A Comparative Study of Staff Removal Algorithms

Christoph Dalitz, Michael Droettboom, Bastian Pranzas, and Ichiro Fujinaga

Abstract—This paper presents a quantitative comparison of different algorithms for the removal of stafflines from music images. It contains a survey of previously proposed algorithms and suggests a new skeletonization-based approach. We define three different error metrics, compare the algorithms with respect to these metrics, and measure their robustness with respect to certain image defects. Our test images are computer-generated scores on which we apply various image deformations typically found in real-world data. In addition to modern western music notation, our test set also includes historic music notation such as mensural notation and lute tablature. Our general approach and evaluation methodology is not specific to staff removal but applicable to other segmentation problems as well.

Index Terms—Document page segmentation, OMR, performance evaluation, performance metric, pixel classification.

1 Introduction

The most characteristic feature of western music scores is groups of parallel horizontal lines, the *stafflines*. Although they are necessary for a human reader to determine the pitch, they are an obstacle to symbol segmentation in most Optical Music Recognition (OMR) systems. Almost every paper on OMR deals with the problem and suggests a specific staff removal algorithm [1]. Although their primary application is as a preprocessing step in the recognition of western music notation, the staff removal problem also occurs in different contexts, for example, the recognition of bank transfer forms [2].

The goal of staff removal is to remove the lines as much as possible while leaving the symbols on the lines intact. This is achieved by different algorithms with varying success and not all algorithms are equally successful in all situations. Despite the wide variety of suggested staff removal techniques, a comparative evaluation of their quality has not yet been done. A first step in this direction was made by Bainbridge and Bell [3]. In this work, two different staff finding algorithms and three slightly different staff removal algorithms were considered and the accuracy of staff finding on manually labeled music images and the runtimes of the three staff removal methods were compared but not their quality.

From a more general point of view, staff removal is a segmentation problem: Background segments (the staff-lines) need to be separated from foreground objects (the music symbols). For an evaluation of different segmentation algorithms, we must find a way to tell when one segmentation is better than another. This boils down to two questions that are crucial in every segmentation evaluation: how do we measure the distance of a given segmentation from a perfect "ground-truth" segmentation, and how do we obtain the ground-truthing data?

To answer the first question, we must define an appropriate error metric. Although from an OMR point of view, the final music recognition rate might seem to be a natural performance measure, this is not a good error metric because it depends on too many other aspects than staff removal quality. This is analogous to the problem of page segmentation evaluation, where goal-directed evaluations like the final Optical Character Recognition (OCR) error rate have recently been abandoned in favor of direct measurements of page segmentation quality [4], [5]. Inspired by these works, we develop three error metrics for our problem. Two of these metrics are in no way specific to staff removal but can be applied to any segmentation problem.

Having established an error metric, we can define the segmentation quality as the distance between an output image of an algorithm and a ground-truth image, where in both images, all black pixels are labeled as either background or foreground (that is, *staff* or *nonstaff*). Even though the labeling of the ground-truth data could be done manually, this is very time consuming and has the disadvantage of an ad hoc classification of dubious pixels belonging both to a staffline and a crossing symbol. Therefore, we generate our music images from postscript images created with music typesetting software, which allows for "perfect" staff removal. To measure the robustness of the algorithms with respect to particular common degradations occurring in real-world data, we deform the "ideal" images, an approach that also has been used in the evaluation of OCR systems [6]. This approach raises yet another problem: The "perfect" staff removal is only possible on the "ideal" postscript images. Hence, we have developed a general scheme for simultaneous deformations of both the unsegmented and the already

- *C. Dalitz and B. Pranzas (formerly Czerwinski) are with Hochschule Niederrhein, Fachbereich Elektrotechnik und Informatik, Reinarzstr. 49, 47805 Krefeld, Germany. E-mail: christoph.dalitz@hs-niederrhein.de, ba.czerwinski@gmail.com.*
- *M. Droettboom is with the Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218. E-mail: mdroe@stsci.edu.*
- *I. Fujinaga is with the Schulich School of Music, McGill University, 555 Sherbrooke W., Montreal QC, Canada H3A 1E3. E-mail: ich@music.mcgill.ca.*

Manuscript received 26 Feb. 2007; revised 24 May 2007; accepted 27 June 2007; published online 19 July 2007.
Recommended for acceptance by E. Saund.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0135-0207.
Digital Object Identifier no. 10.1109/TPAMI.2007.70749.

Fig. 1. The characteristic page dimensions *staffline_height* and *staffspace_height*.

segmented image. This novel approach can also be applied to the evaluation of other segmentation problems.

We have implemented a framework for the evaluation of staff removal algorithms as a toolkit within the Gamera document image analysis framework [7]. We make our code freely available under the terms of the GNU General Public License [8]. As pointed out in [5], this can be utilized not only for the evaluation of other algorithms but also for the optimization of parameters in a staff removal algorithm.

All of the algorithms studied in this paper make no assumptions about the appearance of the symbols that are superimposed on the staff. Some OMR systems use probabilities from later stages, such as symbol or structure recognition, to disambiguate between staffline and nonstaffline sections [9], [10], and it is possible that such approaches improve the accuracy of staff removal. However, these approaches, by their very nature, have tightly coupled subsystems and would be difficult to compare with one another. Therefore, the present paper limits itself to staff removal independent of any of the other parts of a complete optical music recognition system.

This paper is organized as follows: Section 2 gives some basic definitions and covers the related problem of staffline finding without removal. Section 3 categorizes and describes the investigated algorithms. Section 4 describes our method for generating and deforming our test data and describes our test set. The final sections contain our experimental results and the conclusions drawn therefrom.

2 Staffline Detection

The problem of staffline detection is occasionally considered in the literature without the goal of their removal [11], [12]. These methods can be used as a first step in the staff removal algorithms described in Section 3.1. Moreover, they are even useful in segmentation-free OMR approaches like hidden Markov modeling [13], which do not require staff removal.

Stafflines are usually grouped in *staffs*, that is, groups of parallel lines. Common music notation uses five lines per staff, but other forms of notation can also use a different number (chant uses four lines and tablature typically six lines). Most algorithms for staff detection or removal rely on an estimation of the staffline thickness (*staffline_height*) and the vertical line distance within the same staff (*staffspace_height*), see Fig. 1). As shown by Fujinaga [14], these values can be estimated with good accuracy as the most frequent black (staffline_height) and white (staffspace_height) vertical runlength, respectively. Based on these values, we can define a generic scheme for finding stafflines that is a generalization of the method described in [11]. It operates on a set of "staffsegments" and requires methods for linking two "staffsegments" horizontally and vertically and for merging two segments with overlapping positions into one:

1. Add vertical links between staffsegments with a vertical distance around
$$staffline\_height + staffspace\_height.$$
2. Add horizontal links between adjacent staffsegments possibly belonging to the same staffline.
3. Partition the resulting graph into connected subgraphs; each subgraph that is wide and high enough corresponds to a staff.
4. All staffsegments within a system are labeled as belonging to a certain staffline. Segments of the same line at the same horizontal position are merged into one segment.
5. Due to ledger lines, ties, and beams, some subgraphs will contain too many stafflines. To reduce them to a predefined number of lines per staff (typically five for modern notation, four for chant and six for tablature), the outer stafflines of each staff are subsequently removed until the predefined number of stafflines remains.

