INSTRUMENT IDENTIFICATION IN OPTICAL MUSIC RECOGNITION

Yucong Jiang
School of Informatics and Computing
Indiana University, Bloomington
yujiang@indiana.edu

Christopher Raphael
School of Informatics and Computing
Indiana University, Bloomington
craphael@indiana.edu

ABSTRACT

We present a method for recognizing and interpreting the text labels for the instruments in an orchestra score, thereby associating staves with instruments. This task is one of many necessary in optical music recognition. Our approach treats the score system as the basic unit of processing. A graph structure describes the possible orderings of instruments in the system. Each instrument may apply to several staves, may be represented with several possible text strings, and may appear at several possible positions relative to the staves. We find the optimal labeling of staves using a globally optimal dynamic programming approach that embeds simple template-based optical character recognition within the overall recognition scheme. When given an entire score, we simultaneously optimize on the text labeling for each system, as well as the character template models, thus adapting to the font at hand. Our implementation alternately optimizes over the text label identification and re-estimates the character templates. Experiments are presented on 10 different scores showing a significant improvement due to adaptation.

1. INTRODUCTION

In some scores, particularly those for small ensemble, instruments appear in the same position in all systems making it easy to associate instruments with staves. This scheme would be typical for a string quartet or sonata for solo instrument and piano. Occasionally large-ensemble scores follow this convention as well, though it requires considerably more space as empty staves must be written out every time any instrument is not used in a particular system, thus creating longer scores and lowering the density of information. For these reasons many publishers avoid this layout style, instead notating only the instruments that play in a particular system. In this case text labels, usually appearing in the left margin of the system, identify the instrument(s) associated with the individual staves, as in Figure 1. These are the scores we treat here, while our goal is the labeling of each staff with its associated instrument. Such labeling is necessary for nearly any aspect of optical music recognition (OMR) using "instrument-labeled scores", as it allows one to link systems together in a meaningful way.

The first steps of our OMR system [9] are to identify the staves in a page, and then to group these into systems. While these tasks present challenges due to the wide variation in printed scores, they are among the easier OMR tasks, and are handled reasonably well by our system. The resulting score systems, including the precise locations of all the staves they contain, constitute the input to our staff labeling process.

In spite of the mature nature of optical character recognition (OCR), our staff labeling problem is highly challenging when viewed purely in these terms. The text strings we seek to recognize are usually a single word or abbreviation, thus providing only a small portion of data for each recognition problem. Furthermore, even though the vocabulary is constrained to the instrument names used in the score, OCR will encounter difficulties distinguishing similar strings, such as "Violin I" and "Violin II," or "Vln.", "Vla.", and "Vcl.". Finally, there often is other irrelevant text in the area of the names we seek to recognize, further hindering the recognition. But even if OCR were enough to recognize the instrument names, our goal includes more than this. As it is common for text strings to apply to multiple staves in a score, we need the name-to-staff mapping as well.

Thus, unlike the bottom-up approach used in [12], our top-down approach uses a graphical model that generates all legitimate possible partitions of the system into staves and all legitimate possible labelings of these partitions with instruments. The graphical representation enables a dynamic programming approach that embeds rather generic OCR into the "innermost loop" for the recognition of individual text labels. The model may include strong assumptions about the possible orderings of instruments, with the most obvious choice being that the instruments appear in the order initially given on the first score page, (though any subset of instruments can be omitted).

Perhaps the biggest challenge of our task is the font variation between scores. Even controlling for the height of the staff, one still sees considerable variation in the size and shape of the characters between fonts. It might be possible to develop an *omnifont* approach [2,3], meaning a text model that is trained from a variety of fonts, and thus capable of recognizing this same variety. However, as character models are required to accept a wider range of presentations for each given letter, they become less capable of dis-

