Optical Recognition of Psaltic Byzantine Chant Notation

Christoph Dalitz · Georgios K. Michalakis · Christine Pranzas

Submitted: 6. March 2008 / Revised: 14. October 2008

Abstract This paper describes a document recognition system for the modern neume based notation of Byzantine music. We propose algorithms for page segmentation, lyrics removal, syntactical symbol grouping and the determination of characteristic page dimensions. All algorithms are experimentally evaluated on a variety of printed books for which we also give an optimal feature set for a nearest neighbour classifier. The system is based on the Gamera framework for document image analysis. Given that we cover all aspects of the recognition process, the paper can also serve as an illustration how a recognition system for a non standard document type can be designed from scratch.

Keywords Optical Music Recognition (OMR) · Base Line Detection

1 Introduction

Byzantine music is a neume based notation system which uses a modal organisation/restructuration of melodies. The sacred music of this repertory is more commonly known as *Psaltiki* (Ψαλτική). Its notation has for long been used to describe the principal melodic line, although it can theoretically be used as well for polyphonic melodies (see [1] p. 222 for an example). This notation system has gone through many stages, the most recent one having been developed in the early 1800s in Constantinople (today known as Istanbul). As it is still in use today, we shall call it *contemporary psaltic notation* (CPN).

As psaltic music is a very small niche in today's music business, there is not yet much research done on its optical recognition. Most other approaches to early music recognition like Pugin's hidden Markov modeling [3] rely on the presence of stafflines and are thus not applicable to adiastematic neumatic notations. Barton et al. have developed an experimental OCR system for the recognition of Gregorian chant neumes within the context of the NEUMES project [4]. They give little details about the program except that it utilises neural network techniques and provide no performance evaluation, but conclude that OCR for early Gregorian chant notation is of limited practical use due the inconsistent use of symbols, which restricts shape and meaning of a symbol to a particular manuscript source [5].

This restriction does not apply to CPN, which has been standardised since about 1800. Concerning its optical recognition, there is only the pioneering work of Gezerlis who focused on the optical character recognition of individual neumes [6], but did not deal with page segmentation and layout analysis. The aim of our work is to provide algorithms not only for recognising individual neumes, but also for their syntactical grouping based on their grammatical function as well as for page layout analysis and page segmentation.

We make the source code of our system freely available [9] as a toolkit for the *Gamera* framework [7]. Gamera is not itself a recognition system, but, rather, a cross platform Python library for building custom recognition systems. It has already been used successfully not only for building recognition systems for historic music notations like renaissance lute tablature [10] and historic text documents in the Navajo language [11] or early modern Latin [12], but also for building a seg-

C. Dalitz and C. Pranzas
Hochschule Niederrhein, Fachbereich Elektrotechnik und Informatik, Reinarzstr. 49, 47805 Krefeld, Germany

G.K. Michalakis
Université de Poitiers, Faculté de Médecine, Service de Médecine Interne, 2 rue de la Milétrie, 86021 Poitiers Cedex, France

Ref. No.	*Title*	*Year*	*Editor*	*Pages*
HA-1825	Heirmologion Argon	1825	Chourmouzios the Chartophylax	300
HS-1825	Heirmologion Syntomon	1825	Chourmouzios the Chartophylax	240
AM-1847	Anastasimatarion	1847	Theodoros Papa Paraschou of Phoka	350
MP1-1850	Mousike Pandekti, Volume 1	1850	Teachers of the “Mousike Bibliotheke” collection	430
PPAM-1952	Patriarchiki Phorminx: Anastasimatarion	1952	Constantinos Pringos	200
PPD-1969	Patriarchiki Phorminx: Doxastarion	1969	Constantinos Pringos	350

Table 1 Prints of Psaltic music that appeared in Constantinople and to which we have applied our recognition system. The numbers are given for further reference in our text.

mentation evaluation framework for staff removal from music images [13].

This paper is organised as follows: section 2 gives an overview of the music notation and section 3 describes all steps of our recognition system. All algorithms are described and evaluated on sample pages from a variety of printed books. In section 4, we present a summary of the main ideas and experimental results, and in the final section we make some critical comments and suggest starting points for future improvements.

2 The Notational System for Psaltiki

This notation is described quite extensively in the original literature [1]; for an introduction in English see [2]. The particularity of CPN is that only the skeleton of the melodic line is written out according to well defined orthography rules. During performance more notes (*embellishments*) are added, which requires considerable training beside a competent master. More recent editions by 20th century composers extend the orthography rules so as to write out melodies in more detail, yet always using the same CPN neumes. There are hundreds of manuscripts and post 1800 classic editions, some of which are listed in Table 1. Twentieth century editions including written out embellishments are just as numerous, as they describe the same repertoire as the classic 1800s editions.

Some characteristics of CPN can be seen in Fig. 1 which shows musical neumes accompanied by Greek lyrics below. Unlike in common western music notation, there is no staff system for specifying absolute pitches, and melodic formulae are encoded using specific symbols (*neumes*). These convey information that may be classified as quantitative (relative pitch), qualitative (melismatic vocalisations), temporal (dividing and extending neume durations), modulative (fthora and chroa, indicating modulation from one type of tri-, tetra- or pentachord scale to another), intonative (giving information as to the mode and musical gender used: diatonic, chromatic or enharmonic), martyric (giving “witness” attestations as to the relative pitch and mode after several lines of neumes), metric (indicating the type of temporal counting), rhythmic (with diastoles and numbers indicating rhythmic changes), chronagogic (tempo) and, more recently, isokratematic (indicating the relative pitch of a second or even third voice).

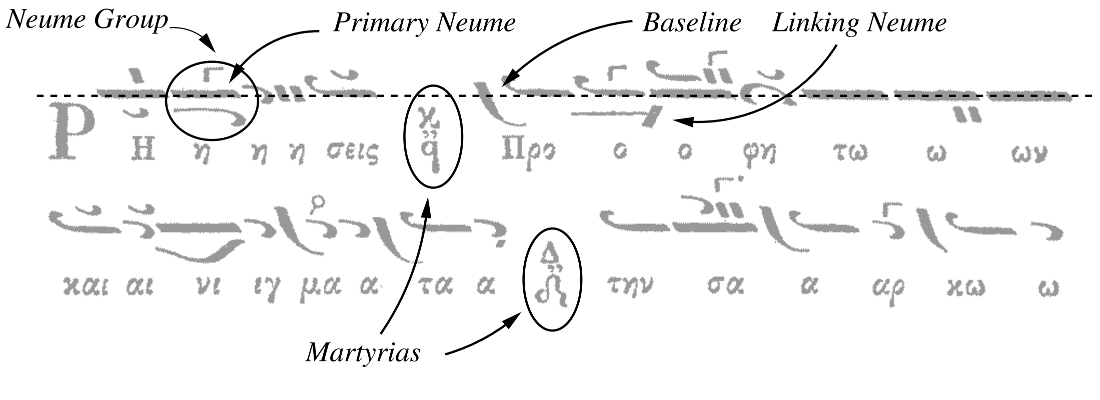

Neume Group
Primary Neume
Baseline
Linking Neume
Martyrias
Ρ Η η η η σεις Προ ο ο φη τω ω ων
και αι νι ιγ μα α τα α την σα α αρ χω ω

Fig. 1 Example of two lines of Psaltic notation using extended formular melodies (from HA-1825). The first symbol below the baseline is the capital letter “rho”, which is not a musical neume, but belongs to the lyrics.

