Hearing Sheet Music: [Still no Clever Backronym]

Stephen Miller
Stanford University
450 Serra Mall Stanford, CA 94305
sdmiller@stanford.edu

1. Introduction

At the moment, I know nothing Schopenhauer's philosophy. Nor, as far as I can tell, do any of my friends. In a world without written language, we'd be at an impasse. If I were serious about learning it, I'd need to go to the Philosophy department, schedule a meeting with a professor, and ask for an explanation. Fortunately, we don't live in that world. I can pick up a copy of The World as Will and Representation and get a rough idea of the concepts. No one spoke: the writing silently communicated everything.

Music, like speech, can be written. But to many of us--particularly amateurs--written notes don't directly convey music in the same way that written words convey ideas. Instead, we need to first sit in front of our instrument of choice and play it note by note, mechanically, listening as we play. Eventually, after some awkward stumbling, there's a moment where the individual notes become a melody and everything clicks. Once the click happens, each note becomes a necessary part of a logical whole, and the learning process snowballs.

Hearing Sheet Music is a project which tries to provide that click. Rather than sitting at a piano bench and hacking away note by note, you take out your iPhone and point it at the music. A cursor appears at the start of the staff . You hit PLAY and the music starts, highlighting notes in your camera feed as theyre being played in real time. Listen to the whole song a few times through. If a particular measure is giving you trouble, highlight it, cut the tempo in half, and re-play.

2. Problem Statement

The above was a high-level description of an end result, which I expect will take a far more polished User Interface than a few weeks will permit. In the short term, I will limit myself to a particular subtask: single image score comprehension. As input, the algorithm will be given an image containing sheet music. As output, the song (if the entire width of the score is visible) will be played – otherwise, multiple parts will be detected, and a user is given the option of selecting which portion to hear. As this is meant to be run on a phone, it must be computationally inexpensive—if at all possible, I wish to avoid any server-side processing.

2.1. Input

I assume that a score consists of printed black characters on a white background, with potential markup on the surrounding page but minimal noise on the staff itself. I make no assumption that the exact typeface of each note is the same, nor that there must be a single melody line. There may be multiple clefs, as well as non-standard key signatures.

In the input image, the score must be reasonably visible—qualitatively, this means the lines of the staff must be visible to a human observer, as

are the note types and locations. I further assume that, while perspective effects will be present, the sheet itself is well approximated by a rigid plane: the paper will not be significantly deformed, and the assumption that the lines of the staff are straight will be (roughly) kept. The image will not, however, be particularly high resolution: as this is built to work on a camera phone, it must be able to handle the resolution of a standard iPhone camera.

2.2. Output

While the music will be (quite trivially) converted to audio, the information output will be a time signature T, set of n measures ($\mathcal{M}$) and a connectivity graph $C \in \{0,1\}^{n \times n}$ where $C_{i,j} = 1$ indicates that measure j follows directly after measure i. A measure consists of a set of notes, which have a pitch (taking sharps, flats, and the key signature into consideration), start time (in units of beats), and duration (also in units of beats). As many instruments allow for multiple notes at a time, these times may overlap.

2.3. Training

The training data consists of labeled images of the form described above. The lines of the staff have been labeled, and bounding boxes for each note, rest, modifier (flat, sharp, natural), and key/time signature are given.

3. Technical Approach

This problem can be broken into a number of parts: detecting the staff, inferring the perspective transform, classifying individual symbols, and finding a measure which is as consistent as possible with these classifications.

3.1. Staff Detection

A local contrast normalization step is first applied to the image, to compensate for lighting effects (which often vary in intensity greater than the lines of the staff themselves). The image is then thresholded by intensity, to infer foreground vs background. The horizontal lines of the staff and vertical measure bars are found via a probabilistic Hough Transform.

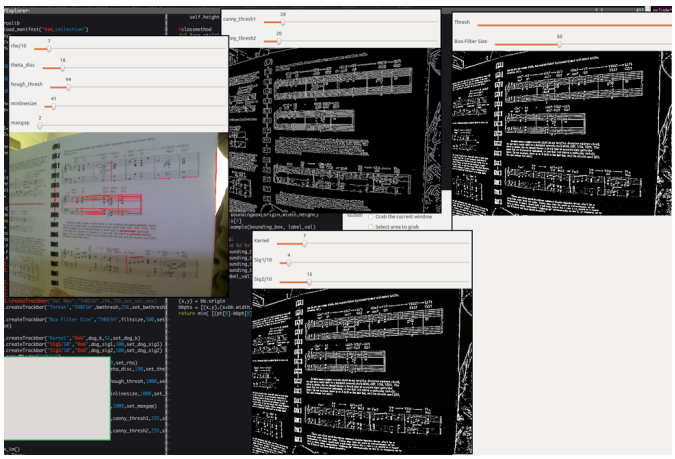

Figure 1. Playing with filters, edge detectors, hough

3.2. Perspective

Once the staff has been located, the perspective can be inferred by assuming each horizontal line of the staff is strictly horizontal in the object frame (likewise for vertical measure bars) and minimizing the reconstruction error. (Note, this may or may not be used: depending on feature invariances, rectifying the image may not be necessary)