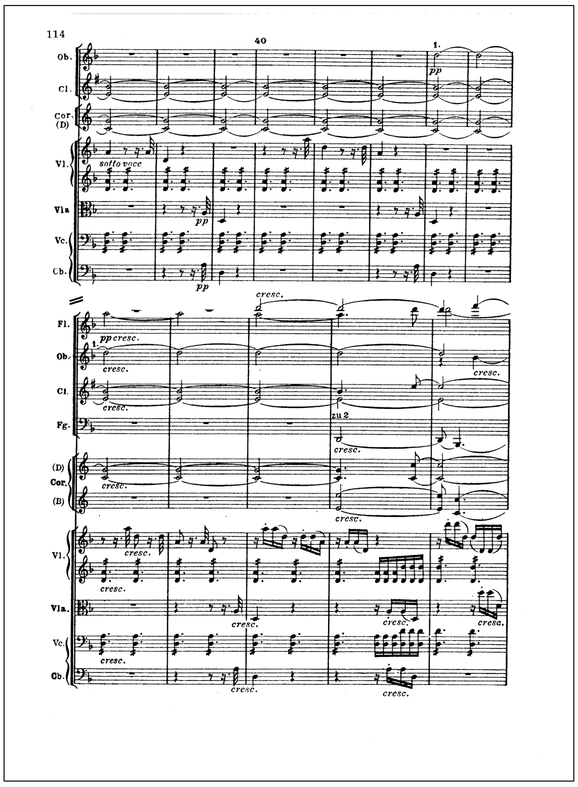

Figure 1. One page with an 8 staves system and an 11 staves system.

tinguishing between different letters. For this reason omnifont models generally perform worse than models tuned for the specific task at hand.

Borrowing a well-known idea from pattern recognition, [11, 13], we address this challenge by *simultaneously* recognizing the instrument labels on the entire collection of systems in a score *and* learning the font model for the document at hand. Our algorithm iteratively recognizes the systems, then retrains the font models using the optimal labeling and text alignments produced in the recognition phase. One might expect that this approach is simply too greedy to succeed, failing to explore the high-dimensional world of possible character models and system interpretations, while almost guaranteed to get stuck in a mediocre local optimum. However, we present experiments that show a large *monotonic* improvement in recognition accuracy as we iterate this process, culminating with excellent recognition results. The approach is feasible due to the highly restrictive assumption made by our graphical model, thus constraining the admissible interpretations to a tiny fraction of those arising without this restriction. Our experiments demonstrate that this graph model is the difference between basically successful and unsuccessful results.

2. LABELING STAVES WITH INSTRUMENTS

The usual notational convention for large-ensemble scores lists all instruments on the first page of a piece or movement, whether or not the instruments play in this page [10].

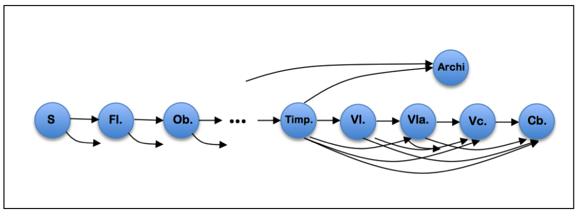

Figure 2. A directed graph representing the possible orderings for the instruments.

In subsequent pages, instruments, perhaps in abbreviated form, are written in the left score margin. Usually the instrument labels are displayed immediately to the left of the associated staff line, though the labels sometimes describe *collections* of staves, such as "Strings", "Horns", etc. In such a case the text label usually appears centered with respect to the group of staves, often emphasized by a bracketing of the associated staves in the left margin of the score. Figure 1 shows a typical example.

2.1 THE MODEL

Our staff labeling procedure requires input from the user explaining the labeling scheme(s) used in the score at hand. Typically, the instruments appearing in a system are a subsequence of the order given on the first page of the score. However, variations are possible such as substituting a collective name for the individual labels, e.g. using the single text label "Strings" instead of the the individual labels "Violin 1", "Violin 2", "Viola", "Violoncello" and "Bass" [10]. We assume that the possible labelings can be described by a directed graph, G, as in Figure 2, where the possible paths through the graph give the legitimate label sequences. We assume the graph is supplied by the user either implicitly or explicitly. Each vertex $g \in G$ is associated with an instrument, $I(g)$, so recovering the correct path will give the sequence of instruments employed in the system.

