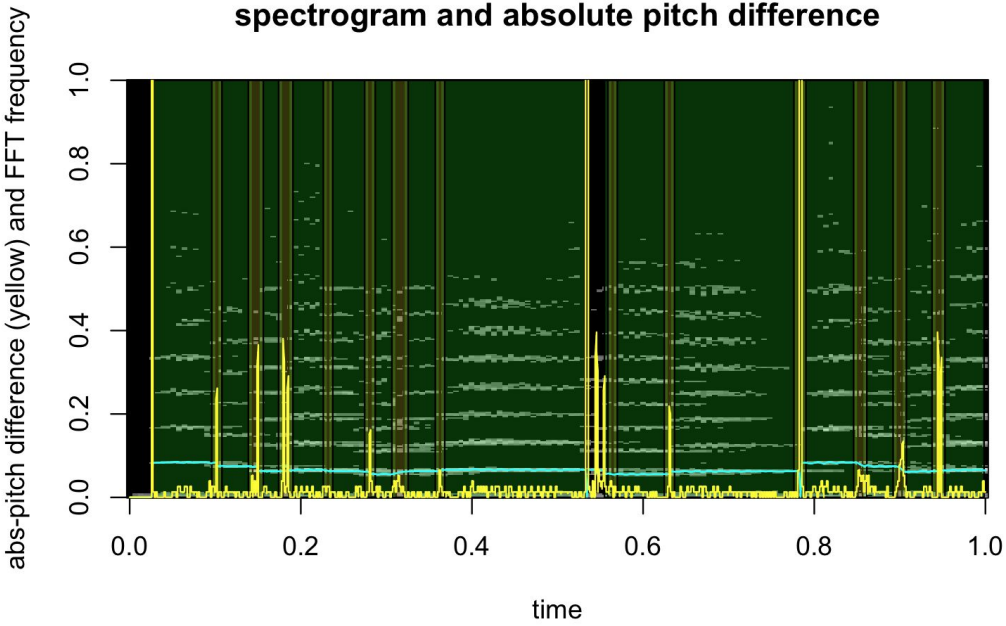



violin



# Rate of change of pitch

- Assumption: transitions between notes will have a large change in pitch
- Seems to work well, and pitch was already calculated by pYIN
- Would need a tuned threshold parameter, and it doesn't do well with slow glisses

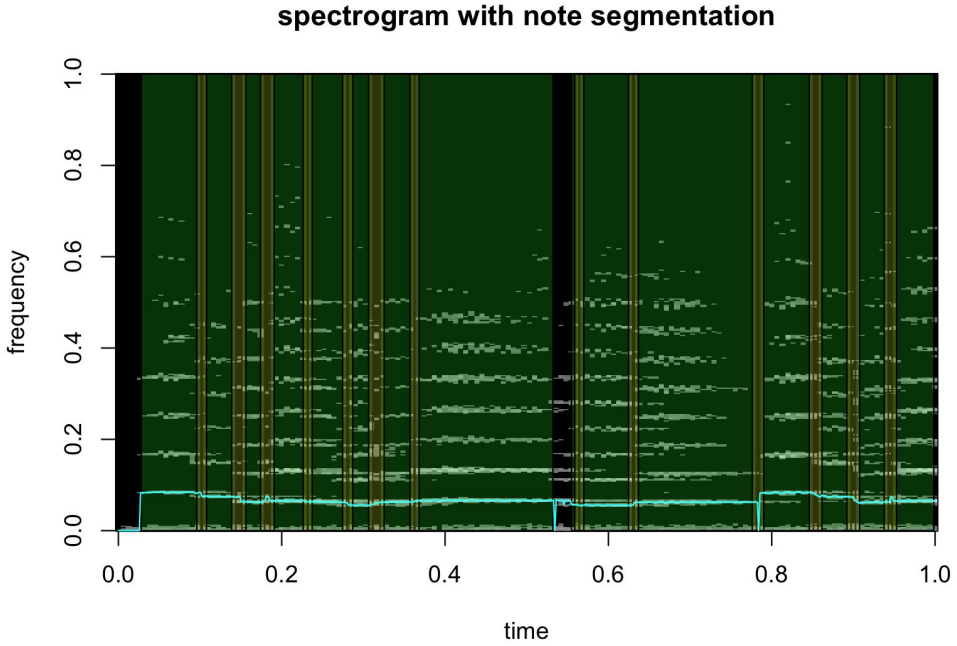


# Greedy pitch tracking

- It's easy to look at the f0 sequence and tell where the notes are (flat areas)
- A "new note" could be considered one that is more than *e* semitones away from the current one, where a good value for *e* is 0.5