Depending on the data representation of a "staffsegment" and on the method for identifying staffsegment candidates, we obtain different staff detectors:

- Miyao's "staffsegments" are points on equidistant vertical scan lines. These are horizontally linked with dynamic programming matching [11].
- Szwoch's "staffsegments" are peaks in the horizontal projection profile of vertical slices. They are horizontally linked when their vertical distance is small enough [12].

Both methods only yield a polygonal approximation of the staffline skeleton, which is not sufficient for the staff removal techniques described in Section 3.1. Bainbridge and Bell's method [3] could be used to follow the skeleton more closely, but it turned out to be rather unstable in our experiments in the presence of staffline interruptions. Hence, we have developed a new algorithm that directly yields the staffline skeleton:

1. We extract horizontal runs with more than 60 percent black pixels within a window of width *staffspace_height*.
2. The resulting filaments are vertically thinned by replacing each vertical black run with its middle pixel. For black runs higher than $2 * staffline\_height$, more than one skeleton point is extracted.
3. The resulting skeleton segments wider than $2 * staffspace\_height$ are the "staffsegments" to which the generic staff-finding algorithm is applied. As Step 1 fills out most staffline interruptions, we omit the horizontal linking step.

3 Staffline Removal

The different approaches to staff removal in the literature can be divided into the following categories:

TABLE 1
Evaluated Algorithms

Algorithm	*Reference*	*Category*
LineTrack Height	[19]	Line Tracking
LineTrack Chord	[3] [16]	Line Tracking
Roach/Tatem	[15]	Vector Field
Carter	[17]	Runlength
Fujinaga	[14]	Runlength
Skeleton	new algorithm	Skeletonization

- *Line Tracking.* Stafflines are first localized by some method. Each line is then tracked to see whether some of the pixels need to be removed based on some criterion. All methods discussed in [3] fall into this category.
- *Vector Field.* Pixels of the one-bit image are converted into vectors. Vector directions and lengths are used as criteria to keep or remove pixels. An early work on OMR already used this approach [15]. The vector field has also been used in combination with line tracking [16].
- *Runlength.* Black runlengths are removed based on their length or line adjacency. The algorithms by Carter and Bacon [17] and Fujinaga [14] fall in this category.
- *Skeletonization.* The bitmap image is skeletonized, and this skeleton is further analyzed to obtain staff segments and symbols. This approach has been used by Ng to detect staff lines in music manuscripts [18], although he does not describe what criteria for staff segments did he actually use.

Table 1 lists all staff removal algorithms that we have evaluated and their category. The following sections describe them in detail.

3.1 Line Tracking

One obvious approach to staff removal is to first detect the staff skeleton and then remove the vertical black run around each skeleton point satisfying a criterion that indicates whether it belongs to a crossing symbol or not. To obtain the staff skeleton, we use the staff detector described in Section 2.

For symbol detection, we have implemented two methods. The first one checks whether the black run through the skeleton point is longer than $2 * staffline\_height$ [19]. This results in algorithm "LineTrack Height" in Table 1.

The other method computes, for a fixed angle resolution of three degrees, the chord lengths through the skeleton point (see Fig. 2). This results in a function $chordlength(\varphi)$ (with φ being the chord angle) for each skeleton point. For staffline pixels, this function has a sharp peak at the staffline orientation angle (typically zero degrees). When the staffline pixel also belongs to a crossing music symbol, the function should have a second distinct peak [16]. In contrast to Martin and Bellissant, who trained a neural network for the "second distinct peak" criterion, we use the following hard-coded thresholds for detecting a "second distinct peak" in the function $chordlength(\varphi)$:

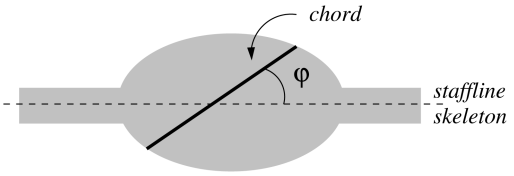


Fig. 2. Length of a chord through a skeleton point at some angle φ.

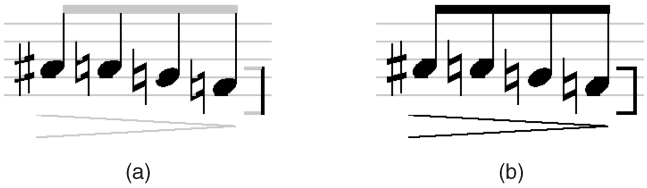


Fig. 3. Roach and Tatem's original algorithm removes all horizontal shapes (marked gray) even when they do not lie on staff lines. This is resolved in our "improved" version. (a) Original. (b) Improved.

- There must be a local maximum with a chord-length greater than $staffline\_height * 5$ at an angle below 30 degrees (representing the staffline) and another local maximum with a chordlength greater than $1.75 * staffline\_height * \sin(\varphi)$ at an angle $\varphi > 30$ degrees (representing the second chord).
- The valley between two maxima must have a depth greater than $1.5 * staffline\_height$.

This leads to algorithm "LineTrack Chord" in Table 1.

3.2 Vector Fields

When the chord length versus angle function described in the previous section is computed for every black pixel, the original image can be transformed into a two-dimensional vector field by picking the angle and length of the longest chord for each black pixel. This will assign pixels on stafflines a high "length" value and an "angle" value around zero.

Roach and Tatem used a labeling scheme based on the angle information and pixel adjacency to identify these staffline pixels [15]. This extracts a number of "horizontal line pixels," some of which belong to music symbols. To avoid the removal of symbol pixels on the stafflines, some horizontal line pixels are iteratively relabeled as nonhorizontal pixels, depending on the labels of their neighboring pixels. Eventually, all remaining horizontal pixels are removed.

As the application of Roach and Tatem was handwritten music, their method simply removes all perfectly straight horizontal shapes. This includes not only staff lines but also beams or parts of lyrics (see Fig. 3). To overcome this shortcoming, we have added the following modification:

- On all horizontal line pixels, we apply our staff-finding algorithm described in Section 2.
- Only line segments on the detected stafflines are removed.
- To avoid beam removal, vertical black runs longer than $2 * staffline\_height$ are not removed.

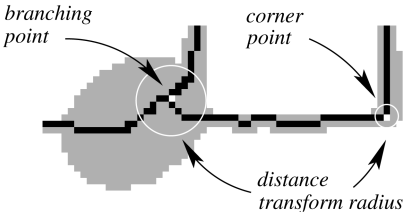


Fig. 4. Pixels within the distance transform radius around each splitting point are removed.