As mentioned above there may be several staves associated with a particular instrument or group of instruments, though we do not require such a convention to be followed consistently. We rely on the user to list, for each instrument, the possible labeling variations encountered in the score. We describe this information as a collection of *patterns* for each graph node, $P(g) = \{p_1, \ldots, p_c\}$, where we suppress the dependence of the list length, c, on g in our notation. Each pattern, p has three attributes: $p = (p^k, p^l, p^a)$ giving the number of staves used for the instrument(s), p^k, the location of the text label with respect to the group of staves, p^l, and the specific character sequence used for the text label, p^a. For instance $p^l = 0$ would mean that the label appears in the middle of the group of staves (next to the middle staff if p^k is odd and between the middle two staves if p^k is even). When $p^l \neq 0$, p^l gives the integral number of inter-staff half spaces above or below the middle location where the text will be found. In Figure 1 $p^l = 0$ for all instruments.

Every pair of adjacent vertices, g', g are connected by

$|P(g)|$ directed arcs labeled with the various patterns, $P(g)$, though these are not explicitly drawn in Figure 2. Thus there may be several arcs that connect g' to g, each accounting for a possible number of staves for the instrument, $I(g)$, location of the text label, and the actual text itself. There is a one-to-one correspondence between the allowable labelings of each system and the legal paths through the graph. That is, if $s, g_1, \ldots, g_M$ is a path beginning from the start vertex, s, with arc labels $p_1, \ldots, p_M$ that correctly accounts for the number of staves in the system, N,

$$\sum_{m=1}^{M} p_m^k = N \tag{1}$$

then the labeling associates the first p_1^k staves with instrument $I(g_1)$, the next p_2^k staves with instrument $I(g_2)$ and so on. The path may terminate anywhere in the graph other than at the start vertex, s, as long as Eqn. 1 is satisfied.

2.2 RECOGNIZING THE TEXT LABELING

We score every legal path through G as a sum of arc scores and compute the best scoring path through dynamic programming (DP). For this purpose we define $E(n, g)$, for $n = 0, \ldots, N$, $g \in G$, as the best scoring interpretation of the first n staves ending in state g. We compute E by initializing $E(0, s) = 0$, then visiting the staves in order: $n = 1, \ldots, N$ computing for each $g \in G$,

$$E(n, g) = \max_{g' \xrightarrow{p} g} E(n - p^k, g') + L(n, p) \tag{2}$$

where the maximum is over all legal arcs going from g' to g with $p^k \leq n$. In Eqn. 2 $L(n, p)$ is the arc score measuring the plausibility that the text label p^a positioned at relative position p^l is used for staves $n - p^k + 1, \ldots, n$. While we present L in more detail in Section 2.3, for now it suffices to say that L measures the quality of the best match of the text, p^a, to the score image data in the area determined by n, p^k, p^l. It is worth noting that $L = 0$ is a neutral result, meaning that the optimal placement of the letters of p^a explains the data as well as a background model. In contrast, positive (negative) scores of L indicate evidence for (against) the labeling implied by the transition of $g' \xrightarrow{p} g$. Thus our algorithm has no inherent bias for assigning more or less text labels in the optimal interpretation.

Having computed $E(n, g)$ for $n = 1, \ldots, N$ and $g \in G$, the score of the optimal path is given by $\max_{g \in G} E(N, g)$, while it is a simple matter to recover the optimal sequence of vertices and transitions that produce the optimal score. We denote these by $g_1^*, \ldots, g_M^*$ and $p_1^*, \ldots, p_M^*$.

2.3 CHARACTER RECOGNITION

Our approach to character recognition is standard template-based [7, 8], and will only be discussed briefly and informally here. $L(n, p)$ evaluates the quality of the hypothesis n, p. The information contained in n and p collectively describes a reasonably precise vertical location in the image. The task in computing L is to search the area around this position for the optimal locations of the characters of p^a, subject to reasonable constraints regarding their spacing.