Table 2: All individual neumes and their functions (P = can be primary, B = defines baseline, L = linking, C = chronos neume, M = always a martyria scale neume, m = can be a martyria scale neume)

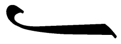
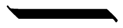
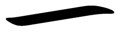
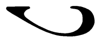
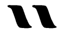
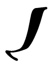
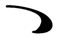
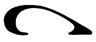
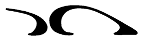
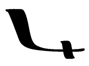
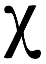
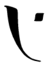
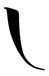
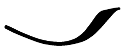
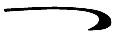
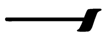
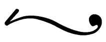
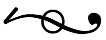
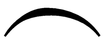
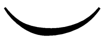
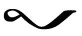
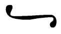
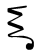
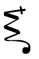
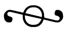
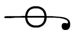
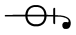
No	*Neume*	*Name*		*No*	*Neume*	*Name*	
1		Ison	PB	2		Oligon	PB
3		Oxeia	P	4		Petasti	P
5		Kendima		6		Kendimata	P
7		Hypsili		8		Apostrophos	P
9		Elaphron	P	10		Syneches-Elaphron	P
11		Hyporrhoe	P	12		Hamili	P
13		Mono-Hemi-Gorgon		14		Mono-Gorgon	
15		Di-Hemi-Gorgon		16		Di-Gorgon	
17		Klasma		18		Hapli (thick dot)	
19		Stigmi (small dot)		20		Koronis	
21		Argon		22		Hemi-Olion	
23		Di-Argon		24		Argon-Gorgon	
25		Chi	C	26		Leima Chronou	P
27		Stavros		28		Comma	
29		Bareia		30		Psiphiston	
31		Anatinagma		32		Omalon	L
33		Heteron	L	34		Endophonon	L
35		Hyphen-Ano	L	36		Hyphen-Kato	L
37		Diesis		38		Diesis-Monogrammos	
39		Diesis-Digrammos		40		Hyphesis	

Table 2: (continued)

No	*Neume*	*Name*		*No*	*Neume*	*Name*	
41		Hyphesis-Monogrammos		42		Hyphesis-Digrammos	
43		Diatonic-Hypo	M	44		Diatonic-Hemi-Phi	M
45		Diatonic-Lamda	M	46		Diatonic-Na	M
47		Diatonic-Delta	M	48		Chromatic-Large-Interval	M
49		Chromatic-Soft-Large-Interval	m	50		Chromatic-Hard-Large-Interval	m
51		Chroa-Chromatic-Zygos	m	52		Diatonic-Ni-Kato	
53		Diatonic-Pa		54		Diatonic-Bou	
55		Diatonic-Ga		56		Diatonic-Di	
57		Diatonic-Ke		58		Diatonic-Zo	
59		Diatonic-Ni-Ano		60		Fthora-Chromatic-Hard-Small-Interval	
61		Fthora-Chromatic-Soft-Small-Interval		62		Chroa-Enharmonic-Kliton	
63		Enharmonic-Zo		64		Enharmonic-Diarkes-Hyphesis	
65		Enharmonic-Diarkes-Diesis		66		Chroa-Enharmonic-Spathi	
67		Diastole	P	68		Diastolic Hyphen-Ano	P
69		Diastolic Hyphen-Kato	P	70	0	Int-0	P
71	I	Int-1	P	72	—	Pavla	
73	Additionally the following text characters can occur as neumes: – the lower case Greek letters π ϐ ϒ δ χ ς ν – the upper case Greek letters Π Β Γ Δ Κ Ζ Ν Ι Μ – the arabic numerals 1 to 9 (“Diastole numbers”) – paranthesis and square brackets						

There are about 100 different individual neumes, which can be combined to form new *neume groups*. In each group there is one *primary neume*, which typically lies upon the baseline; all the other neumes in the same group are considered as *secondary* neumes. Some neumes can never be primary, while others can be either primary or secondary, depending on their relative position.

All neumes belong to at least one neume group, which can be classified as an "ordinary", *martyria* or *chronagogic* group. Even though "ordinary" groups actually can be further classified into melodic, pause, rhythmic and metric neume groups, this distinction is of no significance with respect to the neume grouping algorithm described in Sect. 3.5. While all neume groups are independent of each other, there is a set of neumes called *linking neumes* which may span over several neume groups (typically not more than three) and connect them. Such linking neumes can be occasionally broken in classical editions due to line endings for justification reasons.

Martyria groups ("witnesses") consist of at least two components: a Greek letter representing the note name, and a martyria "appendix", representative of the scale and overall context within which the particular note evolves. These two constituents specify the relative pitch (with respect to the starting point of the melody). Depending on the edition, the uppermost of the two symbols may be found on or below the baseline. Yet, if other symbols are added as well (such as diastoles and fthoras), the entire martyria group components may span both above and below the baseline. The particularity of martyrias lies in the fact that they extend into the lyrics text zone, therefore creating a special segmentation problem in separating lyrics from martyrias. The same applies to *chronagogic* (tempo) groups, which typically consist of the neume "Chi" plus Gorgons and Argons.

Table 2 lists all neumes and their possible functions. Some of these are Greek letters that can also appear in lyrics lines. Several neumes can vary in width (Ison, Anatinagma) or height (Diastole). All neumes marked as P ("can be primary") are primary neumes when they intersect the baseline, with the notable exception of Kendima and Bareia. A single Kendima is never primary, but belongs to the group before it, while a group of two Kendimata on the baseline is primary. A Bareia usually is a secondary neume belonging to the group to its right, except when it is followed by one or more dots, in which case it is a primary neume with the dots (Hapli or Stigmi) belonging to its neume group.

The peculiarity of CPN that secondary symbols can be attached to the left, right, top or bottom of other primary symbols lying on a baseline shows some similarity to the "matras" in Hindi script that can be attached to basic characters resulting in modified characters which in turn can be combined into words [14]. The role of the baseline in CPN corresponds to the header line in Hindi script, with the notable difference that the CPN baseline is imaginary (i.e. invisible), while the header line in Hindi script is explicitly visible as an integral part of the main characters. Concerning layout analysis, an important difference is that in Hindi script the groups (words) are easily identified as connected components while the parts need to be determined by some segmentation method [15]. In CPN, on the other hand, it is the parts that are easily detectable as connected components while the groups need to be determined based on syntactic rules and class membership. Thus Hindi script typically requires a top-down approach as opposed to a bottom-up approach in CPN.

Psaltiki associates different melodic patterns to text according to the distribution of the accentuated syllables. All this information constitues sequences that can be encoded, classified, and searched much like biological gene sequences and linguistic patterns that are used in the transmission of memory: this forms an interesting area of research for musical pattern analysis[1]. Furthermore, its relationship to the Gregorian and Roman chant repertories is an interesting area of research for modern techniques of music information retrieval. In order to build a database of Psaltic chant in a machine readable format that can be used for such comparative investigations, as well as for building a repository of traditionally authentic formulae, an optical recognition system for this type of notation would be of great help.

3 The Recognition System

Like most other document recognition systems, our recognition system sequentially performs the five steps *preprocessing, segmentation, classification, neume layout analysis* and *output generation.* The task of the individual steps is:

1. During *preprocessing*, image defects due to low printing or scan quality (rotation, noise) are improved. Moreover, characteristic dimensions are determined from the image; these can be utilised to make subsequent steps independent from the scanning resolution.

[1] G.K. Michalakis: *Le formulisme dans la transmission de la mémoire de la psaltique et du chant grégorien: une approche par la biologie moléculaire.* Master Thesis, University of Poitiers, France (in preparation)

2. In the *segmentation* step, the individual symbols are isolated, the page is segmented into text (lyrics) and neume lines, and the text is removed.
3. In the *classification* step, the individual neumes are recognised. This step assigns each neume a class label.
4. In the *neume layout analysis*, the mutual relationship of the individual symbols is determined and they are grouped, based on their class names and relative positions.
5. Eventually, a machine readable *output encoding* is generated.