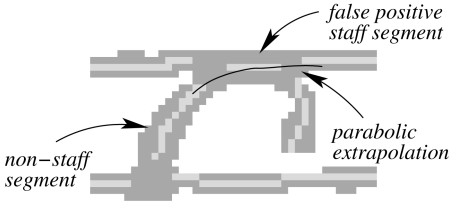


Fig. 5. A falsely detected staff segment that can be identified as belonging to a music symbol because it is approximately tangential to an extrapolated parabola from a nonstaff segment.

3.3 Runlength Analysis

Rather than considering chords in arbitrary directions, it is computationally more efficient to only consider chords in the horizontal or vertical direction, that is, black horizontal or vertical runlengths. The analysis of vertical black runlengths is particularly useful, because staffline sections can be identified as adjacent vertical runlengths with a vertical runlength around *staffline_height*.

Carter and Bacon [17] segment the image using a concept known as the line-adjacency graph. Each segment resulting from this analysis is either part of a staffline or not, so once the sections have been found, no further analysis at the pixel level is necessary. Staffline fragments are found by finding obvious straight and horizontal candidates for staffline sections ("filaments") and then vertically linking filaments that overlap horizontally and have a vertical distance around *staffspace_height*. Eventually, these fragments are horizontally linked to other fragments by horizontal extrapolation. Due to Carter and Bacon's transformation of the line-adjacency graph, the resulting sets of staff filaments do not contain the symbols and can be directly removed.

Fujinaga [14] uses a different method to segment the image. He first removes the black vertical runs larger than twice the *staffline_height* and considers in the resulting image all connected components with a width greater than *staffspace_ height*. As his primary criterion for staff components is their height, it is essential that the stafflines are not rotated nor curved. Hence, he first detects staffs by horizontal projections and then deskews each staff by correlating the horizontal projection profiles of adjacent vertical strips; each strip is sheared to the position with maximal correlation. As this deskewing makes it difficult to compare the results with ground-truthing data, we have added an undoing of this deskewing as a final step to Fujinaga's original staff removal algorithm.

3.4 Skeletonization

Skeletonization is a common technique in OMR but is usually applied *after* staff removal [16], [19], because the stafflines distort the symbol skeletons considerably at intersection points. We have developed a new staff removal algorithm that uses the skeleton information but performs the staff removal on the original image instead of the skeleton.

The method relies on the fact that symbols on the stafflines lead to junction points or corner points in the skeleton. It consists of the following steps:

1. The skeleton is split at branching points and corner points with an angle below 135 degrees. Around each splitting point, a number of pixels (taken from the distance transform at the splitting point) is removed (see Fig. 4). The latter avoids that staffline skeleton segments extend too far into crossing objects.
2. Staff line segment candidates are picked as skeleton segments with the following properties:
   - the orientation angle (least square fitted line) is below 25 degrees,
   - the segment is wider than tall, and
   - the "straightness" (mean square deviation from least square fitted line) is below
   $$staffline\_height^2/2.$$
3. On these staff segment candidates, the generic staff-finding scheme described in Section 2 is applied. Two staff segments are horizontally linked when their extrapolations from the end points with the least square fitted angle come closer than
   $$staffline\_height/2.$$
4. A staff may still contain some false positives. When staff segments assigned to the same line overlap, this is a clear indication of a false positive. Thus, from each overlapping staff segment group on the same line, the one that is closest to its least square-fitted neighborhood is picked, and the others are discarded.

   To check for further false positives, nonstaff segments that have the same branching point as a staff segment are extrapolated by a parametric parabola. If this parabola is approximately tangential to the staff segment, the latter is considered a false positive (see Fig. 5).
5. In order to remove staff lines, all vertical black runs around the detected staff skeleton are removed. As skeleton branches occasionally extend into solid regions (music symbols), vertical runs are only removed when they are not longer than twice the *staffline_height*.

4 Ground-Truthing Data

For a qualitative comparison of the different staff removal algorithms, we need ground-truthing data, that is, music images where all black pixels are labeled as either *staff* or *nonstaff* pixels. A manual labeling is time consuming and has the disadvantage of an ad hoc classification of dubious pixels that belong both to a staff line and a crossing symbol.

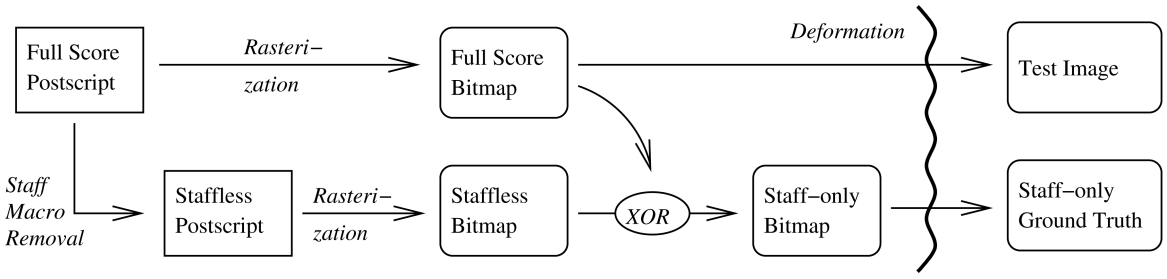


Fig. 6. Control flow for generating test images and ground-truth data. The deformation is performed in parallel on both images.

Hence, we chose a different strategy and generated Postscript images with a music notation software. In the Postscript code, we removed the staff drawing macros and converted the resulting images to one-bit raster images. Thus, we obtain the ideal staffless images. If this is meant to emulate scanned images of music prints, an appropriate model for image defects introduced through printing and scanning is necessary. Rather than introducing these defects in the rasterization stage, we can alternatively add them later as "image deformations" [6]. The process for generating test images is shown in Fig. 6.

4.1 Our Test Set

The test set of ideal images consists of 32 sample pages with a total of about 300 staffs generated with a variety of music notation programs. It covers a wide range of music types (common music notation, lute tablature, chant, and mensural notation) and music fonts. Table 2 shows how the samples are distributed over the different notation programs and notation types. We have made the test set freely available together with our source code [8].

TABLE 2
Distribution of Notation Programs and Types in Our Test Set

Staffs	*Percentage*	*Program*
78	26.1	abctab2ps
56	18.7	Philip's Music Writer
49	16.4	LilyPond
29	9.7	Django
25	8.4	MusixTex
24	8.0	Finale
24	8.0	Music Publisher (mup)
14	4.7	lute tab
299	100.0	total sum

Staffs	*Percentage*	*Type*
197	65.9	modern
58	19.4	tablature
44	14.7	historic (chant, mensural)
299	100.0	total sum

4.2 Image Deformations

In order to test the robustness of the different staff removal algorithms, we created distorted images from the ground-truth data. There are two categories of image distortions:

- *deterministic* deformations (for example, rotation), which depend on certain parameters, and
- *random* defects (for example, noise), which usually also have parameters like mean and variance but additionally rely on a pseudorandom number generator.

In both cases, it is necessary to apply the deformation in parallel both to the original and the ground-truth staff image.

Although the defects *resolution*, *rotation*, and *line interruption* in Table 3 are self explanatory, the others need some explanations. The *curvature* is done as a half sine wave over the entire staffwidth. The strength of the resulting curvature can be measured as the ratio of amplitude (height) and width of the wave (see Fig. 7b).

