Hearing Sheet Music: Towards Visual Recognition of Printed Scores

Stephen Miller
554 Salvatierra Walk
Stanford, CA 94305
sdmiller@stanford.edu

Abstract

We consider the task of visual score comprehension. Given an image which primarily consists of printed sheet music, we wish to output the audio. In this work, we first consider the task of unconstrained sheet music and motivate the unique challenges it presents. We then show a first step towards a solution, by building a system to solve a more constrained version: one in which only distinct notes are present. This pipeline consists of synthetically generating labeled training examples, finding measure bounds and estimating the perspective via a Hough Transform, and training sliding window detectors to infer the pitch and type of each note.

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.

This work is the beginning of a project which attempts to automate that click. Namely, I wish to develop an iPhone or Android application in which a user can point at never-before-seen piece of sheet music in standard lighting conditions, and hear the song. As a first step, I here consider the task of reading sheet music from a single image, and playing it in audio.

The remainder of the paper is laid out as follows: in Section 2 we discuss relevant prior work. In Section 3 we formalize the general problem, and characterize the variety found in real-world sheet music and subtleties inherent to the task. In Section 4 we present a first step towards the general problem, by building an end-to-end pipeline to solve a simplified version of this task. We evaluate our results in Section 5.

2. Prior Work

To my knowledge, the problem of sheet music recognition had only been considered in the realm of scanned sheet music (see [6] and [5]).[1]

Additionally, the task of sheet music recognition does have many similarities to Optical Char-

[1] This paper ([1]) has recently come to my attention. It may well be that this has been done before, although not in a mobile phone setting.

acter Recognition [7], in that it attempts to read from a discrete set of characters printed on a page. However, while the meaning of characters in OCR is given solely by their shape, the meaning of notes is dictated in part by their shape, and in part by their position and location relative to other symbols.

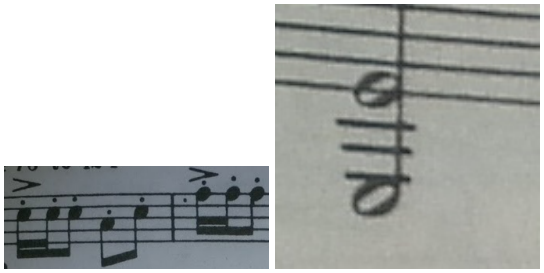

Figure 1. Example groups from user-submitted sheet music

3. Problem Statement

The task of this work is as follows: given a single image containing one or more measures of printed music (or "score") output a *song*: a potentially overlapping collection of pitches, start times, and durations (in relative units of "beats"). While there are many minor points and subtle deviations, nearly all sheet music consists of the following components:

- The staff: a set of 5 evenly spaced horizontal lines, over which all notation must lie. Symbols present at the start of the staff determine how notes will be interpreted in subsequent measures.
- Note: a particular symbol whose shape, combined with its relation to neighboring symbols, dictates its duration. Its vertical placement dictates its pitch. We consider the octave which is fully contained within the staff lines, and thus the pitches referred to are enumerated: E,F,G,a,b,c,d,e.
- Accidental: a symbol (sharp, flat, natural) which, when placed to the left of a note, modifies its pitch.
- Rest: a symbol whose shape dictates the amount of time no note will be played.

The following are terms used to throughout this paper:

- Beat: The fundamental unit of time in music. Assuming 4/4 time, it is represented by a distance of $b_x = \frac{m_w}{bpm}$ where m_w is the width of a measure and bpm is the number of "beats per measure"—assumed 4 throughout the paper.
- Step: The fundamental unit of pitch, corresponding to $\frac{1}{8}$ an octave. Although musically incorrect given the presence of accidentals, in this work I refer to a "step" as a change in pitch by a single letter value, signified in sheet music by a vertical traversal of $p_y = \frac{m_h}{8}$ where m_h is the height of a measure. Staff lines are separated by a distance of $2p_y$.

