USING THE GAMERA FRAMEWORK FOR BUILDING A LUTE TABLATURE RECOGNITION SYSTEM

Christoph Dalitz **Thomas Karsten**
Niederrhein University of Applied Sciences
Reinarzstr. 49, 47805 Krefeld, Germany
christoph.dalitz@hsnr.de, thomas.karsten@gmx.de

ABSTRACT

In this article we describe an optical recognition system for historic lute tablature prints that we have built with the aid of the Gamera toolkit for document analysis and recognition. We give recognition rates for various historic sources and show that our system works quite well on printed tablature sources using movable types. For engraved and manuscript sources, we discuss some principal current limitations of our system and Gamera.

Keywords: Optical Music Recognition, Lute Tablature.

1 LUTE TABLATURE

From the 16th and early 17th century a large body of lute tablature sources is extant. As a major part of this music is derived from vocal models, it can be an ideal investigation object for music information retrieval questions. Consequently there are efforts like the ECOLM project [1] to build a data base of machine readable tablature encodings of lute music sources.

Usual optical music recognition (OMR) systems designed for common music notation (CMN) cannot be used for this purpose because lute music is written in tablature rather than CMN. Figure 1 shows the characteristics of lute tablature notation: rather than specifying the sound of the music, it specifies when and in which frets the strings of the instrument are stopped.

The symbols used for fret and rhythm had not been standardized, but almost every historic source used its own unique set of symbols. This is an important difference to CMN, which consists of a limited set of symbols which are consistent across different music scores. Consequently a system for optical tablature recognition (OTR) must not be designed to work with a single set of a priori known symbols, but to be adaptable to differing tablature symbols. We shall see below that the conception of training in

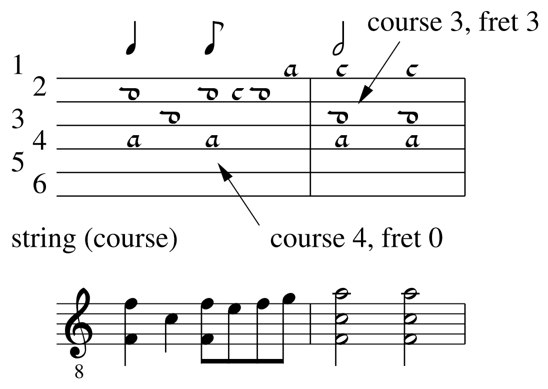

Figure 1: Lute tablature sample and music transcription

statistical pattern recognition provides an adequate framework for this.

Besides the above problem, the complexity of recognizing a particular source depends on two factors: how much the individual symbols vary within the source and whether symbols overlap and intersect. While the former can hinder the identification of individual symbols, the latter poses problems on their proper segmentation. In our present work we only consider sources printed with movable types because in these the symbol variance is limited to random defects and symbols are not overlapping.

2 THE GAMERA FRAMEWORK

The Gamera framework [2] has been developed by M. Droettboom et al. as a flexible toolkit for building recognition systems. Gamera essentially is a library for the *Python* programming language and has a number of distinctive features:

- Gamera already provides functions for image segmentation (projections, connected component analysis) and classification (kNN) and it has a classifier training interface
- all methods can be combined flexibly because they are provided as python modules. Own C++ functions for image processing can be added as plugins
- platform independence, which has let us develop parallel on both Linux and MacOS X without noticeable differences

- Gamera's source code is freely available under the GNU general public license

The last point is particularly important for research projects, because it allows for full access to the underlying techniques and enables other researchers to participate in the development of Gamera. Consequently we have also made the full source code of our system freely available as a Gamera toolkit from our project homepage [3].

3 OUR RECOGNITION SYSTEM

The recognition process in our system consists of the four steps *preprocessing, segmentation, classification* and *postprocessing*. Table 1 provides an overview over the individual operations and shows what we could use from Gamera and what we had to implement ourselves. The following sections describe the individual steps in more detail.