TABLE 3
Investigated Image Defects

Deformation	*Type*	*Parameter description*
Resolution	deterministic	dots per inch
Rotation	deterministic	rotation angle
Curvature	deterministic	height:width ratio of sine curve
Typeset emulation	both	gap width, maximal height and variance of vertical shift
Line interruptions	random	interruption frequency, maximal width and variance of gap width
Staffline thickness variation	random	Markov chain stationary distribution and inertia factor
y-variation of staffline	random	Markov chain stationary distribution and inertia factor
degradation after Kanungo et al.	random	$(\eta, \alpha_0, \alpha, \beta_0, \beta, k)$, see [21]
white speckles	random	speckle frequency, random walk length and smoothing factor

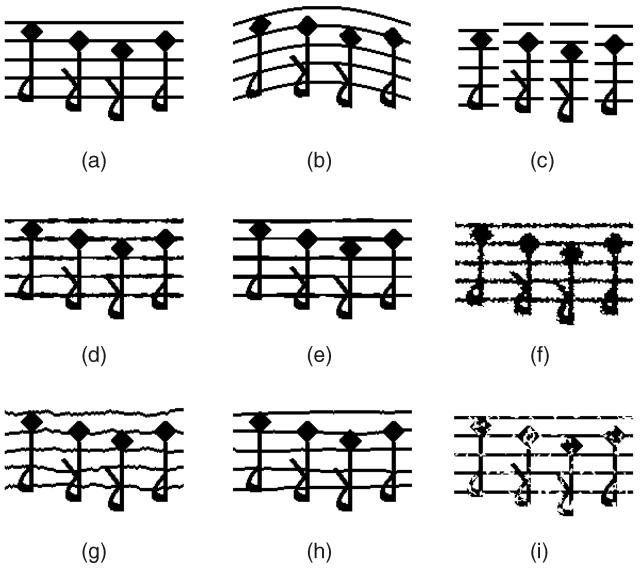


Fig. 7. An "ideal" image and its deformations. (a) Ideal image. (b) Curvature. (c) Typeset emulation. (d) Line thickness variation $(n, c) = (6, .5)$. (e) Line thickness variation $(n, c) = (6, .93)$. (f) Degradation after Kanungo $(\eta, \alpha_0, \alpha, \beta_0, \beta, k) = (0, 1, 1, 1, 1, 2)$. (g) Line y-variation $(n, c) = (5, .6)$. (h) Line y-variation $(n, c) = (5, .93)$. (i) White speckles $(p, n, k) = (.025, 10, 2)$.

The particular defect *typeset emulation* tries to imitate sixteenth century prints that are set with lead types and thus have staffline interruptions between symbols and a random vertical shift of each vertical staff slice containing a symbol (see Fig. 7c).

Some defects like staffline *thickness variation* or *y-variation* are best depicted by a Markov chain describing the evolution of the staffline thickness from left to right, because, usually, the thickness at a particular x-position depends on the thickness at the previous x-position. In these cases, the parameter is the transition probability matrix P of the Markov chain with

$$p_{ij} := \text{probability of transition from thickness or } y\text{-deviation } i \text{ to thickness or } y\text{-deviation } j.$$

The thickness or y-deviation can be one of n different values ("states"). Let $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ be the stationary distribution of the individual states i, that is,

$$\pi P = P \quad \text{and} \quad \sum_{i=1}^{n} \pi_i = 1.$$

For this distribution, we assume a symmetric binomial distribution, that is,

$$\pi_i = \binom{n-1}{i-1} \cdot \frac{1}{2^{n-1}}.$$

We associate the mean value $(n-1)/2$ of this distribution with the original value in the undeformed image (*staffline_height* for the thickness or zero for the deviation from the original y-position). We generate the Markov chain with the Metropolis-Hastings algorithm [20] where our choice for the transition probability matrix Q for picking candidate transition points is

$$q_{ij} = \begin{cases} c & \text{for } j = i \\ 1 - c/2 & \text{for } j = i \pm 1 \\ 0 & \text{otherwise.} \end{cases}$$

The probability c for not changing the state can be considered as an inertia factor that allows for smooth transitions: the closer c is to one, the slower is the state variation (see Figs. 7g and 7h)).

Kanungo et al. have suggested a degradation model for emulating local distortions introduced during printing and scanning [21]. The model has six parameters $(\eta, \alpha_0, \alpha, \beta_0, \beta, k)$ with the following meaning:

- Each foreground pixel is flipped with probability $\alpha_0 e^{-\alpha d^2} + \eta$, where d is the distance to the closest background pixel.
- Each background pixel is flipped with probability $\beta_0 e^{-\beta d^2} + \eta$, where d is the distance to the closest foreground pixel.
- Eventually, a morphological closing operation is performed with a disk of diameter k.

This degradation model is designed such that it primarily effects the contour of a one-bit image and has little effect on bulk pixels. Hence, we add a second degradation model for generating white speckles within the music symbols and stafflines. Our model has three parameters (p, n, k) with the following meaning:

- Each black pixel in the original image is taken with probability p as a starting point for a random walk of length n.
- An image containing the random walk is smoothed by a closing operation with a rectangle of size k.
- Eventually, the image with the random walks is subtracted from the original image, which results in white speckles at the random walk positions.

Consequently, p can be interpreted as the speckle frequency, n as a measure for the speckle size, and k as a smoothing factor.

Table 4 gives the values for all parameters that we have applied to our test images. It should be noted that in real scan data, the deformations usually do not occur in their pure form. When working with facsimile copies of sixteenth century prints, we typically found a combination of curvature, typeset emulation, line interruptions, staffline thickness variation, and white speckles. For the purpose of an evaluation of segmentation algorithms however, it is more instructive to investigate each defect in isolation so that conclusions can be drawn about the actual reason for the breakdown of a particular algorithm.

5 Error Metrics

Although we can create ground-truth data as described above, it is not obvious how to match these data to the corresponding output of the different staff removal algorithms in order to establish a quality measure for staff removal. We, therefore, used different error metrics that seem

TABLE 4
Ranges of Deformation Parameters in Our Tests Given as *min:step:max*

Deformation	*Parameter range*
Resolution	*dpi* = 50:25:500
Rotation	*angle* = -18:1:18
Curvature	*amplitude/staffwidth* = 0.02:0.02:0.3
Typeset emulation	*n_gap* = 1:1:12, *n_shift* = 1:1:10
Line interruptions	frequency α = 0.01:0.01:0.1, binomial parameter for width: n = 1:1:10, p = 0.5
Staffline thickness variation	inertia c = 0.5:0.05:0.95, maximum thickness n = 2:1:10
y-variation of staffline	inertia c = 0.5:0.05:0.95, maximum deviation n = 2:1:10
degradation after Kanungo et al.	η = 0, k = 2, α_0 = [0.5,1], α = 0.25:0.25:1.5, β_0 = [0.5,1], β = 0.25:0.25:1.5
white speckles	smoothing factor k = 2, random walk length n = 10, speckle frequency p = 0.01:0.01:0.5

reasonable. In the following, we give error metrics based on *individual pixels*, *staff-segment regions*, and *staff interruption location*.