In the subsequent sections we describe these steps in detail and report their performance on the prints from Table 1.

3.1 Preprocessing and Symbol Segmentation

As our primary method for detecting neume baselines and lyrics textlines uses horizontal projections (see section 3.3 below) it is important that a skew angle introduced through scanning be corrected. This was achieved with Postl's projection profile method [16] which already has proven to be quite reliable for lute tablatures [10]. The method determines the rotation angle α as the angle with the highest variation of the skewed projection profile

$$h_\alpha(y) = \sum_{x=-\infty}^{x=\infty} f(x\,\cos\alpha - y\,\sin\alpha, x\,\sin\alpha + y\,\cos\alpha)$$

where $f(x, y)$ is the document image pixel value at position $(round(x), round(y))$ and zero outside the document. The variation of this profile is defined as

$$V(\alpha) = ||{h_\alpha}'||^2 = \sum_{y=-\infty}^{y=\infty} [h_\alpha(y+1) - h_\alpha(y)]^2$$

As a naive brute force search for the angle α that maximises $V(\alpha)$ would be rather slow, we did a brute force search for the angle only at a coarse angle resolution and then used the three points around the maximum among these values as a starting point for a golden section maximum search [17].

To improve the image quality, we removed noise consisting of white and black speckles. White speckles were typically small enough in our images to be removed with a median filter using a 3x3 window [18], which, for onebit images, is incidentally the same as an averaging filter. Most black speckles, however, were too large to be erased by the median filter and we identified and removed them instead as connected components (CCs) having a "small" black area. Ideally, "small" would mean "small with respect to the characteristic page dimension *oligon_height*" (see below). Unfortunately this dimension can only be detected reliably *after* despeckling because, when speckles are present, they can be so frequent as to dominate the runlength histogram. Hence, we used a hard coded speckle size of three black pixels.

As all symbols in CPN are well separated and usually do not touch, individual symbols can be isolated using a connected component (CC) extraction [19].

3.2 Characteristic Dimensions

To make all subsequent operations independent from the scanning resolution, we determined two characteristic dimensions for each page: *oligon_height*, which corresponds to the vertical stroke thickness of the wide, frequently encountered neume Oligon, and *oligon_width*, which corresponds to the horizontal width of this same neume.

In many diagram recognition problems, the stroke thickness can be measured from the histogram of black runlengths. For example, the staffline height in common western music notation corresponds to the most frequent black vertical runlength [13]. In the case of CPN however, this histogram is dominated by thinner strokes from lyrics, noise and different neumes (see Fig. 2). As the most characteristic feature of the Oligon is that it is significantly wider than high, we created filtered images in which CCs with a ratio width/height less than three had been removed. This filtering is independent of the scanning resolution because the aspect ratio is scale invariant.

The neume distribution among the remaining wide CCs is shown in Table 3: the most frequent wide neume is the Oligon, followed by the Ison. Both neumes together form the majority of all wide CCs on each page.

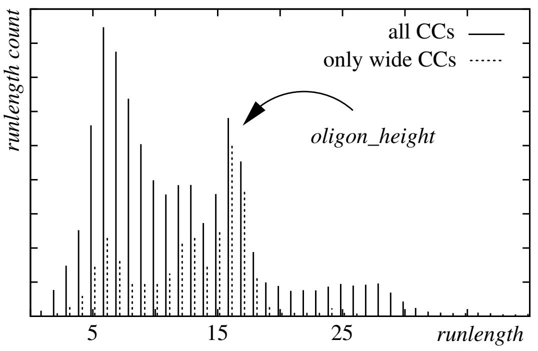


Fig. 2 Black vertical runlength histogram for a complete CPN image (solid) and the same image with all CCs with $width/height < 3$ removed (dashed).

Source	HA-1825	HS-1825	AM-1847	MP1-1850	PPAM-1952	PPD-1969
oligon_height	15.8 ± 0.4	16.7 ± 0.7	13.4 ± 1.0	17.5 ± 1.0	13.0 ± 0.5	15.0 ± 0.0
oligon_width	132.8 ± 1.1	136.4 ± 7.0	122.0 ± 2.6	152.5 ± 2.5	125.1 ± 2.5	163.3 ± 3.5

Table 4 Averages and standard deviations of the characteristic dimensions as measured with our algorithm on 25-30 pages from each book.

Neume	*Minimum*	*Maximum*	*Mean ± Stddev*
number of wide CCs per page	76	127	104 ± 14
oligon	36.5%	68.4%	53.5% ± 7.9%
ison	2.6%	45.9%	23.9% ± 9.9%
psiphiston	1.9%	21.1%	10.0% ± 4.2%
other	0.0%	15.6%	6.4% ± 3.4%
anatinagma	0.0%	11.4%	4.8% ± 3.6%
omalon	0.0%	9.7%	1.5% ± 2.4%
ison or oligon	63.1%	86.7%	77.3% ± 6.0%

Table 3 Per-page distribution of CCs with *width/height* ≥ 3.0 counted on 48 pages from the books listed in Table 1. Note that the minimum percentages of different neumes usually occur on different pages, so that all percentages in the "Minimum" column do not add up to 100. The same applies to the "Maximum" column.

As both the width and the vertical stroke thickness of Oligon and Ison are comparable, we can determine the characteristic dimensions from the filtered image as follows:

– *oligon_height* is the most frequent black vertical runlength (see Fig. 2)
– *oligon_width* is the median of the CC width

These values turned out to be quite stable in our experiments over different pages, as can be seen from the low standard deviations in Table 4: even the largest mean error for *oligon_width* in source HS-1825 is only about 5%. For all other sources the variances are much smaller. The robustness of these two values makes them appropriate base units for thresholds used in subsequent rule based decisions.

3.3 Page Segmentation

The page segmentation step consists of the following tasks, which we describe in detail in the corresponding subsections:

– detection of the baselines around which the neumes are grouped
– detection of the text (lyrics) lines between the baselines
– lyrics removal

3.3.1 Page Layout Analysis and Lyrics Removal

Neume baselines are the lines around which the frequent neumes Oligon and Ison are aggregated. Consequently, they can be detected by an analysis of the horizontal projection profile of the image containing CCs with a width/height ratio greater than three (see Fig. 3), because this projection profile is dominated by Isons and Oligons, as we have shown in the preceding section. Baselines correspond to maxima in the projection profile with a height greater than 0.8 times *oligon_width*. As this criterion can yield more maxima than corresponding baselines, we first applied a low-pass filter of width *oligon_height* to the projection profile. For each projection value greater than 0.8 times *oligon_width* found at height y, we only selected the largest maximum within a window $[y, y+oligon\_width)$. As an additional constraint we demanded that the distance between two baselines be larger than one *oligon_width*. This threshold is based on the reasoning that baselines cannot be closer due to the height of neume groups and due to the lyrics line between adjacent baselines.

While searching for a maximum in the projection profile of the unfiltered image, textlines can be found between two baselines, close to the middle. Due to the characteristic shapes of the Greek characters, the largest maximum will always be at the upper or lower edge of the lower case letters. To make our textline more robust with respect to curvature, we interpolated between the two largest maxima near the centre between adjacent baselines.

The algorithm described above yields a single y-position for each baseline and textline. This implies that the image is not too strongly rotated or curved. Although we have found this condition to be met after

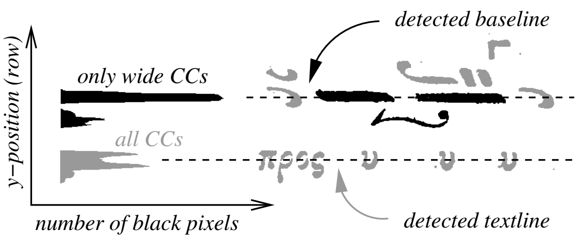


Fig. 3 Neume baselines correspond to maxima in the projection profile of only the *wide* CCs (black). Textlines can be detected from maxima in the projection profile of *all* CCs (black and grey).

applying the rotation correction (described in Sect. 3.1) in the prints on which we have worked (see Table 1), it should be noted that this does not hold in general, in particular when manuscripts are considered.