Table 1: Operations used from Gamera and implemented by ourselves

Preprocessing:	
rotation correction	own implementation (now in Gamera)
smoothing	Gamera (convolution)
staff line removal	own implementation
Segmentation:	
symbol isolation	Gamera (CC analysis)
Classification:	
heuristic	own implementation
statistic	Gamera (kNN, grouping)
Postprocessing:	
semantic interpretation	own implementation
tablature coding	own implementation (abc) (midi with abc2midi, postscript with abctab2ps)

3.1 Preprocessing and Segmentation

In order to be independent of the scanning resolution we first determine the characteristic dimensions *staffline_height* (staff line thickness) and *staffspace_height* (distance between adjacent staff lines) as the most frequent black and white vertical runlengths [4].

When scanning a tablature book the image is usually slightly rotated. Such a rotation has a negative effect on both the detection of staff lines and the classification of the tablature symbols and hence should be corrected, which we have done with Postl's projection profile method [5]. Although a naive implementation is rather slow, we have obtained reasonable performance by scaling the image down to a *staffline_height* of two pixels and by searching the maximum with a golden section search [6].

The image quality of most historic lute tablatures is far from ideal. Typical defects are random black speckles and broken symbols. We identify black speckles as connected components with a small black area. Small white holes within tablature symbols or staff lines can be filled with a convolution operation. We have chosen a kernel width equal to staffline height and a height half as high. This rectangular shape has the effect that stafflines become better connected.

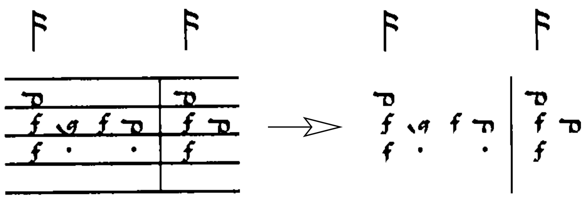
Figure 2: Staff line removal

While the above operations merely improve the image quality and are optional, there is one final obligate preprocessing operation: staff line removal. Figure 2 shows that the symbols are only separable with a connected component analysis after the staff lines have been removed, because otherwise all symbols are connected by staff lines.

Our removal method is based on Fujinaga [4] with two major modifications: we have omitted the line rectifying shearing step and have added an option to remove a staff line section even when it touches a symbol. The latter is used for fret letters *between* the lines. When they are *on* the lines we simply keep vertical runlengths longer than two times *staffline_height*. Although this distorts the line crossing part of symbols considerably, the effect on symbol classification is less negative than one might expect, because the distorted symbols are also used for training.

Before the staff lines are removed, we remember their position, which we find via an analysis of the horizontal projection profile. Although this only yields a single row value per staff line, it is sufficient because the vertical variation of a staff line after the skew correction is considerably less than *staffspace_height*.

3.2 Symbol Classification

Once the tablature symbols are isolated they are passed one by one to the classifier which assigns each symbol to a specific class like "eighth flag", "bar line", "fret number zero". For all symbols (except bar lines) we use a statistical classification method, which means that the feature distribution of the symbols needs to be trained before recognition. This makes the system adaptable to a wide range of tablature prints because no a priori assumptions about the actual form of the individual symbols are made.

For classification we use the *k nearest neighbor* (kNN) method [8] in combination with Gamera's grouping algorithm [7], which cares for disjoined symbols like the letter "g" in figure 2. As this grouping algorithm requires the group to be pretty close to a glyph from the training data, it did not work well with bar lines, which are often broken in historic tablature prints. Hence we classify bar lines heuristically based upon their aspect ratio, width and total height of a group of bar line fragments.

The kNN classifier recognition rate strongly depends on the chosen set of features. Of Gamera's 15 built in features (see [2] for their description) 14 are useful for segmentation based classification, resulting in 2^{14} possible feature combinations. We have used Morlaye's tablature book from 1552 with a training set of 3303 training symbols and 1403 test symbols (four pages with a total of 16 staves) to measure the recognition rate for all possible feature sets. The results for single features are given in table 2. Note that some features are vector values; hence the dimension is also given in the second column.

Table 2: Recognition rates for individual features

Feature	*Dimension*	*Recognition Rate*
area	1	70.4%
aspect ratio	1	74.5%
black area	1	33.9%
compactness	1	13.2%
moments	9	98.0%
ncols	1	52.8%
nholes	2	67.3%
nholes extended	8	87.0%
nrows	1	58.2%
skeleton features	5	70.5%
volume	1	42.8%
volume16regions	16	96.6%
volume64regions	64	97.3%
zernike moments	26	96.7%

For feature combinations, figure 3 shows how the recognition rate depends on the feature set size. For all sets of the same size the highest recognition rate value is displayed. Our results nicely illustrate the point that the addition of features does not necessarily increase the recognition rate.