5.1 Pixel Level

When we consider staff removal as a two-class classification problem at the pixel level ("staff line pixel" or not), a natural performance measure is the error rate for this classification, that is (# means "number of")

$$\frac{\#\text{misclassified staff pixels} + \#\text{misclassified non staff pixels}}{\#\text{all black pixels}}.$$

Although this error rate indicates how badly the symbols are distorted when compared to the ideal staff-less image, it gives little information how well the staff removal algorithm separates symbols that are otherwise connected by staff lines. To measure the latter, we have developed two other error metrics.

5.2 Segmentation Region Level

Staff removal can be considered as a segmentation problem: staff segments are to be separated from symbol segments. Although in an OMR application, staff segments are considered as "background" and the symbols are the "segments of interest," for the purpose of staff removal evaluation, the situation is reversed: staff segments are "of interest" and the rest constitutes the "background." This segmentation problem shows some analogy to the page segmentation problem in text documents for which performance metrics based on missed, split, and merged segments have been suggested [22], [4], [5].

Following the notation in [4], we have two segmentations for the set of black pixels in the test image: The ground-truth segmentation $G = G_{obj} \cup \{g_{noise}\}$, with $G_{obj} = \{g_1, \ldots, g_M\}$, and the segmentation guessed by the algorithm $S = S_{obj} \cup \{s_{noise}\}$, with $S_{obj} = \{s_1, \ldots, s_N\}$, where each g_i and s_j contains the black pixels of a contiguous staff segment, respectively, and g_{noise} and s_{noise} contain the remaining background black pixels, respectively.

TABLE 5
Staff Segment Extraction Errors Based on the Number of Segments in an Equivalence Class r

Segments from G_{obj}	*Segments from S_{obj}*	*Error description*
1	1	correct
1	0	missed segment
0	1	falsely detected segment
1	> 1	segment split
> 1	1	segments merged
> 1	> 1	both splitting and merging occurred

In the set of all staff segments from both segmentations $G_{obj} \cup S_{obj}$, we build equivalence classes of overlapping segments, that is, two segments are considered equivalent $a \simeq b$ when a sequence $c_1, c_2, \ldots, c_n$ exists with $c_1 = a$, $c_n = b$ and $c_i \cap c_{i+1} \neq \emptyset$. For each resulting equivalence class r we count the contained numbers of segments from G_{obj} and S_{obj} and can thus detect recognition errors. All possible cases are listed in Table 5.

Note that Thulke et al. [4] additionally take into account whether a class r overlaps with s_{noise} and g_{noise} and, thus, obtain many more error cases. This is however not appropriate in our situation because overlaps of detected staff segments with ground-truth background, and vice versa, always occur and would consequently spoil our error rate.

Although the numbers for the individual errors in Table 5 are of interest for a detailed error analysis of an individual algorithm, we can also consider the error rate among the equivalence classes as a single error measure, that is,

$$\frac{\#\text{all classes } r \; - \; \#\text{classes representing a correct recognition}}{\#\text{all classes } r}.$$

5.3 Staffline Interruptions

As it is possible to extract the ideal location of the staff line skeleton from our ground-truth data, we can reduce the matching problem to a matching problem of one dimensional intervals. To do so, we follow each staff line from left to right in the images containing only the removed staff fragments and look for interruptions in the staff line. Each interruption represents a detected music symbol that crosses the staff line. This yields for each staff line two sets of intervals: The interrupting intervals $G = \{g_1, \ldots, g_M\}$ in the ground-truth data and those in the algorithm output $S = \{s_1, \ldots, s_N\}$.

For establishing an error metric, we create a bipartite graph by adding links between intervals g_i and s_j that overlap (see Fig. 8). This reveals two types of errors:

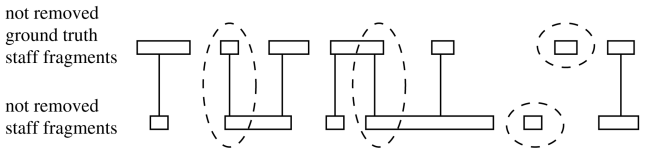


Fig. 8. Matching remaining staff fragments from ground-truth data to staff removal result. Errors are marked with dashed ellipses.

intervals from G and S without a link and intervals with more than one link. To count the number of errors of the second type, we compute the maximum cardinality matching [23] in this graph, which will remove the minimal number of links leading to the second type error. As a resulting error rate, we use

$$\frac{\min\{\#\text{ground-truth interruptions},\ \#\text{interruptions without link} \ + \ \#\text{removed links}\}}{\#\text{ground-truth interruptions}}.$$

Note that this error metric can not be computed from the images alone that are to be compared but requires additional information about the location of the staff skeleton. Moreover, it only measures the segmentation on the staffline and ignores all other errors like erroneously removed segments that do not touch the stafflines.

5.4 Methods of Statistical Analysis

For a comparative performance, evaluation an appropriate statistical analysis is necessary. The simplest approach would be to use the *average error rate* over all test images as an estimator for the performance of an algorithm. In this estimator, all images have the same weight, regardless of whether they have many staffs and, thus, more symbols or fewer staffs. A more representative estimator for the performance on the test set is the *total* or *overall error rate*, that is, the error rate that is obtained when all test images are considered as if they were a single large image.

The aforementioned averages can be used to visualize the qualitative behavior of an algorithm over a range of deformation parameters but are not sufficient to determine whether one algorithm performs better than another because they lack the information whether a difference in the error rate average is significant or not. To answer this question, Mao and Kanungo [5] proposed the following *paired model approach*.

Let X_{ij} be the observed error rate of algorithm i on image j. For the paired model approach to be applicable, we assume that observations for the same algorithm on different images are statistically independent, so that the observations X_{ij}, $j = 1, \ldots, n$ for a fixed algorithm i are independent and identically distributed (iid) random variables, whose mean value μ_i is the "true" performance of the algorithm. Observations of different algorithms on the same image however are not assumed to be independent, because we would expect that a difficult to segment image leads to worse error rates for several algorithms. Now, we consider a new observable

$$W_{ii'j} = X_{ij} - X_{i'j} \quad \text{for } i \neq i'$$

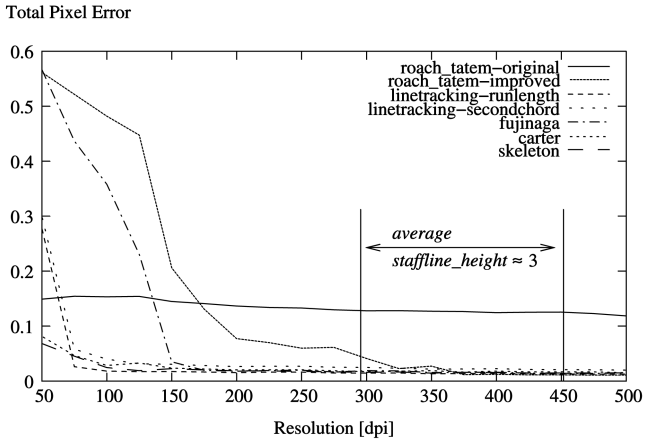


Fig. 9. Overall pixel error rate for all algorithms at different resolutions.