3.1. Difficulties

Despite the rigid structure of sheet music, it presents a number of surprisingly difficult challenges. For instance, while an individual note may be fully understood by its appearance, it is frequently the case that notes are grouped together or stacked atop each other (see Figure 3.1). While the latter case poses a great challenge to precise localization, the former case proves even more difficult: the duration of a grouped note can only be determined by the way in which it is connected to its neighbors. Furthermore, because notes may be arbitrarily grouped or stacked, it is infeasible to simply enumerate all possible symbols and treat them as different detections: they must be, in a sense, understood.

4. Our Pipeline

The above problems proved extremely interesting: far too interesting, unfortunately, for time constraints to allow. After many unsuccessful attempts, I chose instead to begin with a proof of concept. To do so, I first greatly simplify the problem by removing the challenges presented by local context. Namely, I consider only sheet music comprised of a single melody line with physically disconnected notes and no accidentals. In what

follows, I detail my pipeline. What pieces of this pipeline may generalize, and what was learned in the process, will be discussed in Section 6.

4.1. Overview

In both training and testing, we begin with an image of sheet music, taken by a camera phone. To efficiently generate labeled training examples, I generated my own dataset of synthetic sheet music. I then photograph this sheet music, and attempt to warp it back into its canonical reference frame. Note detectors are then trained and run on these warped images on a per-measure basis, and used to predict the type and pitch of each note. Finally, these are strung together into a song, which may be played through speakers.

4.2. Implementation Details

As I intend to deploy the working system on a phone in the future, I implemented this on a platform which is portable to both Android and iPhones: OpenCV [2], particularly its Python bindings.

Much functionality—image processing and finding lines via a Hough Transform, for instance—is built into this library. Support Vector Machine training was done with both LIBSVM [3] and PyML[**?**].

To organize pieces, allow for ease in open sourcing, and ideally port this to the PR2 upon completion, this work was developed as a package in the Robot Operating System (ROS).

4.3. Dataset Generation

One great challenge in the initially-proposed general problem was the issue of scalability. Namely, for every user-submitted image, generating training data required meticulously labelling potentially-skewed bounding boxes. After a number of attempts (see Fig. 2) it became painfully clear that this would not scale to the large number of images I would like the end-result to handle.

To automate this procedure, as well as to ensure that we are only given music which follows our simplifying assumptions, I implemented my own

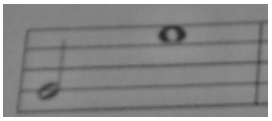

Figure 3. When a perspective is inferred from lines on another part of the image, other measures are skewed.

sheet music generator. It first generates a random song of desired length, assuming a uniform distribution of note types and pitches. It then renders this visually on a score, such that the rules of our domain (such as the number of beats per measure) remains consistent. The layout (e.g. measures per row, staff height) may be varied, to ensure we do not overfit to a particular scale. The note symbols themselves were taken from online sheet music. In these experiments, only a single typeface was used.

This rendered sheet music was then printed out and photographed under real-world lighting, perspective, and blur. Each photograph was paired with metadata about the song which generated it, so that the bounding boxes and note labels in the original score would be known, both for use in supervised training, and ground truth in testing.

4.4. Perspective Estimation

As the vertical location of a note on its page uniquely determines its pitch, precise localization of the note in the page's reference frame is necessary. Thus, to aquire this information and reduce noise in the classification task, we first wish to warp each note into a canonical reference frame.

As there is often curveature in the paper's surface (as is the case when held and, particularly, near the spine of a booklet), no global perspective projection is sufficient to project onto the paper's reference frame (see Fig. 4.4). However, empirical results show that within a single measure, the perspective is well-approximated by an affine projection (see Fig. 4.4). Thus, for each measure, we wish to infer an affine projection matrix P which will make its staff lines horizontal, and its width and height that of an arbitrarily sized canonical measure. This can be uniquely determined from 3 points.