The highest recognition rates were achieved by several feature combinations followed closely by other combinations. Hence we have based our decision for a particular feature set not on the recognition rate alone, but also on the complexity of the feature computation, and settled on the combination *aspect ratio, moments, nrows, volume64regions* (measured recognition rate: 99.2%).

Another parameter that needs to be chosen in kNN classification is the number k of neighbors taken into account. According to theory [8], k should be considerably smaller than the smallest class population among the training samples. Moreover the class population ratios in the training set should be representative for the original data, which implicitly takes the a priori probabilities into account. These constraints have the effect that k could be hardly larger than one, because there are some comparably rare symbols in lute tablature like high fret numbers or very long or very short rhythm values.

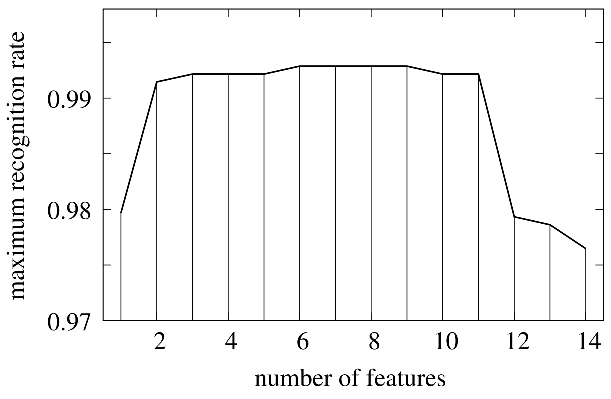

Figure 3: Maximum recognition rate for different feature set sizes

3.3 Postprocessing

After the individual symbols are recognized, their musical semantics needs to be determined. Essentially this means to organize the symbols as a linear sequence of *chords*. Here “chord” does not only mean “set of simultaneous notes”, but any set of symbols which belong together at a single point on the time axis. For instance a “chord” can be a bar line, a time signature or an actual chord of notes together with its rhythm value.

In order to establish this chord sequence, we assign each symbol a staff and course number based upon the staff line positions determined during the preprocessing phase. Then all symbols within the same staff can be grouped into chords. A natural grouping criterion is whether symbols overlap horizontally. In order not to miss small symbols like dots we do not use an absolute overlapping threshold, but a threshold for the ratio of overlap and symbol width. When using this criterion it is necessary to first ignore wide symbols that span more than one chord and take care of them after grouping. The threshold whether a symbol is “wide” can be based upon *staff-space_height* or the median symbol width.

Once the chord sequence is established, the tablature code can be generated. Unfortunately there is currently no widely accepted open lute tablature encoding format.

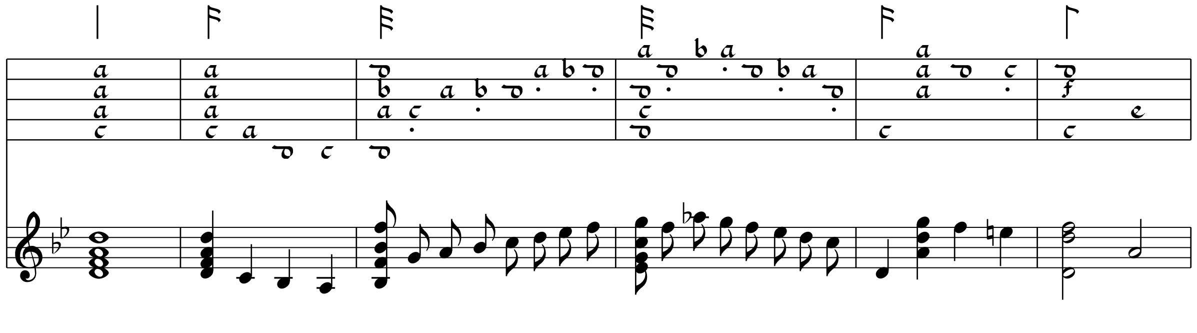

Figure 4: Sample result of our OTR system

ECOLM uses T. Crawford's *TabCode* [1], but apart from the internal ECOLM software there is currently no software available that can read this format. Hence we have used C. Dalitz' tablature extension of the *abc* format, for which free software is available [9]. It should be noted however that the actual tablature encoding does not matter, because conversion programs could be easily written or our OTR system could be easily extended to produce other codes.