The simplest approach for lyrics removal would be to remove all CCs that cross the textline. This would, however, also remove part of martyria and chronos signs, both of which contain components that overlap with lyrics lines, as can be seen in Fig. 1. To distinguish martyrias from lyrics we tested two different methods, one based on a trained classifier, and the other based on pre-defined rules.

The training based approach requires that lyric glyphs be trained as "lyrics" and that all CCs on the image be classified (see section 3.4). As some of these glyphs can also be part of neume groups, we cannot simply remove all glyphs recognised as "lyrics", but must first look for glyphs recognised as "martyria" (or "martyria-fthora" or "chronos"). Each glyph that touches the textline and is not itself a "martyria" and is not below or above a glyph recognised as a martyria is considered as being part of the lyrics and is removed.

In the rule based approach, we first determined the lyrics character height (*character_height*) as the median height of all glyphs touching the textlines. All glyphs touching the textline were removed, unless they met one of the following criteria:

- there is no glyph on the baseline above
- the glyph's upper edge rises above the baseline more than 1.5 * *character_height*
- the glyph has a width/height ratio greater than 2.2

The last two criteria avoid that two types of neumes, that frequently extend into the lyrics region, are inadvertantly removed: the second criterion is for Bareias which generally cross the baseline and the third criterion is for linking neumes which can be distinguished from Greek characters by their width (see Table 2).

Theoretically, lyrics always need neumes on the baseline above, so that glyphs meeting the first criterion could not be lyrics. In our sources however, lyrics were often not well aligned with the neumes, and this required an additional criterion. We therefore utilised the fact that martyria groups always consist of two vertically stacked components (see Fig. 1); the same holds for chronos groups. Consequently, we only consider a neume meeting the first criterion a martyria or chronos neume when a second component is found above it with the following properties:

- It is narrower than 0.75 * *oligon_width*. This rules out wide secondary neumes, which tend to extend beyond the primary neume due to their width.
- It is less than a vertical distance of 1.5 * *character_height* above. Neumes too far apart are not perceived as a connected group by a reader and thus are unlikely to be meant as group.
- The total height of both glyphs is greater than 2 * *character_height*. This is necessary to avoid confusion with broken lyrics characters and noise.

The numerical threshold values have been chosen heuristically so that a number of common decision errors on selected pages from the different prints could be minimised.

In our experiments described in the next section, the rule based approach was slightly better, though not significantly so. This does not mean however, that a training based approach generally performs poorer. It may as well be due to insufficient training data. Concerning deterministic approaches to lyrics removal, the adaption of other sophisticated page layout analysis algorithms originally developed for text documents might be a potentially promising area of future research [20].

3.3.2 Results

We tested the baseline and textline detection algorithm on 65 random pages from the six prints listed in Table 1. From a total of 764 baselines only 2 were missed and no non-existent baseline was falsely found. For each detected baseline the corresponding textline was correctly identified.

That baselines were missed was due to a systematic error that occurs when a baseline does not contain any ison or oligon at all, in which case no neumes from that line remain after filtering the wide CCs before baseline detection. This can occur when a melodic line is only partially filled with a melodic formula that coincidentally contains no ison or oligon.

To compare the quality of our alternative algorithms for lyrics removal, we first manually removed the lyrics from 10 random pages for each source of Table 1, resulting in a total of 60 test pages. For both algorithms (training and rule based, respectively) we counted the number of non-removed connected components (CCs) that were lyrics ("missed" CCs) and the number of falsely removed CCs that were not lyrics ("excess" CCs). As the images still contained considerable noise even after the preprocessing described in Sect. 3.1, we only counted CCs taller than *oligon_height*.

The results are listed in Table 5 together with the results for the simple algorithm of removing all CCs touching the textline ($\pm$*character_height*/2 to allow for slight curvature). It can be seen that the latter algorithm removes many glyphs that are not lyrics. Even though the other two algorithms introduce additional

Source	*all true lyrics CCs*	*textline touching* missed	excess	error	*training based* missed	excess	error	*rule based* missed	excess	error
HA-1825	3163	37	120	4.95%	48	5	1.68%	62	9	2.24%
HS-1825	3508	33	134	4.76%	80	11	2.59%	52	5	1.62%
AM-1847	3601	56	146	5.61%	90	32	3.39%	103	32	3.75%
MP1-1850	3117	102	99	6.45%	149	24	5.55%	155	9	5.26%
PPAM-1952	2684	8	140	5.51%	38	7	1.68%	24	11	1.30%
PPD-1969	3304	34	67	3.06%	68	11	2.39%	55	7	1.88%
Total sum	19377	270	706	5.03%	473	90	2.91%	451	73	2.70%

Table 5 Numbers of wrongly classified CCs for the simple lyrics removal algorithm that removes all CCs touching the textline, as well as for the other two more sophisticated algorithms on 10 sample pages from each source and the resulting error rates $(missed{+}excess)/all$.

errors by not removing some lyrics, they lead to a significant reduction of the error rate with the rule based approach having the fewest errors overall.

Nevertheless, when pages are compared individually, there are those for which the training based algorithm was better. To test whether the overall error rate difference is significant, we used the *statistical paired model* proposed by Mao and Kanungo [21]. For each of the n test pages ($n = 60$ in our case), we computed the difference W of the error rates between both algorithms. Under the assumption that these observations are independent for different test images, Mao and Kanungo have argued that a confidence interval for the true mean difference Δ at a given confidence level α is given by

$$\Delta \in \overline{W} \pm \frac{t_{\alpha/2,n-1}V}{\sqrt{n}}$$

where $\overline{W}$ and V^2 are the sample mean and variance of the n observed W and $t_{\alpha/2,n-1}$ is the percentile of the t distribution with $n-1$ degrees of freedom. As a condition for a statistically significant difference of the error rates at a given confidence level α, Mao and Kanungo give the following criterion

$$P_{val} = \int_{-\infty}^{-|T|} f(t)\,dt + \int_{|T|}^{\infty} f(t)\,dt < \alpha$$

where $T = \overline{W}\sqrt{n}/V$ and $f(t)$ is the probability density function of the t distribution with $n-1$ degrees of freedom.

The results of this statistical estimation for the "missed", "excess" and "total" (missed + excess) error rate are shown in Table 6. It turns out that, although our rule based approach is on average slightly better, this difference is not significant.

Error rate	*Difference*	P_{val}
missed lyrics	-0.1702 ± 0.3928	0.3894
excess lyrics	-0.0784 ± 0.1547	0.3148
missed + excess	-0.2486 ± 0.4243	0.2457

Table 6 Error rate difference between our rule based and training based lyrics removal algorithms, estimated with a confidence level $\alpha = 0.05$ in the statistical paired model. A negative difference means that the rule based algorithm is better.

3.4 Individual Neume Classification

As already shown by Gezerlis, the individual neumes can be recognised by a kNN classifier [6]. In designing the classifier, two goals need to be achieved:

- The recognition system should be adaptable to a wide range of Psaltiki sources: this requires an appropriate abstraction layer in the training process.
- The classifier error rate should be low: this strongly depends on the chosen feature set.

Both aspects are investigated in detail in the following subsections.