Suppose the characters of p^a: $c_1, \ldots c_L$ have rectangular templates $m_1, \ldots, m_L$ which are hypothesized to be placed at image locations $(x_1, y_1), \ldots, (x_L, y_L)$. Each template is a matrix of values $m_l(i, j) \in \mathcal{M}$ where $\mathcal{M} = \{b, w, t, n\}$ indexes *black*, *white*, *transitional*, and *null* grey level probability models denoted by P_b, P_w, P_t, P_n. P_b mostly "expects" to see low grey levels, P_w "expects" to see high grey levels, P_t is a mixture of these two models, and P_n is a *null* or *background* model taken as the normalized grey level histogram of the entire image. $L_{\mathbf{x},\mathbf{y}}(n, p)$ is then defined to be the normalized data log likelihood given by

$$L_{\mathbf{x},\mathbf{y}}(n, p) = \sum_{l=1}^{L} \sum_{i,j} \log \frac{P_{m_l(i,j)}(J(x_l + i, y_l + j))}{P_n(J(x_l + i, y_l + j))} \tag{3}$$

where $(\mathbf{x}, \mathbf{y})$ denotes the entire collection of template locations, $J(x, y)$ is the image grey level at pixel (x, y) and the inner sum uses the range for i, j appropriate for the lth rectangular character template.

In computing Eqn. 3 we consider a variety of possible vertical positions for the text baseline, and all reasonable positions for the characters along that baseline so that

$$L(n, p) = \max_{\mathbf{x},\mathbf{y}} L_{\mathbf{x},\mathbf{y}}(n, p).$$

Thus the computation consists of a loop over baseline positions with each iteration accomplished by a DP computation that optimally locates the character templates.

2.4 SIMULTANEOUS OPTIMIZATION

Section 2.2 gives our procedure for finding the optimal text labeling for the staves of a system. Computing this labeling requires at least reasonable character templates, though it would be preferable to have templates that represent the font at hand. Unfortunately, fonts differ greatly from one music document to another, both in size and shape, so we have no way of knowing *a priori* the font used for instrument names in any given score. Our approach here is to *simultaneously* estimate both the optimal text labeling and the optimal character templates, thus *adapting* to the font at hand while we recognize. While simultaneous estimation of both interpretation and model parameters is infeasible for many recognition problems, we rely here on the strong graph-based assumptions we have made on the family of possible labelings. In essence, our assumptions about instrument order are powerful enough to get reasonable estimates of the instrument labels even with poorly specified character templates. Thus we can use this labeling, and the precise character template positions that come with it, to re-estimate our character templates. Our overall approach then becomes an iteration between the (re)estimation of instrument labels and the (re)estimation of character templates, similar with [5, 6]. In practice this approach converges after only a few iterations, and usually does so with significantly better recognition accuracy than with the original character templates, as discussed in Section 3.

More precisely, we let z denote a possible text labeling for the entire collection of system staves. Thus z includes a path of vertices and arcs through G for *each* system in the score, as discussed in Section 2.2, as well as the character template positions that result from the $L(n,p)$ computations of Section 2.3. Let θ denote the complete collection of character templates employed in Section 2.3, for all letters and punctuation used in the text labels. Finally we let $\bar{E}(z,\theta)$ be the summed data log likelihood score of Eqn. 3 produced by evaluating the complete set of recognized characters in z at their precise positions using the character templates of θ.

Starting from our initial character templates θ_0, the basic two-stage iteration of our algorithm is then

$$z_{l+1} = \arg\max_z E(z,\theta_l) \quad (4)$$

$$\theta_{l+1} = \arg\max_\theta E(z_{l+1},\theta) \quad (5)$$

for $l = 0, 1, \ldots$. The update for z in Eqn. 4 is simply the dynamic programming procedure from Section 2.2 applied to each system in the score, which is guaranteed to produce a global optimum for z. The θ update is accomplished by maximum likelihood estimation, as follows. Suppose our alphabet of characters and punctuation is $c^1, \ldots, c^Q$. Also suppose that c^q appears at locations $(x_1^q, y_1^q), \ldots, (x_R^q, y_R^q)$ as defined through the text labeling and implicit template alignment of Section 2.2. We then let

$$c^q(i,j) = \arg\max_{\mu\in\mathcal{M}} \sum_{r=1}^{R} \log \frac{P_\mu(J(x_r^q+i, y_r^q+j))}{P_n(J(x_r^q+i, y_r^q+j))}$$

for $q = 1, \ldots, Q$, which is, by definition, Eqn. 5.