3.3. Symbol Detection

I have not fully settled on the approach. Due to the difference in size of various symbols, I will likely train one-against-all SVMs separately for each class. While certain things (rests, key and time signatures, flats, sharps, etc) are extremely consistent and should be very simple to detect, notes post an interesting problem since they may vary greatly in perceived size. Consider, for example, 4 consecutive sixteenth notes. Each individual "note" is identical to a quarter or eighth note: their type is inferred by looking at the stem which connects them. It will likely be simplest, then, to train an SVM to detect only "Quarter or less" vs "Half" note bulbs (the rounded, lower portion). Once these have been detected, the stem can be traced, and the distinction between Quarter, Eight, Sixteenth, etc found by the number of

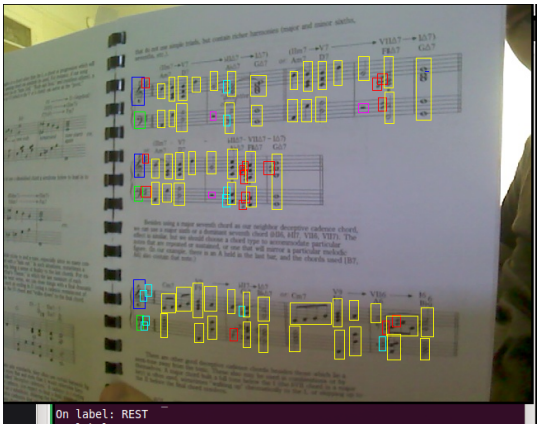

Figure 2. Interactive note labeller (Python/HighGUI), bounding box editor, with file IO

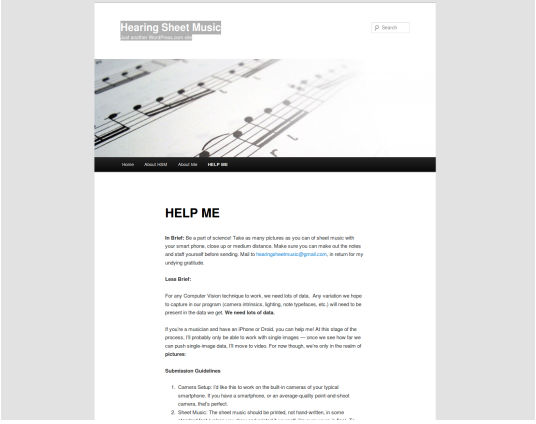

Figure 3. Website for data collection

edges.

3.4. Measure Computation

Given a set of symbols and their rough location, the final step is to pool them together into a consistent measure. This means determining their relative pitch (by which staff line they intersect), absolute pitch (applying modifiers and key signature directions), and correcting their duration (if, for example, the dot of a dotted eight note was not initially detected). This can be turned into a fairly nice optimization problem, where it is required that the sum of all beats in a given measure equals the number of beats per measure given in the time signature.

4. Intermediate Results

4.1. Data Collection

To help gather data and get a variety in typefaces which sheet music found on Google didn't seem to have, I made a website at

`http://hearingsheetmusic.wordpress.com`

posting it on a few social networks and inviting others to submit images (see Fig. 4.1). Submissions are rolling in, and I've done the necessary scripting to label symbol bounding boxes in the image.

4.2. Image Processing

I've played with a number of different ways to preprocess the image to account for the effect of shadows: so far local contrast normalization has worked best for finding hough lines. Other approaches (Difference of Gaussians, doing no preprocessing but running Canny and feeding the resultant edge image into the Hough score detector) have worked reasonably as well. In the interest of running this on a phone which may not contain OpenCV though, I am hoping to limit this to simple filters and simple hand-coded algorithms.

I have not yet trained note detectors, particularly because there were a lot of subtleties I had not considered when only looking at my own, single-melody-line sheet music. Having others submit their own made me reconsider my approach, as certain features of the note (such as the bulb) are critical, while variations in size/stacking/grouping may make my initial "detect every possible note individually" require far too much variety in input data.

4.3. Measure Computation / Audio

I've build the infrastructure and worked out the math to go from an ordered collection of notes (which only know their own type and staff location) to a measure which takes key- and timing-signatures into account. This is then converted to an absolute frequency representation, and output by MIDI.