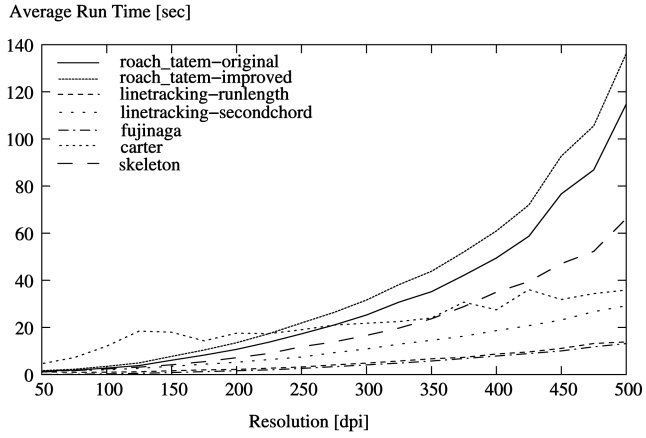


Fig. 10. Runtime for all algorithms at different resolutions.

that is the error rate difference between algorithms i and i' on image j. Under the above independence assumption, $W_{ii'j}$ and $W_{ii'j'}$ are independent for $j \neq j'$ (that is, on different images) and, thus, independent observations of the same random variable $W_{ii'}$. Consequently, the sample mean $\overline{W}_{ii'}$ over all images is an estimator for the true mean error rate difference $\Delta_{ii'}$ between the two algorithms i and i'. As shown by Mao and Kanungo [5], a confidence interval for the true difference $\Delta_{ii'}$ at a given confidence level α is given by

$$\Delta_{ii'} \in \overline{W}_{ii'} \pm \frac{t_{\alpha/2,n-1} V_{ii'}}{\sqrt{n}},$$

where n is the number of test images, $V_{ii'}^2$ is the sample variance of the n observed $W_{ii'}$, and $t_{\alpha/2,n-1}$ is the *percentile* (the inverse CDF) of the t distribution with $n-1$ degrees of freedom.

To test whether the true error rate means of algorithms i and i' are statistically different, we consider the null hypothesis $\Delta_{ii'} = 0$. Under this hypothesis, the test statistic $T = \overline{W}_{ii'}\sqrt{n}/V_{ii'}$ is distributed approximately as a t distribution with $n-1$ degrees of freedom, which have the probability density $f(t)$. Thus, we reject the null hypothesis when

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

TABLE 6
Paired Model Results $\Delta_{ii'}$ in Percent for the Different Error Metrics on the Undeformed Test Set

Pixel Error

	Fujinaga	*Linetrack Runlength*	*Linetrack Secondchord*	*RoachTatem improved*	*RoachTatem original*	*Skeleton*
Carter	-0.63 ± 1.12 $P_{val} = 0.268$	0.24 ± 0.41 $P_{val} = 0.2472$	$\mathbf{-0.82 \pm 0.56}$ $P_{val} = 0.0056$	0.20 ± 0.41 $P_{val} = 0.3233$	$\mathbf{-9.58 \pm 4.70}$ $P_{val} = 0.0002$	0.03 ± 0.43 $P_{val} = 0.8808$
Fujinaga		0.87 ± 1.04 $P_{val} = 0.0984$	-0.18 ± 1.01 $P_{val} = 0.7115$	0.84 ± 1.04 $P_{val} = 0.1105$	$\mathbf{-8.94 \pm 4.78}$ $P_{val} = 0.0006$	0.67 ± 1.00 $P_{val} = 0.1841$
Linetrack Runlength			$\mathbf{-1.06 \pm 0.44}$ $P_{val} = 0.0000$	-0.03 ± 0.05 $P_{val} = 0.1574$	$\mathbf{-9.81 \pm 4.76}$ $P_{val} = 0.0002$	$\mathbf{-0.21 \pm 0.16}$ $P_{val} = 0.0132$
Linetrack Secondchord				$\mathbf{1.02 \pm 0.44}$ $P_{val} = 0.0000$	$\mathbf{-8.76 \pm 4.54}$ $P_{val} = 0.0004$	$\mathbf{0.85 \pm 0.38}$ $P_{val} = 0.0001$
RoachTatem improved					$\mathbf{-9.78 \pm 4.75}$ $P_{val} = 0.0002$	$\mathbf{-0.17 \pm 0.16}$ $P_{val} = 0.0319$
RoachTatem original						$\mathbf{9.61 \pm 4.72}$ $P_{val} = 0.0002$

Segmentation Error

	Fujinaga	*Linetrack Runlength*	*Linetrack Secondchord*	*RoachTatem improved*	*RoachTatem original*	*Skeleton*
Carter	$\mathbf{-6.05 \pm 4.22}$ $P_{val} = 0.0064$	-0.66 ± 2.68 $P_{val} = 0.6214$	-0.63 ± 2.22 $P_{val} = 0.5660$	-1.81 ± 2.45 $P_{val} = 0.1429$	$\mathbf{-41.02 \pm 15.33}$ $P_{val} = 0.0000$	$\mathbf{-3.54 \pm 3.46}$ $P_{val} = 0.0451$
Fujinaga		$\mathbf{5.39 \pm 3.19}$ $P_{val} = 0.0017$	$\mathbf{5.42 \pm 4.19}$ $P_{val} = 0.0129$	$\mathbf{4.24 \pm 2.99}$ $P_{val} = 0.0069$	$\mathbf{-34.98 \pm 13.53}$ $P_{val} = 0.0000$	2.51 ± 3.10 $P_{val} = 0.1089$
Linetrack Runlength			0.02 ± 3.19 $P_{val} = 0.9875$	-1.15 ± 1.22 $P_{val} = 0.0640$	$\mathbf{-40.37 \pm 15.34}$ $P_{val} = 0.0000$	-2.88 ± 2.90 $P_{val} = 0.0513$
Linetrack Secondchord				-1.17 ± 3.02 $P_{val} = 0.4346$	$\mathbf{-40.39 \pm 15.25}$ $P_{val} = 0.0000$	-2.91 ± 3.35 $P_{val} = 0.0864$
RoachTatem improved					$\mathbf{-39.22 \pm 14.92}$ $P_{val} = 0.0000$	-1.74 ± 2.58 $P_{val} = 0.1795$
RoachTatem original						$\mathbf{37.48 \pm 14.48}$ $P_{val} = 0.0000$