The proposed algorithm is simply coordinate-wise optimization over z and θ, which guarantees that the sequence $E(z_1,\theta_1), E(z_2,\theta_2)\ldots$ is non-decreasing. Furthermore, the sequence is guaranteed to converge due to our finite (but large) domain. In practice, this happens in only a few iterations.

It is worth noting that our strategy is different from the usual EM scheme [4] for performing maximum likelihood estimation of model parameters. To implement EM we would need a probabilistic model for the graph transitions, which would be easy to supply, but absent from our current formulation. However, EM attempts to increase the *marginal* data likelihood rather than the likelihood of the optimal path. Thus, if we were to replace our parameter estimation step by an iteration of EM, we still may end up decreasing our objective function. That said, EM is less greedy in its approach than our proposed algorithm, which, in principle, seems like a good attribute. In practice, we doubt there would be any significant difference in the performance of these two approaches.

3. EXPERIMENTS AND RESULTS

Table 1 describes the collection of scores used in our evaluation, all obtained from the IMSLP website [1], consisting of 10 scores from 6 different publishers. In our evaluation we used about 20 pages from each score.

score index	IMSLP ID	pages	Publisher
1	24831	50-69	New York: Charles Foley
2	03631	2-21	Moscow: Muzgiz/Muzyka
3	65460	2-18, 21,22,23	
4	00569	2-19	Leipzig: Breitkopf & Härtel
5	31875	2-21	
6	01086	2-20	
7	06307	2-21	Leipzig: Ernst Eulenburg
8	00191	2-21	
9	08535	2-21	Vienna: Universal Edition
10	07354	2-25	Berlin: Schlesinger

Table 1. Information about the scores.

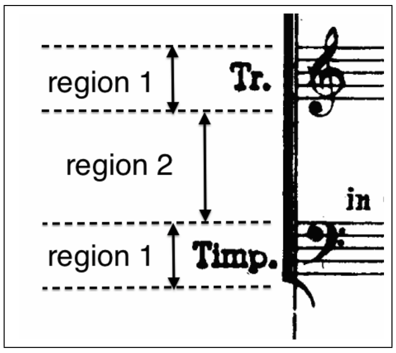

Figure 3. Two kinds of regions for possible text positions.

Our OMR system begins by computing the locations of the staves and the partition of staves into systems for each page of the score. These systems constitute the input to our system.

In all cases we use a graphical model based on the first page of the score, allowing for any subsequence of the named instruments, as in Figure 2. Several scores allow for the collective labeling of the strings with a single text tag, rather than an enumeration of instruments. For each $g \in G$ (i.e. each instrument) we supply the appropriate patterns, $P(g) = \{p_1, \ldots, p_c\}$, by hand. Some instruments have two patterns, though most have only one. For all of the instruments and scores in our test set we only consider patterns where $p^l = 0$ meaning that the text must lie in the middle of the collection of staves associated with an instrument. Referring to Figure 3, this means we search in the text in region 1 when $p^k = 1$, and region 2 when $p^k = 2$, with obvious extensions to larger staff groupings. All of our test scores place instrument names in the left margin, though our approach easily accommodates other possible positions. We also supply the text strings, p^a, and the number of staves for each pattern, p^k. Even with instruments having two patterns, both used the same text (p^a) in our models.

score index	number of staves	number of errors			
		θ_0	θ_1	θ_2	θ_3
1	399	19	4	2	**1**
2	395	13	**0**		
3	317	21	10	4	**0**
4	331	44	**0**		
5	391	11	8	**1**	
6	443	33	2	**0**	
7	273	3	2	1	**0**
8	364	34	**0**		
9	381	4	**0**		
10	436	72	52	45	38
		θ_4	θ_5	θ_6	θ_7
		25	23	22	**21**

Table 2. Total number of staves and number of errors in each iteration for each score.