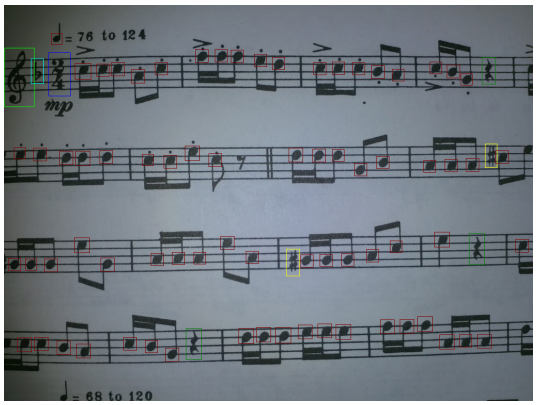

Figure 2. An example of labeled bounding boxes in a small region of a user-submitted score

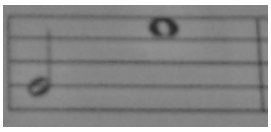

Figure 4. In the local measure regime, however, they are fairly well normalized.

We thus look for the bounding corners of each measure, located at the intersection of the top and bottom staff lines with the left and right measure bars. More precisely, we wish to locate *crossings* in the image: those points in which the vertical measure bars and horizontal lines of the staff intersect. A number of attempts were tried: the most effective was to use a Hough Transform on a dilated Canny image to locate predominantly vertical and horizontal lines in the image, and compute their intersection. An example result is shown in Fig. 5.

While this proved somewhat promising, the predicted intersections were quite noisy—particularly due to the fact that the tail of notes strongly resemble vertical measure bars, and also intersect the staff. Unfortunately, imprecision in this step proves disasterous in future iterations. Thus, while a solution certainly may be found, I chose to experiment the success of the detectors using hand-annotated crossings, and leave fine-tuning this step for future work.

4.5. Note Detection

Given a measure in a canonical reference frame, the next step is to detect, and precisely locate, notes in the image. To do so, I use a patch-based sliding window classification scheme.

In particular, at train time, patches are are sampled both at the ground-truth location of each note. To avoid overfitting to our automated bounding boxes (which capture no translation invariance in the note), we additionally sample from shifted versions of each, given random noise. Negative training examples are drawn from locations in which there is guaranteed to be no note given the structure of music—namely, in the beats directly following a half or whole note. As we generated this particular training data, all bounding boxes were known explicitly, so there was no risk of sampling unlabeled positives: however, it is interesting to note that this method would be robust, even when not all positives are labeled.

Given a patch classifier, the note detection sequence proceeds as follows:

- Pass through the image measure by measure
- The measure is traversed beat-by-beat ($dx = b_x$).
- For each beat location, the staff is traversed step-by-step ($dy = s_y$).
- For each candidate pitch location, patches are sampled from a small ($|dx| < frac b_x 2, |dy| < frac s_y/2$) neighborhood around this location.
- Each patch in the neighborhood is classified as “whole”, “half”, “quarter”, or “none.” If

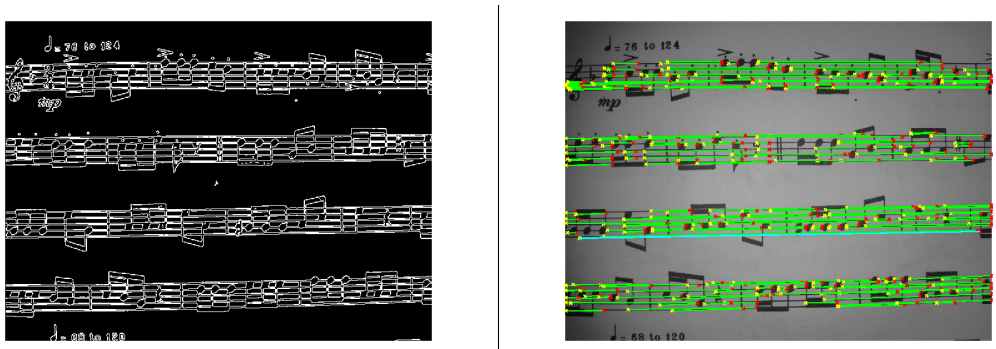

Figure 5. Left: The canny image. Right: Hough lines which are found. Note the noise.

any positive note labels are predicted, the one with the highest confidence is selected as the candidate at that pitch.

- The pitch with the maximally responding candidate is chosen, and the note is labelled accordingly.