Algorithm:

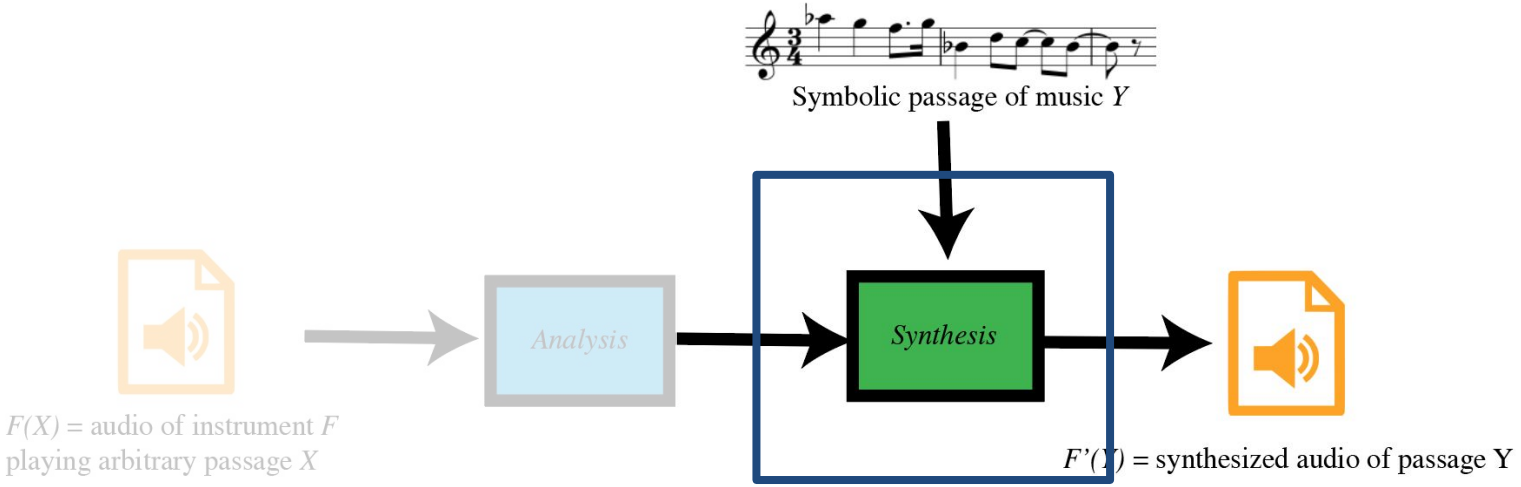
1. Begin with a small range **r**
2. *m = mean(pitch(***r***))*
3. While |*pitch(i)-m*| < *e*:
   add the next index *i* to **r**.
4. Record **r** and repeat with the section directly after **r**.