3.4.1 Training Abstraction Layer

The kNN classifier requires that class names be trained on sample images before the classification phase. While

Keyword	*Meaning*
`primary`	a neume that can be primary
`linking`	a linking neume
`secondaryright`	a secondary neume that always belongs to the group to its right when it appears on the baseline (normally isolated secondary neumes on the baseline are attached to the group to their left, see Fig. 4a)
`martyria`	a martyria scale neume
`martyria-fthora`	a neume that is a fthora when overlapping with a primary neume and a martyria when no primary neume is above it
`chronos`	a chronos neume
`dot`	symbol must be treated as a dot (has variable meanings)
`gorgon`	neume is a gorgon variant
`trash`	symbol can be ignored completely

Table 7 Modifier keywords of class names for training neume functions.

Source	HA-1825	AM-1847	MP1-1850	PPAM-1952	PPD-1969
number of glyphs	4081	4108	4288	6949	4375
glyphs in class "trash"	16.32%	50.17%	23.86%	63.23%	22.19%
glpyhs of six most frequent neumes	62.07%	33.64%	54.17%	24.03%	52.14%
number of classes	49	51	51	57	58
classes with < 3 glyphs	13	13	11	13	14

Table 8 Properties of the training data sets used in our kNN classifier for the different sources. In all sources the six most frequent neumes are Apostrophos, Kendima, Oligon, Ison, Mono-Gorgon and Klasma.

it were possible to only rely on the class names from Table 2 and their particular meaning in CPN as specified in [1], this would make the system very inflexible with respect to notational variants and to the introduction of additional neumes. We therefore not only trained neume names, but also neume functions (primary, linking, ...) as optional attributes. These functions are specified as a set of optional keywords during training. The supported keywords are listed in Table 7. In our implementation, the function keywords are conveyed through the class name as an optional list of dot-separated fields preceding the actual class name, e.g. `primary.oligon`.

3.4.2 Feature Selection

Gezerlis [6] used some features which are not built into Gamera (Euler number, principal axis direction, discrete wavelet transform). As reported by Gezerlis, these features were not sufficient to distinguish a number of different, but similar neumes. To tackle these confusions, he used a postclassification scheme to handle the different cases of confusion individually. On the other hand, one of the authors has observed in his work on the recognition of lute tablature prints that a selection of features built into Gamera can lead to a holdout recognition rate of over 99% [10]. Hence we have made extensive experiments with these latter features which show that they lead to a good recogniton rate for psaltic neumes as well.

For each of the sources from Table 8, we created a training data set for the kNN classifier. Source HS-1825 is missing in Table 8 because it uses the same typeface as HA-1825, so that the same training data can be used for both sources. In all training data sets, the class population ratios are representative for the sources from which they are drawn. According to Davies [22], this ensures that the *a priori* probabilities of the individual classes are correctly taken into account by a kNN classifier.

Some properties of our training data are listed in Table 8. The glyphs classified as "trash" are speckles that still remain after our preprocessing operations. Their frequency can be considered as a measure for degradations due to low print or scan quality. Each training data set only contains about one fourth of all possible symbols, because not every symbol occurs in every print and some symbols are very rare. Even among the symbols occurring in our training sets, a considerable number is represented with less than three glyphs. This has the consequence that we cannot choose the number k of neighbours in the kNN rule larger than one, leading effectively to a *nearest neighbour* classifier rather than a kNN classifier.

At the time of writing, Gamera provided 15 built in features (see the Gamera documentation [8] for details), of which the 14 features listed in Table 9 were useful for segmentation based recognition. For our recognition system, we chose the feature combination *aspect_ratio, moments, nrows, volume64regions*, because these had the best "leave-one-out" performance in the experiments described in the next section. It is interesting to note that this feature combination also had an excellent "holdout" performance on lute tablature prints [10], which leads us to the conjecture that these features generally are a good choice for printed sources.

3.4.3 Experimental Results

We evaluated the performance of the individual features on each training set with the "leave-one-out" method, i.e. by classifying each training glyph against the other training glyphs. The results are listed in Table 9, which also gives the dimension of each feature, as some features are actually vector values rather than a single value. For all features, the performance values are roughly comparable over all sources, with the notable exception of *nholes* and *nholes_extended.* These features count the number of black-white transitions per row or column and are thus very sensitive to white speckles, resulting in a poor performance on the lowest quality source PPAM-1952. The different values for the average runtime of the leave-one-out evaluation in the last column are not only due to the different feature dimensions, but also to the runtime complexity of the feature computation: e.g. the *zernike_moments* [23] have a longer runtime than *volume64regions* even though their dimension is lower.

Feature	*Dimen-sion*	*Leave-one-out performance on the training sets* HA-1825	AM-1847	MP1-1850	PPAM-1952	PPD-1969	*Average Runtime* [s]
area	1	82.7%	81.8%	76.0%	88.1%	79.5%	0.44
aspect_ratio	1	77.3%	77.0%	71.9%	83.1%	77.7%	0.41
black_area	1	68.0%	68.7%	60.1%	78.3%	58.4%	0.52
compactness	1	44.4%	56.7%	41.6%	66.4%	50.4%	1.16
moments	9	97.2%	95.2%	96.3%	97.5%	97.1%	1.79
ncols	1	72.6%	74.2%	72.2%	84.0%	67.2%	0.44
nholes	2	69.7%	77.8%	71.3%	22.4%	71.9%	0.71
nholes_extended	8	77.7%	85.2%	85.0%	28.1%	80.2%	1.54
nrows	1	63.3%	65.2%	52.8%	77.5%	46.4%	0.44
skeleton_features	5	73.3%	73.8%	65.2%	79.6%	73.2%	4.52
volume	1	61.9%	66.1%	51.6%	72.3%	57.8%	0.57
volume16regions	16	98.6%	97.0%	97.8%	98.4%	97.7%	3.11
volume64regions	64	98.6%	97.8%	98.7%	98.9%	98.3%	13.81
zernike_moments	26	97.7%	96.1%	96.1%	97.9%	97.1%	61.84

Table 9 kNN classifier performance of Gamera's individual features on the training sets from Table 8 with $k = 1$.

Feature Combination	*Dimen-sion*	*Performance on training sets* *Avg*	*Min*	*Max*
aspect_ratio, moments, nrows, volume64regions	75	99.40%	99.05%	99.70%
aspect_ratio, compactness, nrows, volume64regions	67	99.40%	99.03%	99.70%
aspect_ratio, nrows, volume64regions	66	99.40%	99.00%	99.70%
aspect_ratio, nrows, volume, volume64regions	67	99.39%	99.00%	99.70%
moments, nholes_extended, nrows, volume64regions	82	99.38%	99.03%	99.61%
aspect_ratio, nholes_extended, nrows, volume64regions	74	99.38%	99.00%	99.68%
aspect_ratio, nrows, volume16regions, volume64regions	82	99.38%	99.00%	99.68%
ncols, nholes_extended, nrows, volume64regions	74	99.37%	99.17%	99.58%

Table 10 The eight best performing feature combinations on all training sets from Table 8 with a feature set size up to four.

The best performing feature is *volume64regions* with an average recognition rate above 98%. This feature simply counts the percentage of black pixels ("volume") in each cell of an 8×8 grid. Although it is scale invariant, it is not invariant to rotation or changing stroke positions. The latter variations are less likely to be found in printed books than in manuscripts, and, by consequence, the good performance on our sources (exclusively printed books) is not surprising.

To further improve the recognition rate, we have evaluated the leave-one-out error rates for feature combinations. As brute force testing of all possible combinations is exponential in the number of features, we have only tested all combinations up to a feature set size of four, because in experiments on lute tablature prints the combination of more than four of Gamera's builtin features did not increase the recognition rate any further [10]. Table 10 lists the eight best performing feature combinations on every training set. Each of these combinations contains the individually best performing *volume64regions*. It is interesting to note that in each of the best performing combinations there is *nrows* or *ncols*, which are the height or width of a glyph. This leads us to the conclusion that the absolute size of a symbol is also an important distinguishing feature in our training sets. This is easily understandable because speckles (classified as "trash" in our training sets) can have any shape, yet are typically small.