We also generate a music transcription in a "literal" rather than an "interpretative" way [10]. Figure 4 shows the OTR result after it has been processed with the program *abctab2ps*. The key signature is automatically chosen in such a way that the number of accidentals is minimized. In figure 4 this leads to an E flat, because there are two flat E's, but only one natural E.

3.4 Performance and Limitations

We have tested our system with several sources using different tablature types (both "french" and "italian" lute tablature) and obtained the error rates given in table 3. Note that this is not just the error rate for classifying individual symbols but also includes errors introduced by our postprocessing operations.

Table 3: Error rates of our system for various historic tablature prints (for details about the sources see [11])

Source	*Training Symbols*	*Test Symbols*	*Error Rate*
Morlaye 1552a	3303	1673	0.6%
Denss 1594	3390	1827	3.6%
Francisque 1600	2959	1645	4.1%
Spinacino 1507a	3023	2286	1.8%
Giovanni Maria da Crema 1546a	2275	1850	3.8%
Casteliono 1536	1577	1713	3.9%

Most of the individual recognition errors can be traced down to the following reasons:

- artifacts from the staff line removal procedure (da Crema, Casteliono)
- errors in bar line recognition (Francisque)
- wrong chord grouping
- touching symbols (Denss)

In order to improve the staff line removal, we have begun to investigate different staff line removal techniques suggested in the literature. The bar line recognition might be improved by a semi-heuristic approach: first vertical line fragments could be classified statistically and among these groups forming bar lines could be searched.

The last problem of touching symbols however is more fundamental. It is due to our approach of a symbol agnostic segmentation via a connected component analysis. Although Gamera can deal to a certain extent with over-segmented images [7], it currently cannot cope with under-segmented symbols. Consequently our system is not applicable to manuscripts or engravings, which typically do not only have touching symbols but also a lot of intersecting symbols.

4 CONCLUSION

According to the Gamera homepage [2], Gamera is a tool for "domain experts, who have a strong knowledge of the documents in a collection, but may not have a formal technical background". This statement should be taken with a grain of salt. It holds as long as everything can be achieved with methods that Gamera provides out of the box, because Python is an easy to learn language particularly for people without a training in computer science. The addition of custom plugins however requires some knowledge of image processing techniques and advanced C++ programming concepts.

Thus we consider Gamera ideal for "domain experts" who also have some technical background. That we have implemented our OTR system within the Gamera framework has reduced our development time considerably. Actually we consider our OTR system as a demonstration that Gamera's modular and extensible conception is successful.

REFERENCES

[1] G. Wiggins, T. Crawford, M. Gale, D. Lewis: "An Electronic Corpus of Lute Music (ECOLM)". *http://www.ecolm.org/*

[2] M. Droettboom: "The Gamera Project Homepage". *http://gamera.sourceforge.net/*

[3] Homepage of our OTR project: *http://lionel.kr.hsnr.de/~dalitz/data/projekte/otr/*

[4] I. Fujinaga: "Staff Detection and Removal". In S. George (editor): *Visual Perception of Music Notation*, pp. 1-39, Idea Group, 2004

[5] W. Postl: "Detection of linear oblique structures and skew scan in digitized documents". *Proc. 8th Int. Conf. on Pattern Recognition*, pp. 687-689, 1986

[6] W.H. press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling: "Numerical Recipes in Pascal". *Cambridge University Press*, 1993

[7] M. Droettboom: "Correcting broken characters in the recognition of historical printed documents". *Joint Conference on Digital Libraries*, pp. 364-366, 2003

[8] B.V. Dasarathy: "Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques". *IEEE Computer Society Press*, 1991

[9] C. Dalitz: "abctab2ps". *http://www.lautengesellschaft.de/cdmm/*

[10] M. Orphée: "A History of Transcriptions of Lute Tablature from 1679 to the Present". *The Lute, Journal of the Lute Society*, Volume XLIII, pp. 1-43, 2003

[11] Répertoire international de sources musicales (RISM), Series A/I: "Single Prints before 1800". *Bärenreiter*, 1971-1981