Interruption Error

	Fujinaga	*Linetrack Runlength*	*Linetrack Secondchord*	*RoachTatem improved*	*RoachTatem original*	*Skeleton*
Carter	$\mathbf{-3.34 \pm 3.02}$ $P_{val} = 0.0313$	-2.01 ± 2.46 $P_{val} = 0.1058$	-2.52 ± 2.98 $P_{val} = 0.0945$	$\mathbf{-4.89 \pm 4.61}$ $P_{val} = 0.0384$	$\mathbf{-3.42 \pm 3.25}$ $P_{val} = 0.0399$	$\mathbf{-4.92 \pm 4.46}$ $P_{val} = 0.0319$
Fujinaga		1.32 ± 2.00 $P_{val} = 0.1855$	0.82 ± 3.03 $P_{val} = 0.5869$	-1.55 ± 3.69 $P_{val} = 0.3988$	-0.08 ± 3.23 $P_{val} = 0.9587$	-1.58 ± 3.95 $P_{val} = 0.4205$
Linetrack Runlength			-0.51 ± 2.98 $P_{val} = 0.7302$	-2.87 ± 3.26 $P_{val} = 0.0817$	-1.41 ± 2.50 $P_{val} = 0.2604$	-2.91 ± 3.99 $P_{val} = 0.1475$
Linetrack Secondchord				-2.37 ± 3.50 $P_{val} = 0.1778$	-0.90 ± 3.05 $P_{val} = 0.5520$	-2.40 ± 2.97 $P_{val} = 0.1098$
RoachTatem improved					1.47 ± 3.21 $P_{val} = 0.3582$	-0.03 ± 3.10 $P_{val} = 0.9837$
RoachTatem original						-1.50 ± 3.84 $P_{val} = 0.4320$

Negative values mean "row is better than column." Bold entries are significant at $\alpha = 0.05$.

In other words, this is the condition for a statistically significant difference at a given confidence level α.

6 Results

6.1 Choice of Resolution

Concerning the bitmap image resolution, it is always necessary in the OMR to find a compromise between two different effects: Although the recognition accuracy increases with a higher resolution, memory requirement and runtime also increase with the square of the resolution.

Figs. 9 and 10 show that the same holds for staff removal on our test set. For resolutions beyond 350 dpi, the error rate does not improve. For all algorithms except Roach and Tatem's, the error rate saturates already at a lower resolution. In our test set, the resolution between 300 and 450 dpi corresponds to an average staffline_height of around three pixels. Thus, our results provide experimental evidence for Fujinaga's rule of thumb that the thinnest relevant objects (in our case the stafflines) should have a thickness of about three pixels [14].

The runtime differences of the individual algorithms in Fig. 10 are not necessarily inherent to the algorithms but can also be due to different levels of optimization in our implementation. Carter and Bacon's algorithm, for example, is written in pure python, whereas Fujinaga's is written

purely in C++. Nevertheless, some qualitative effects can be seen. Unlike for the other algorithms, the runtime of Carter and Bacon's algorithm does not increase with the square of the resolution, because it does not operate on pixels but on the line adjacency graph. The most expensive individual operation is the computation of the vector field for each pixel, resulting in Roach and Tatem's algorithm being the slowest. The skeleton-based algorithm is the next slowest one because the computation of the skeleton is another quite expensive operation.

We have made all subsequent tests with a resolution of 300 dpi, because for higher resolutions, the runtime increases considerably without much improvement of the error rate. Moreover, 300 dpi is a commonly used scanning resolution nowadays.

6.2 Performance on Undeformed Images

Table 6 shows the results with respect to the paired model analysis described in Section 5.4 with significant differences marked with bold face. Some conclusions can be drawn from this table:

- Both for segmentation and pixel error, Roach and Tatem's original algorithm is significantly the poorest. This is due to false positives (see Fig. 3), because with respect to the interruption error there is no significant difference between this algorithm and others.
- The "secondchord" approach for the "linetracking" method is not worth the additional runtime. With respect to the pixel error, it even worsens the removal quality (because the symbols extend further into the staff region); otherwise, the difference is insignificant.
- With respect to the interruption error, Carter and Bacon's algorithm is the best. Otherwise, there are no significant differences.
- With respect to the segmentation error, Fujinaga's algorithm performs poorer than most other methods. The new skeletonization-based method does not perform better than any other method (except Roach and Tatem's of course).
- From all algorithms except Roach and Tatem's, there is none significantly better than another both with respect to pixel and segmentation error.

As Roach and Tatem's original algorithm performs significantly poorer than all other algorithms, we have only included its improved version in subsequent figures and tables.

More detailed information about the weaknesses of the individual algorithms can be drawn from the samples shown in Fig. 11. Most problematic are symbols that touch or cross stafflines at angles below 45 degrees, like slurs, white notes, the tablature letter "d," or bass clefs. For the runlength-based line tracking, Fujinaga's and Roach and Tatem's algorithm incorrectly remove the line crossing parts of these symbols. The secondchord-based line-tracking algorithm fills some of these holes but also adds parts of the stafflines as artefacts to the images, which even result in a worse pixel error rate (see Table 6). Both Carter and Bacon's and the skeleton-based approach keep more of these symbols intact, with Carter and Bacon's algorithm being the only one that correctly leaves whole notes intact. A particular problem of Carter and Bacon's algorithm however is that it occasionally misses entire staffs that contain a line without a crossing symbol, which can easily happen in bar-less music (quite common in historic notation).

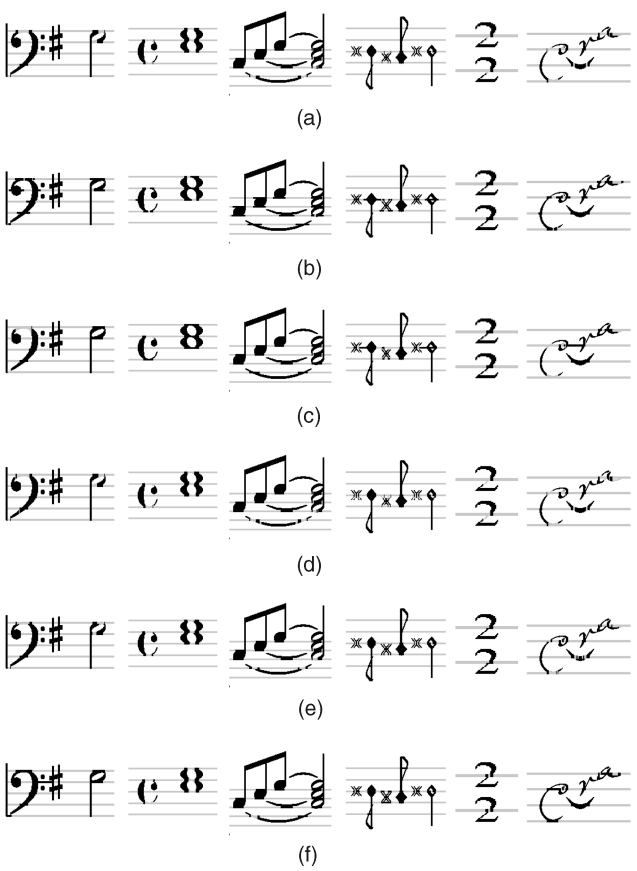


Fig. 11. Details of the results of different staff removal algorithms on our undeformed test set. Removed pixels are marked gray. The details show (from left to right) the bass clef followed by sharp and half note, cut time symbol followed by a chord of two whole notes, beamed eighth notes tied to a chord of half notes, sixteenth century music typeface, modern guitar tablature, and the seventeenth century French lute tablature. (a) Linetracking Runlength. (b) Linetracking Secondchord. (c) Carter and Bacon. (d) Fujinaga. (e) Roach and Tatem (improved). (f) Skeleton.