Based on these results, we have chosen the feature combination *aspect_ratio, moments, nrows, volume64regions* for our nearest neighbour classifier, because it is the best performing combination when Table 10 is sorted by *Avg* and *Min*.

3.4.4 Compound Neumes

Some neumes in Table 2 consist of more than one connected component, some of which even have a different meaning when appearing in combination (e.g. Kendima versus Kendimata and Bareia versus Leima Chronou).

One approach would be to train the compound neumes as "groups" in Gamera and let Gamera's grouping algorithm [24] deal with them. This, however, requires that the combinations appear sufficiently often in all possible variants in the training data. Moreover, the distance between their components must not be too large, because otherwise the grouping algorithm will have to test too many possible combinations, resulting not only in a long runtime, but also in falsely detected groups.

We have therefore chosen a different approach and added a post-processing step that replaces certain neume combinations with a compound neume, based upon a translation table. Entries in the translation table are of the form

```
neume1,neume2,maxdist: neume3
```

This means that the adjacent neumes *neume1* and *neume2* following each other in the horizontal direction with a bounding box distance of at most *maxdist * oligon_height* are to be replaced with the single, pre-combined neume *neume3*. All such newly, post-processing introduced compound neumes are treated like any other neume in the subsequent layout analysis.

3.5 Neume Layout Analysis and Grouping

Once the various individual symbols have been recognised, their mutual relationship needs to be determined. Essentially this means organising the symbols as a linear sequence of neume groups. For each neume group, a primary neume must be identified. Furthermore, each linking neume must be attached to the appropriate neume groups.

3.5.1 Rules for Neume Grouping

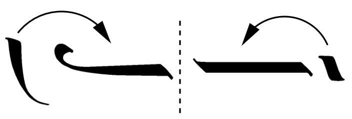
Neume groups are always separated by some space on the *baseline*. All neumes trained as `primary` and found on or near the baseline are considered as primary neumes: they form the core of a neume group. When there are two primary neumes that overlap horizontally, the larger one is considered as the primary neume.

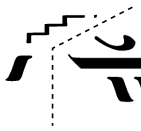
Once the primary neumes have been identified, the neume groups are built as follows:

– secondary neumes are attached to the primary neume with which they have the largest horizontal overlap
– non primary neumes on the baseline are attached to the group on the left, unless they have been trained with the keyword `secondaryright` (see Table 7)

This grouping scheme cannot be used for the neumes Gorgon (and its variants), the linking neumes and the Gorgon associated dots, because they often extend into the x-position of a neighbouring neume group (see Fig. 4 b) and c)). These neumes must therefore be ignored by the above grouping algorithm and must be post-processed as follows:

– linking neumes are always associated to the rightmost group with which they overlap horizontally. Note that this is just a simplifying assumption which looses information in the (rare) case where more than two groups are linked.
– Gorgons are associated with the leftmost neume group with which they overlap horizontally
– any dot following or preceding a Gorgon is associated with the Gorgon


(a) Non-primary neumes lying on the the baseline usually belong to the preceding group, like the Kendima in the example on the right. However, some non-primary neumes belong to the following group, just like the Bareia in the example on the left.


(b) Gorgons and associated dots may extend into the following neume group

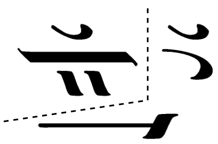

(c) Linking neumes are attached to the rightmost group with which they overlap

Fig. 4 Special cases of the general neume layout analysis based upon primary neume detection and horizontal overlaps

All neumes belonging to martyria and chronos groups fall through the grouping scheme described above because these neumes do not belong to any `primary` class and do not overlap horizontally with any primary neume. We can thus identify martyria or chronos groups by joining all neumes that overlap horizontally with neumes that are of a `martyria` or `chronos` class.

All neumes still falling through the grouping scheme (this may happen, e.g., for Diastoles) are considered as groups of their own without a primary neume.

3.5.2 Results

We have measured the error rates both for the recognition of the individual neumes and for the neume grouping on 65 pages from the sources of Table 1. On all pages we had manually removed the lyrics so that we could investigate the recognition and grouping error rates independently from errors introduced through the automated lyrics removal described in Sect. 3.3.

For the recognition of the individual neumes we used the nearest neighbour classifier with the feature set *as-*

	neume groups			*individual neumes*		
Source	*total*	*errors*	*error rate (%)*	*total*	*errors*	*error rate (%)*
HA-1825	1876	35	2.0 ± 0.6	3694	93	2.6 ± 0.5
HS-1825	1542	9	0.7 ± 0.4	2223	37	1.7 ± 0.5
AM-1847	1999	51	2.6 ± 0.7	3767	179	4.8 ± 0.7
MP1-1850	2150	61	2.9 ± 0.7	4464	99	2.3 ± 0.4
PPAM-1952	1900	30	1.7 ± 0.6	3324	108	3.3 ± 0.6
PPD-1969	2014	19	1.0 ± 0.4	3796	58	1.6 ± 0.4

Table 11 Error rates for the neume grouping and the recognition of the individual neumes. The recognition errors are the sum of misread, unread, and excess neumes. The given confidence intervals are Agresti-Coull intervals for a confidence level $\alpha = 0.05$.

pect_ratio, moments, nrows, volume64regions in combination with Gamera's grouping algorithm [24] with a maximum group size of two components, i.e., only adjacent pairs of glyphs were tested whether they "look like" a broken variant of a single connected glyph in the training set.

The results are listed in Table 11. In contrast to the *leave-one-out* error rates of Table 10, the error rates for the individual neume recognition are *holdout* error rates, i.e., they are measured on test data different from the training set. This means that the errors on the test set are independent Bernoulli trials with an error probability p for misclassifying a neume or misgrouping a group. As p is typically a low value, the classical $(1-\alpha)$ confidence interval taught in introductory statistic textbooks can be expected to have a poor coverage property, and we use the Agresti-Coull confidence interval instead, as recommended by Brown et al. [25]:

$$\tilde{p} \ \pm \ z_{1-\alpha/2} \cdot \sqrt{\frac{\tilde{p}(1-\tilde{p})}{\tilde{n}}}$$

where $\tilde{n} = n + z^2_{1-\alpha/2}$ and $\tilde{p} = (k + z^2_{1-\alpha/2}/2)/\tilde{n}$ with n being the number of neumes or groups, k the number of misclassified neumes or groups and $z_{1-\alpha/2}$ being the $(1-\alpha/2)$ percentile of the standard normal deviation. It should be noted that this confidence interval is not centred around the estimator k/n for the error rate, but around the slightly higher value $\tilde{p}$. For $\alpha = 0.05$, we have $z_{1-\alpha/2} = 1.9600 \approx 2$, so that $\tilde{p} \approx (k+2)/(n+4)$, i.e., the Agresti-Coull estimator for p adds four trials and two errors.

The holdout error rates in Table 11 are all higher than the optimistically biased leave-one-out error rates in Table 10, because the test data also contains heavily distorted, broken or touching symbols, which are absent in the training set. To examine the actual reasons for the difference in more detail, we have also counted the number of errors due to touching or broken symbols and found that

– 46 percent of the neume recognition errors and
– 26 percent of the neume grouping errors

were due to broken or touching symbols. This is an observation also made in other OCR applications, where a considerable part of the recognition errors is typically due to segmentation errors [26]. A technique commonly deployed in OCR is to post-correct the recognition results by looking for lexical or syntactic errors [32]. To estimate whether such a post-correction could also be useful for our recognition system, we have additionally counted which of the errors lead to a syntactically impossible neume combination and found that this was the case for

– almost all of the neume recognition errors and
– more than 90 percent of the neume grouping errors

Consequently, syntactical plausibility checks could automatically detect the major part of the recognition errors. The downside of such a post-processing would however be that certain notational rules had to be wired into the system, making it applicable to only a narrow range of neumatic notational conventions.