To compensate for lighting effects, the image was first made to have a consistent local mean, by convolving it with the filter:

$$K = \begin{pmatrix} \frac{-1}{k} & \cdots & \frac{-1}{k} & \cdots & \frac{-1}{k} \\ \frac{-1}{k} & \cdots & 1 - \frac{1}{k} & \cdots & \frac{-1}{k} \\ \frac{-1}{k} & \cdots & \frac{-1}{k} & \cdots & \frac{-1}{k} \end{pmatrix}$$

$$I_m = K \star I + 128$$

To extract features, patches were normalized to a size of 26x26. I then tried a number of features, including the gradient image, Canny edge image, and HOG [4] (as implemented in OpenCV). In the end, however, raw grayscale pixel values proved to be sufficient for the task, and enabled faster computation.

Two classification techniques were considered:

- K-nearest neighbors and SVMs. The K-nearest neighbors were computed using the FLANN library [8], and their "confidence metric" was given by the the negative distance from the nearest neighbor of the same label.

- Linear Multiclass SVMs were trained using the PyML library, with leave-one-out cross validation. Their confidence was given by the distance from the feature to the decision boundary.

- Gaussian RBF Kernel SVRs were trained using LIBSVM, with a grid search used to select the parameters, and their confidence values explicitly given.

4.6. Audio Generation

The above details the end of the vision parts of this task, converting an input image to an output sequence of notes with corresponding pitches and durations. As a simple proof of concept, I wrote a script which takes input notes of the same pitch/type format and outputs the audio of a song, using the tkSnack library to generate waveforms. However, due to time constraints, this piece has not yet been integrated into the end-to-end system.

5. Results

The above system was run on a dataset of images, collected by Android and iPhone cameras, under multiple lighting conditions: indoor lighting, sunlight, and camera flash. They were taken such that the entire width of the score was in view and reasonably focused, lying on a planar surface, subject to reasonable perspective effects but remaining generally upright.

I trained on a set of 23 images and tested on a set of 5, where the test images had noticably more visible notes than the training. These yielded roughly 150 and 80 instances per class, respectively.

	LIBSVM			KNN			PyML		
	w	h	q	w	h	q	w	h	q
w	77	2	1	70	10	0	35	26	19
h	0	88	0	0	84	4	30	44	14
q	0	5	98	0	8	95	37	35	31
misses	0	0	0	0	0	0	0	0	0
falsep	5	1	0	40	159	3	89	98	18

Figure 6. Confusion matrices and precision recall for the detectors.

The results of the 3 classifiers are shown in Fig. 5. The three classifiers yielded absolute pitch errors of 0.22 steps, 0.51 steps, and 0.69 steps respectively.

As can be shown form the provided results, the LIBSVM detector does quite well at inferring the music, even given only raw black and white features. However, KNN, which is much simpler and faster to train and test on, does not suffer much, despite having no analog to the cross validation of the SVM detector. The performance of the PyML detector is notably poor, although it may be that, without proper regularization, it is simply fitting to a great deal of noise: this is most evident by the high number of false positives, in which notes were detected in empty space.

6. Conclusion

In this work, I attempted to use Computer Vision techniques to autonomously play sheet music from a single image. So doing, I quickly learned that the world of sheet music is much more varied than I'd originally believed, and that the problems inherent to it were quite difficult. I chose, instead, to consider a limited, toy subset of possible sheet music instances.

I proposed a mechanism for generating arbitrary amounts of labeled training data on the fly. I then used this data to get real-world photos of sheet music on standard camera phones. By using correspondences (currently hand-labeled) between measures, I transformed the sheet music to its approximate reference frame. I then trained discriminative classifiers to detect notes and, from their detections, infer their pitch. These detections were combined to create a song, which can be played on a computer's speakers.

That said, there were many issues with this work, largely due to time constraints. For one, this could have easily been extended to recognizing more note types, even without solving the problem of local context. Further, the lack of variation in the training and test data means this may be overfitting to the typefaces used: although the variation in typefaces appears to be quite minimal across all scores.

In the future, I am excited to port many of these ideas to a more scalable system. In particular, the ability to generate arbitrary training data may be of great use in a scalable system. Furthermore, the idea of using partially labeled data (such as ground truth in the music's reference frame) can be extended to user submissions, by requesting that some metadata be provided with each score, such that the ground truth can be looked up autonomously.