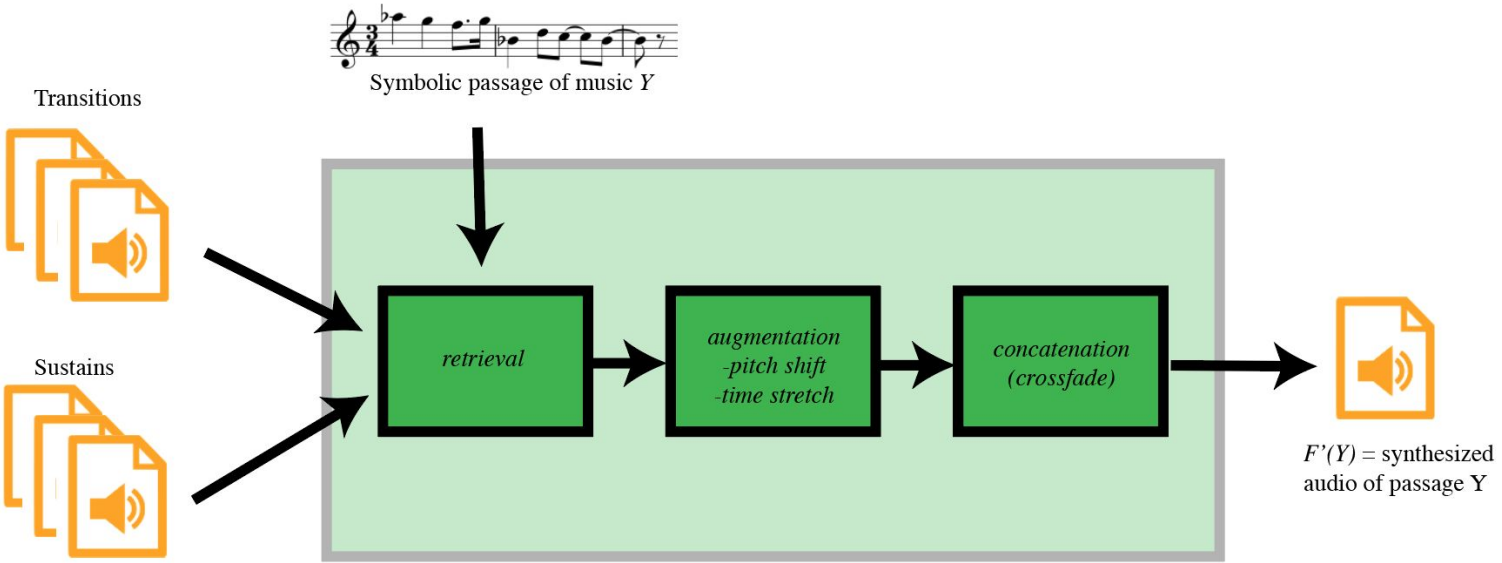
# Output from the analysis

- The segmented notes are extracted and stored as short (0.5+ s) WAV files, stored by the exact mean pitch within the segment.

# Overall Problem Workflow



# Synthesis Workflow



# Retrieval

- Retrieval of a sustained note is done by looking for a note in a similar range (assumed to be of a similar timbre).
- Retrieval of a transition is done by looking for a transition spanning a nearby interval, and within a nearby range.

# Augmentation/Concatenation

- Time stretch:

  STFT => ΔPhase => resample STFT => ∑Phase => $STFT^{-1}$

- Pitch shift:

  Time stretch => resample

- Equal-Power Crossfade (sum of squares is constant):

  For each pair of notes, I used equal-power crossfading, blending the first sustained note into the transition **z** by overlap |**z**|/2, then crossfading that result with the second sustained note, same overlap.

$$\text{crossfade}(x, y, t) = x_t\sqrt{1-t} + y_t\sqrt{t}$$ where *x* and *y* are the overlapping portions of two signals

# Results

*Amazing Grace* synthesized from clarinet recording

*Amazing Grace* synthesized from flute recording

C Major Scale synthesized from violin recording, **no transitions**

C Major Scale synthesized from violin recording, **with transitions**

# Challenges

- Relying on the pitch tracking for segmentation means portions will occasionally start or end with reverberations of the previous note.
- Time-stretching can sound very inauthentic (though pitch shifting tends to be effective)
- Noise and amplitude variations between and within notes
- Multiple simultaneous notes (polyphony or reverb)
- Very limited data, have to augment it
- In its current form, all transition intervals must appear within the input recording.

# Other possible approaches

Automated splicing of audio may be a good-sized final project, but it can only go so far. Here are some other possibilities I hope to explore in the future:

- **Generative Model:** instead of a database, learn a generative timbral model through analysis of the STFT as a sequence.
- **More Synthesis/Analysis Parameters:** One of the issues coming out of this project was the discontinuity of “line”, or varying intensity, between notes. If segments were analyzed further, there could be finer control over the more “human” aspects of the synthesized sound.
- **Other Improvements:** Because this is a “pipeline” sort of process, parts could potentially be switched out or collapsed. For example, it may make more sense to segment the audio while performing pitch extraction.

# Questions

INDIANA UNIVERSITY BLOOMINGTON