3.6 Output Encoding

Recognition of the neumatic music results in a machine readable output code. Ideally, this would be represented in the form of a well documented open file format, for which commodity software is available, comparable to *MusicXML* as a widely deployed interchange format for common western music notation [27]. MusicXML does not, however, provide any means by which to encode neumatic notations and there is no other widely accepted file format for psaltic music notation.

A development project for an XML based music encoding scheme particularly tailored to the needs of scholarly critical editions is the *Music Encoding Initiative (MEI)* [28]. While supporting common music notation out of the box, MEI also allows for the inclusion of user defined modules as extensions. Such a module has recently been developed by the *TüBingen* project to encode late medieval diastematic neumes [29]. Both this module and the MEI specification are currently under development and still a moving target.

A different file format specification is currently developed by the *NEUMES Project* [4] as a universal XML encoding scheme for medieval chant notation. It aims at covering a wide range of neumatic notations and also addresses the uncertainty problem of yet poorly understood notation. This introduces more complexity than necessary for our very limited scope of contemporary psaltic notation. Like the MEI neumes extensions, the NeumesXML specification is still under active development and thus subject to changes.

A different option is to use the file format of a graphical Psaltiki editor like *Melodos* [30]. Apart from the problem that this format is undocumented, this would also mean that the output would be useless without this particular software or on platforms for which this software is not available. This would be particularly inappropriate for storing the results in a database, because no custom third party software (e.g. for further musical analysis) could be written.

Yet another way of entering and publishing psaltic music is the use of an ordinary word processing program in combination with some special font. Ideally, the font encoding from the Unicode Standard [2] could be used, which specifies code numbers for the individual neumes, but does not cover their relative positions. Unfortunately, word processing programs are inappropriate for the two-dimensional CPN, because they are only designed for lines of characters in a one dimensional sequence. Hence two different crutches using custom fonts are in use:

– encoding neume groups using pre-combined neumes rather than individual neumes as characters
– using different characters for the same neume at different offsets

Generating such an output would also mean to opt for some special non standard font encoding, thereby limiting the usability of the output considerably.

An interesting compromise between GUI systems and heavily-tagged XML input is currently developed by Haralambous in the Byzantine notation typesetting system ΘΑΜΥΡΙΣ based on luaTEX[2]. He uses the Unicode characters from [2] combined with ASCII characters (`/`, `-`, `<`, `>`) to represent a vertical relation, absence of base character, and offset of diacritic. This system is still under development and not yet ready for production use.

We therefore decided to create our own CPN code, which is both simple (so that converters to different formats can be written without much effort) and does not loose any layout information of the music print.

[2] Yannis Haralambous: ΘΑΜΥΡΙΣ, *A Byzantine Musical Notation Typesetting System based on OTPs, luaTEX callbacks and OpenType fonts.* TUGboat - The communications of the TEX Users group (in preparation)

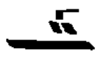
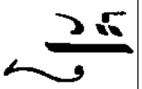
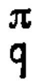
Neume Group	*Output Code*
	`(primary.oligon[0,0];kendimata[-3,2];gorgon.mono-gorgon[-5,3])`
	`(primary.oligon[0,0];linking.heteron[-5,-3];apostrophos[-6,2];kendimata[-2,2];gorgon.mono-gorgon[-4,4])`
	`(martyria.diatonic-hemi-phi[0,-6];letter.small.pa[0,0])`

Table 12 Examples of the output code for single neume groups.

The output is a simple ASCII text file where each line of text represents a line of neume groups in the input image and groups are enclosed in parentheses. The primary neume (or the main martyria or chronos neume) in each group is marked by an appropriate prefix and to each neume its coordinates are attached in square brackets (`[x,y]`). These coordinates are measured in the following coordinate system:

– $y = 0$ on the baseline, $x = 0$ at the right edge of the primary neume bounding box
– the grid unit size is *oligon_height*
– the given coordinate is the position of the lower right edge of the neume bounding box, except for Gorgons, where the leftmost lower edge is used. Note that this results in mostly negative x-coordinates.

Table 12 shows some examples for the encoding of individual neume groups in our code.

4 Summary

Our recognition system covers the complete process from reading a raster image of CPN notation to generating a machine readable code. This includes the measurement of characteristic page dimensions, page and symbol segmentation, neume recognition and syntactical neume grouping.

Two characteristic dimensions (*oligon_width* and *oligon_height*) are measured on a filtered image in which narrow connected components ($width/height < 3$) have been removed. Our experiments show that width and vertical stroke height of the frequent neume *Oligon* can be determined with good accuracy from the histogram of CC widths and black vertical runlengths, respectively. Neume baselines are determined from maxima in the horizontal projection profile of the same filtered image.

An important page segmentation step is the separation of chant text ("lyrics") from neumes. The technical problem of this step lies in the fact that certain neume groups (mostly *martyrias*) extend into the lyrics zone and that they can contain ordinary Greek letters that also appear in the chant text. Our system does this in two stages: first it determines text lines from the horizontal projection profile of the full (unfiltered) image, while utilizing the previously found baseline positions. Then it removes all CCs around the text lines, unless they "seem to belong to a martyria". For the latter criterion, we have devised two different approaches, one purely rule based and one primarily based on trained recognition. Our experiments revealed that both of our approaches have their shortcomings, with the rule based approach being slightly better, though not significantly.

The individual symbols are separated with connected component labeling and their recognition is done with a nearest neighbour classifier. Our experiments on a variety of printed sources have shown that for these sources even simpler features than those proposed by Gezerlis [6] yield good recognition rates. While the leave-one-out performance of the chosen feature set was greater than 99% on all training sets, the final recognition rates for the individual neumes on the test images were lower (between 95% and 98.5%, depending on the source). A considerable fraction of these errors was due to touching or broken characters.

The final neume layout analysis step builds neume groups based on horizontal overlaps. Additionally, our system uses a class naming convention by which not only classes can be specified during training, but also possible grammatical neume functions. This approach worked quite well and lead to grouping error rates between one and three percent, depending on the source.

5 Conclusions and Perspectives

We have developed a prototype of the described system that is freely available [9] and works well on printed books. To further improve its recognition quality, we suggest three starting points: the automatic lyrics removal, the symbol segmentation, and a syntactic post-correction.

Even though the reported error rates for lyrics removal might seem low at first sight, they can require tedious manual correction of the final recognition results. Hence we plan to add a graphical user interface for manually correcting the automatic lyrics removal as an optional interactive step between the page segmentation and recognition stages. Independent from this workaround, the lyrics removal leaves room for further improvement by trying to adapt general page layout analysis methods for complicated layouts, like the use of area Voronoi diagrams [31].

As a considerable fraction of the neume recognition errors was due to touching characters, these can hardly be diminished by further optimising the feature set. Actually the chosen feature set already has a leave-one-out performance of over 99%. It thus seems more promising to have a look at classification based strategies for character segmentation [26], rather than trying to further optimise the feature set.

Another means to improve the final recognition rate could be a lexical or syntactic post-correction, a technique commonly used for improving OCR results [32]. As in our tests most errors made by our system lead to syntactically impossible neume combinations, many of the recognition errors could be automatically detected with the aid of a program for generating CPN notation that utilises the notational conventions of CPN which can be considered as some kind of diagram notation [33]. The recently published third party program *Melodos* [30] actually offers an automatic correction module, which could provide a useful option to improve the recognition rates of our system.