References

[1] P. Bellini, I. Bruno, and P. Nesi. Optical music sheet segmentation. In *Web Delivering of Music, 2001. Proceedings. First International Conference on*, pages 183–190. IEEE, 2001.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. Ieee, 2005.

[5] C. Fremerey, M. Müller, F. Kurth, and M. Clausen. Automatic mapping of scanned sheet music to audio recordings. *Proceedings of the ISMIR, Philadelphia, USA*, pages 413–8, 2008.

[6] F. Kurth, M. Müller, C. Fremerey, Y. Chang, and M. Clausen. Automated synchronization of scanned sheet music with audio recordings. *Proc. ISMIR, Vienna, AT*, pages 261–266, 2007.

[7] S. Mori, C. Suen, and K. Yamamoto. Historical review of ocr research and development. *Proceedings of the IEEE*, 80(7):1029–1058, 1992.

[8] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Int. Conf. on Computer Vision Theory and Application (VISSAPP)*, 2009.

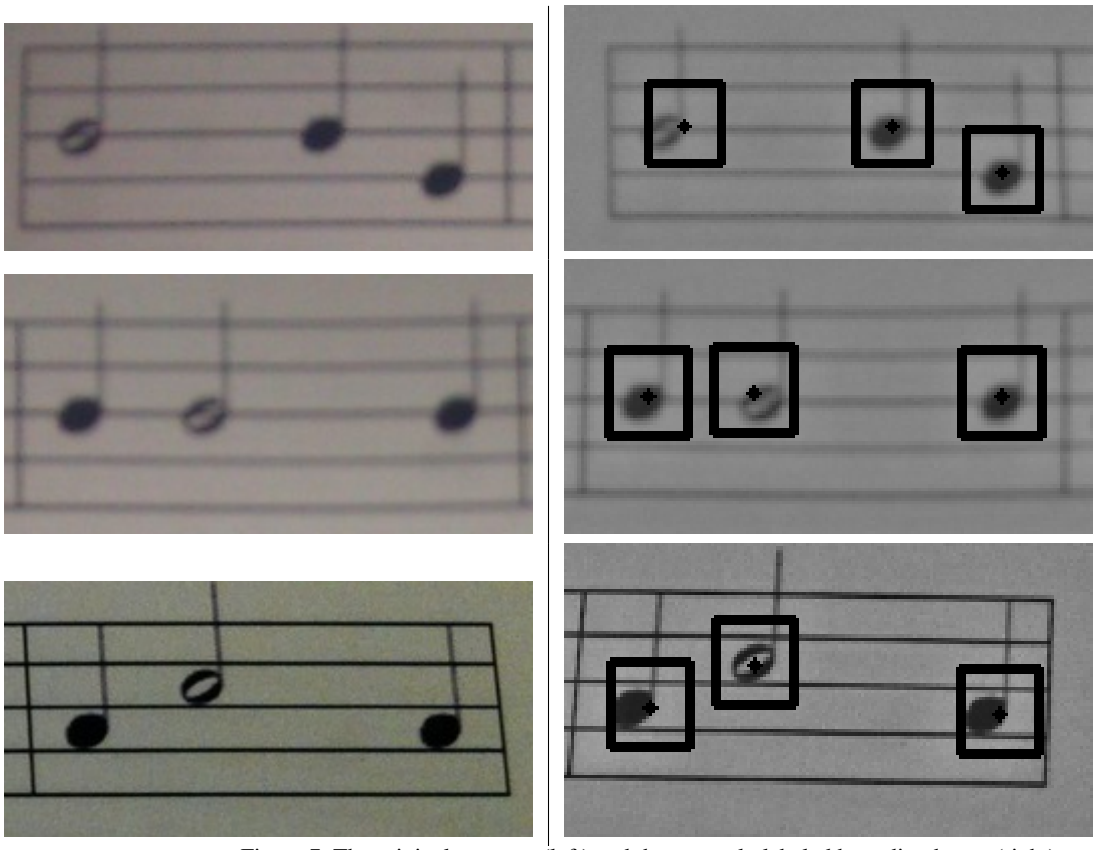

Figure 7. The original measure (left) and the correctly labeled bounding boxes (right)