3.1 ORIGINAL TEMPLATES

The character template set includes all the (case sensitive) letters used in the instrument names of the 10 scores, in addition to a comma, hyphen, period, and space, giving 34 characters in all. Each score uses a subset of this collection in labeling instrument names. We create our original set of templates, θ_0, by, for each character, randomly choosing an example from one of the 10 scores and thresholding the grey levels to choose probability models for each pixel. This collection will be the initial configuration for all 10 scores before we begin the adaptation process. We sampled from the test scores as a way of ensuring we could find examples of all the required templates, and note that, on average, only a tenth of these initial templates come from any individual score. A better scheme might use an omnifont model as our starting place.

We simultaneously estimate the staff labelings and the trained collection of character templates, iterating the approach of Section 2.4 until the results converge. Table 2 lists the total number of staves assigned incorrect instrument labels after each iteration of the algorithm, for each of the 10 scores. As shown in the table, 7 scores correctly labeled all the staves, 2 scores have only one labeling error, while the 10th score has 21 out of 436 staff labeling errors (4.82%), which is still low. The algorithm converged on all scores within 4 iterations, except for the the score containing the most errors, which used 8 iterations, as shown in the table.

As the original set of templates, θ_0, comes from a variety of different scores with different fonts and sizes, they don't match any particular score font well. Our hope is that, through our iterative training process, the templates will *adapt* to the current score. Figure 4 gives an example from Score 6 with the original and learned characters drawn on top of the score image at their estimated locations. Clearly the original templates matched the actual font poorly, especially in size, as can be seen in the middle panel of the figure. The right panel of the figure shows the analogous result after two iterations of recognition and

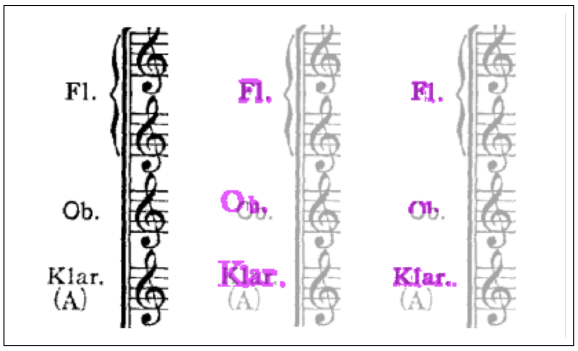

Figure 4. Comparing templates before (middle) and after (right) training. The instrument names in the actual score (left) are "Fl.", "Ob." and "Klar." in order.

retraining. After this process most of the character templates match the font better, though not all of them. Due to the greedy nature of our algorithm it is necessary that our original templates, hence the original recognition and match, are close enough to pull the result into the correct local optimum. In this case the 'O' and 'b' in "Ob." were misspecified and consistently matched poorly in the first iteration. As these characters don't appear in other instrument names, there was no counteracting force helping to guide the models toward reasonable results, thus the outcome of the figure.

Although some of the trained results don't look particularly good, Table 2 shows striking improvement in staff labeling due to training, *showing monotonic decrease in the number of errors*. Here is where the strength of the graphical model comes into play. Even with the poorly specified character models for "Ob.", this is the only instrument name that can come between "Fl." and "Klar." for which we have good models. This leads to the correct labeling in spite of the uneven training.

There are three scores having errors in our experiments. The one error in Score 1 is caused by unrelated text appearing in the left margin which was recognized as an incorrect instrument name. The one error in Score 5 mistakes "Vc." as "Vla." with "l" and "a" squeezed together. This happens in a three-staff system, thus the constraints imposed by graph ordering are less potent.