An interesting area of further research could be the reformulation of the neume grouping as a constraint satisfaction problem [34]. The grouping can be considered as a labeling of the neumes under constraints imposed by the notational conventions. This would provide a general framework both to formulating syntactically impossible combinations and for their detection already during the neume layout analysis step.

Our recognition system is not limited to the particular neumes of CPN listed in Table 2. Because of its training abstraction layer, it can be adapted to other variants of psaltic chant notation, including notations in Rumanian and Slavonic as well as paleographic notations. With such an extension into the domain of handwritten manuscripts, we are to expect that some of our algorithms will require modifications due to a higher variance both in the shape of the neumes and in their positioning.

As a first step in the direction of psaltic chant manuscript recognition, we plan to investigate the manuscripts by Angelos L. Boudouris, who was the disciple and First Domestichos of Iakovos Nafpliotis at the Patriarchate of Constantinople during the turn of the 20th century. These manuscripts, approximately 10,000 pages distributed in 18 volumes, use the same CPN as the prints discussed in the present paper.

We hope that our research will eventually help building a machine readable repository of this repertoire that can be used for further musicological research.

Acknowledgements Christine Pranzas is grateful to the association Stoudion, Le Buyet, 26 190 Saint Laurent en Royans

(France) for financial support of her work. Moreover, we thank the anonymous reviewers for their valuable comments.

References

1. Chrysanthos, Archbishop of Dirrachios: Μέγα Θεωρητικὸν τῆς Βυζαντινῆς Μουσικῆς , edited by Panagiotis G. Pelopidis from Peloponnesos, Greece, and published by Michele Weis, Tergest (1832)
2. N. Nicholas: *Unicode Technical Note: Byzantine Musical Notation.* Version 1.1, February 2006. http://www.unicode.org/notes/tn20/ (2006)
3. L. Pugin: *Optical Music Recognition of Early Typographic Prints using Hidden Markov Models.* Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR), pp. 53-56 (2006)
4. L.W.G. Barton, J.A. Caldwell, P.G. Jeavons: *E-Library of Medieval Chant Manuscript Transcriptions.* Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries, pp. 320-329 (2005)
5. D. Hiley: *Western Plainchant: A Handbook.* Oxford University Press (1995)
6. V.G. Gezerlis, S. Theodoridis: *Optical character recognition of the orthodox Hellenic Byzantine Music notation.* Pattern Recognition 35 (4), pp. 895-914 (2002)
7. M. Droettboom, K. MacMillan, I. Fujinaga: *The Gamera framework for building custom recognition systems.* Symposium on Document Image Understanding Technologies, pp. 275-286 (2003)
8. M. Droettboom et al.: *The Gamera Project Homepage.* http://gamera.sourceforge.net/ (2004-2006)
9. C. Dalitz, G.K. Michalakis, Christine Pranzas: *Psaltiki Toolkit for Gamera.* http://psaltiki4gamera.sourceforge.net/ (2007)
10. C. Dalitz, T. Karsten: *Using the Gamera Framework for building a Lute Tablature Recognition System.* Proceedings ISMIR 2005, pp. 478-481 (2005)
11. K. Canfield: *A Pilot Study for a Navajo Textbase.* Proceedings of The 17th International Conference on Humanities Computing and Digital Scholarship (ACH/ALLC), pp. 28-30 (2005)
12. S. Reddy, G. Crane: *A Document Recognition System for Early Modern Latin.* Chicago Colloquium on Digital Humanities and Computer Science (2006)
13. C. Dalitz, M. Droettboom, B. Pranzas, I. Fujinaga: *A Comparative Study of Staff Removal Algorithms.* IEEE Transactions on Pattern Analysis and Machine Intelligence 30, pp. 753-766 (2008)
14. U. Pal, B.B. Chaudhuri: *Indian script character recognition: a survey.* Pattern Recognition 37, pp. 1887-1899 (2004)
15. H. Ma, D. Doermann: *Adaptive Hindi OCR Using Generalized Hausdorff Image Comparison.* ACM Transactions on Asian Language Information Processing 2, pp. 193-218 (2003)
16. T.M. Ha, H. Bunke: *Image Processing Methods for Document Image Analysis.* In H. Bunke, P.S.P. Wang (editors): "Handbook of Character Recognition and Document Image Analysis." World Scientific, pp. 1-47 (1997)
17. W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling: "Numerical Recipes in Pascal". *Cambridge University Press*, 1993
18. R.C. Gonzalez, R.E. Woods: *Digital Image Processing.* Second Edition, Prentice-Hall (2002)
19. A. Rosenfeld, J.L. Pfaltz: *Sequential Operations in Digital Picture Processing.* Journal of the ACM 13 (4), pp. 471-494 (1966)
20. R. Cattoni, T. Coianiz, S. Messelodi, C.M. Modena: *Geometric Layout Analysis Techniques for Document Image Understanding: a Review.* ITC-irst Technical Report TR#9703-09 (1998)
21. S. Mao, T. Kanungo: *Empirical performance evaluation methodology and its application to page segmentation algorithms.* IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (3), pp. 242-256 (2001)
22. E.R. Davies: *Training sets and a priori probabilities with the nearest neighbour method of pattern recognition.* Pattern Recognition Letters 8, pp. 11-13 (1988)
23. A. Kothanzad, Y.H. Hong: *Invariant Image Recognition by Zernike Moments.* IEEE Transactions on Pattern Analysis and Machine Intelligence 12, pp. 489-497 (1990)
24. M. Droettboom: *Correcting broken characters in the recognition of historical printed documents.* Joint Conference on Digital Libraries, pp. 364-366, 2003
25. L.D. Brown, T.T. Cai, A. DasGupta: *Interval Estimation for a Binomial Proportion.* Statistical Science 16 (2), pp. 101-117 (2001)
26. R.G. Casey, E. Lecolinet: *A Survey of Methods and Strategies in Character Segmentation.* IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (7), pp. 690-706 (1996)
27. M. Good: *Lessons from the Adoption of MusicXML as an Interchange Standard.* Proceedings of XML 2006 (2006). See also: http://www.musicxml.org/xml.html
28. P. Roland, J.S. Downie: *Recent Developments in the Music Encoding Initiative Project: Enhancing Digital Musicology And Scholarship.* 19th Joint Conference on the Digital Humanities, Conference Abstracts, pp. 186-189 (2007). See also: http://www.lib.virginia.edu/digital/resndev/mei/
29. G. Schräder: *Ein XML-Datenformat zur Repräsentation kritischer Musikedition unter besonderer Berücksichtigung von Neumennotation.* Studienarbeit, Musikwissenschaftliches Institut der Universität Tübingen (2007). See also: http://www.dimused.info/
30. S. Papadopoulos: *Melodos - Byzantine Music Composer Software.* http://www.melodos.com/ (2008)
31. K. Kise, A. Sato, M. Iwata: *Segmentation of page images using the area Voronoi diagram.* Computer Vision and Image Understanding 70, pp. 370-382 (1998)
32. A. Dengel, R. Hoch, F. Hönes, T. Jäger, M. Malburg, A. Weigel: *Techniques for improving OCR results.* In H. Bunke, P.S.P. Wang (editors): "Handbook of Character Recognition and Document Image Analysis." World Scientific, pp. 227-258 (1997)
33. D. Blostein, L. Haken: *Using Diagram Generation Software to Improve Diagram Recognition.* IEEE Transactions on Pattern Analysis and Machin Intelligence 21, pp. 1121-1136 (1999)
34. E.P.K. Tsang: *Foundations of Constraint Satisfaction.* Academic Press, London and San Diego (1993)