For Score 10, the errors are caused by badly trained templates. 8 out of 19 templates used in this score converged into unrecognizable glyphs. We suppose this happens because the font size of this score is obviously smaller than other scores and thus harder to adapt to. But surprisingly, this score still has reasonably accurate instrument labeling, which is our objective.

3.2 DROPPING THE GRAPHICAL CONSTRAINT

For comparison we ran a similar experiment without the ordering constraint imposed by the graph, thus allowing any group of staves to be labeled with any instrument. In this case all instrument orders are possible, even allowing for repetition of instruments. The results are shown in Table 3. After four iterations, the number of errors doesn't seem to

score index	**number of staves**	**number of errors**			
		θ_0	θ_1	θ_2	θ_3
5	391	78	71	69	73
6	443	59	49	48	57
8	364	102	91	103	103

Table 3. Total number of staves and number of errors in each iteration for 3 scores without the graphical ordering constraint.

decrease. This is because many of the trained templates become unrecognizable — some of them become pure white space! Without the strong ordering constraint there is less gravitational pull toward the desired optimum, while we imagine the joint space of interpretations and models to be filled with local optima.

3.3 WORD MODELS VS. CHARACTER MODELS

Out of curiosity, we modified our model to view the instrument names as single rigid glyphs rather than character based models that allow for some flexibility in the placement of individual characters. Our experiments (not presented here) show that this approach works well when the actual document is consistent with the assumption we are making, but fails badly otherwise. Given the wide variety of typographical conventions encountered in music scores, we don't recommend this approach.

4. CONCLUSIONS

We have presented a method of interpreting the instrument name labels, which are a common way of labeling staves in large ensemble scores, showing nearly perfect recognition in all but one of the test scores we examined. The unusual aspect of our approach is that we simultaneoulsy estimate both the labels we seek, as well as the text font used for the score. The experiments show convincing evidence that the strong assumption we make regarding possible labelings is powerful in practice and largely responsible for the reliability of the approach. In future work we will consider initial text models trained from a large variety of scores, as well as feature based, rather than template based, data models.

5. REFERENCES

[1] IMSLP website. `http://imslp.org`.

[2] Henry S Baird and George Nagy. Self-correcting 100-font classifier. In *IS&T/SPIE 1994 International Symposium on Electronic Imaging: Science and Technology*, pages 106–115. International Society for Optics and Photonics, 1994.

[3] Issam Bazzi, Richard Schwartz, and John Makhoul. An omnifont open-vocabulary OCR system for english and arabic. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(6):495–504, 1999.

[4] Jeff A Bilmes et al. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.

[5] Gary E Kopec and Mauricio Lomelin. Document image decoding approach to character template estimation. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 213–216. IEEE, 1996.

[6] Gary E Kopec and Mauricio Lomelin. Document-specific character template estimation. In *Electronic Imaging: Science & Technology*, pages 14–26. International Society for Optics and Photonics, 1996.

[7] Ayatullah Faruk Mollah, Nabamita Majumder, Subhadip Basu, and Mita Nasipuri. Design of an optical character recognition system for camera-based handheld devices. *CoRR*, abs/1109.3317, 2011.

[8] Shunji Mori, Ching Y Suen, and Kazuhiko Yamamoto. Historical review of OCR research and development. *Proceedings of the IEEE*, 80(7):1029–1058, 1992.

[9] Christopher Raphael and Jingya Wang. New approaches to optical music recognition. In *ISMIR*, pages 305–310, 2011.

[10] G. Read. *Music Notation: A Manual of Modern Practice.*

[11] Zheng Song, Qiang Chen, Zhongyang Huang, Yang Hua, and Shuicheng Yan. Contextualizing object detection and classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1585–1592. IEEE, 2011.

[12] Verena Thomas, Christian Wagner, and Michael Clausen. OCR based post processing of OMR for the recovery of transposing instruments in complex orchestral scores. In *Proceedings of the 12th International Society for Music Information Retrieval*, pages 411–416, 2011.

[13] Frank Wessel and Hermann Ney. Unsupervised training of acoustic models for large vocabulary continuous speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 13(1):23–31, 2005.