6.3 Effects of Deformations

The effects of the different deformations over the respective parameter ranges are shown in Figs. 12 and 13. It turned out in our experiments that the qualitative effects of the deformations on the error rates are similar for all three error metrics. Hence, we have used those error metrics in the plots for which the qualitative effect is best visible. Whenever we say in the following that one algorithm performs "better" than another, this is meant with respect to a 5 percent significance level in the paired model.

With respect to rotation and curvature, our new skeleton-based approach is the most robust and performs better than all other algorithms for rotation angles between 5 and 17 degrees and curvatures greater or equal than an amplitude of 0.04 per staff width (this corresponds to a

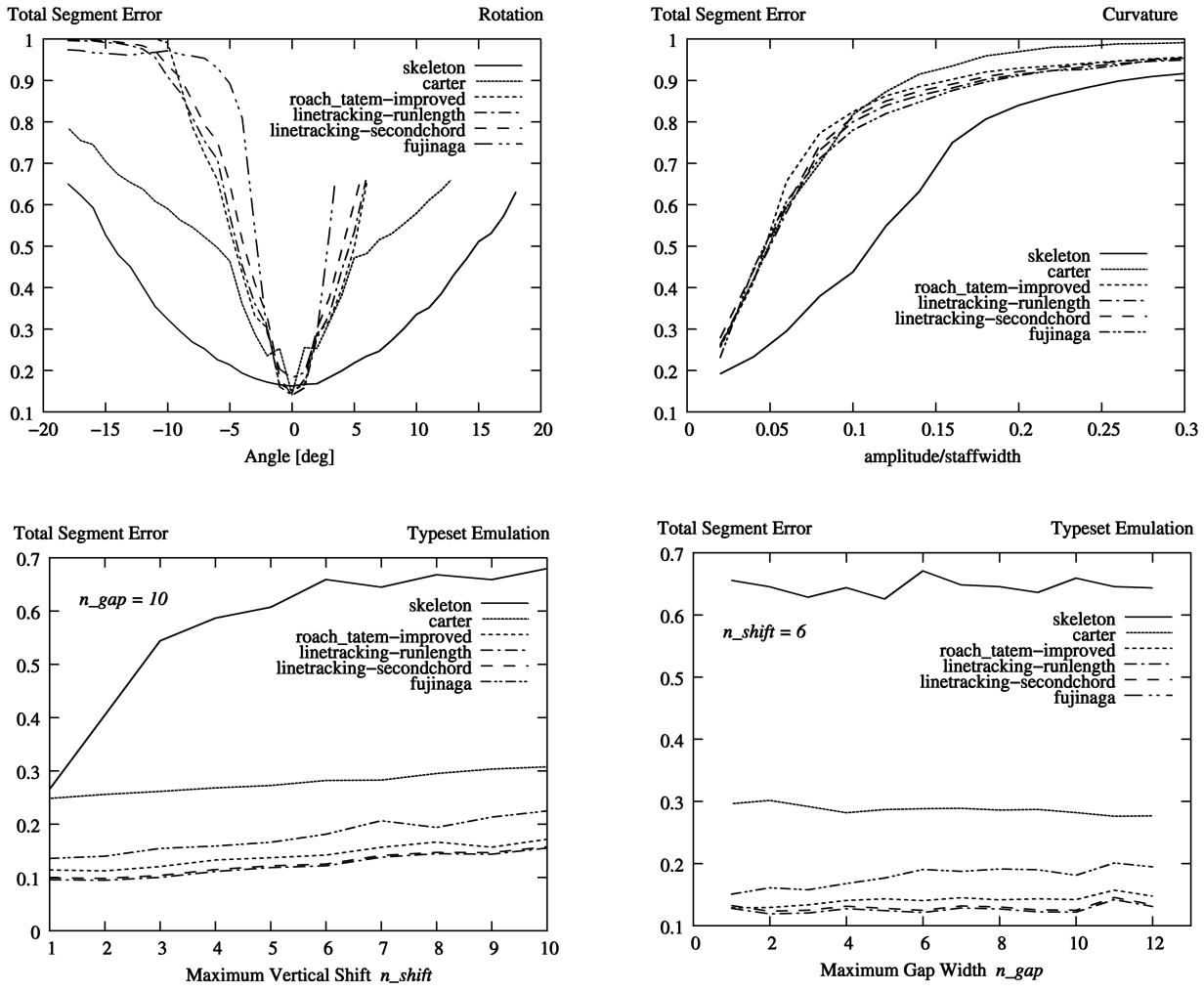


Fig. 12. Effect of different deformations on the overall error rates (from left to right and top to bottom): rotation, curvature, typeset emulation for fixed *n_gap*, and fixed *n_shift*.

curvature angle of $\tan\alpha \geq 0.04$ or $\alpha \geq 2.3$). It is interesting to note that robustness with respect to rotation does not necessarily imply robustness with respect to curvature: Carter and Bacon's method is the second best for rotations greater than 5 degrees but not for any curvature value.

For typeset emulation, it is shown in Fig. 12 that vertical shifts (see the plot with fixed *n_gap*) have a more severe effect on staff removal performance than staffline gaps (see the plot with fixed *n_shift*). This is due to the fact that discontinuous vertical jumps make an extrapolation of staffline segments more difficult than a gap in an otherwise perfectly horizontal staffline. The skeleton-based approach is the poorest for maximal shift widths greater than one, that is, even for rather small deformations.

With respect to thickness variation all algorithms perform poorer for higher variations, but no algorithm is clearly more robust than the others. The oscillating performance of "linetracking runlength" between odd and even values for the maximum thickness in Fig. 13 is due to our threshold of $2 * staffline\_height$ for keeping or removing pixels, which leads to more falsely kept staffline slices for certain combinations of the maximum thickness and the most frequent black vertical runlength.

When the vertical position of each staffline varies randomly ("y-variation"), the skeleton-based approach and Fujinaga's algorithm are the most robust, with the skeleton method as the best for maximum deviations n between 2 and 8, and Fujinaga's method the second best for n between 3 and 5.

For staffline interruptions, we have found that the gap width has little effect on the error rates (no figure included), which is consistent with the results for typeset emulation. The interruption frequency α however has an effect on the error rates. As can be seen in Fig. 13, Carter and Bacon's algorithm is most susceptible to interruptions and performs poorer than all other algorithms for an interruption probability α greater or equal than 0.02 per staffline skeleton pixel.

To visualize the effect of white speckles, we have plotted the error rates over the rate of whitened pixels, because this is a more intuitive parameter than the parameters (n, p, k) of the deformation algorithm. As can be seen in Fig. 13, each algorithm breaks down at a certain whitening rate, with Carter and Bacon's algorithm breaking down first and the skeleton approach last.

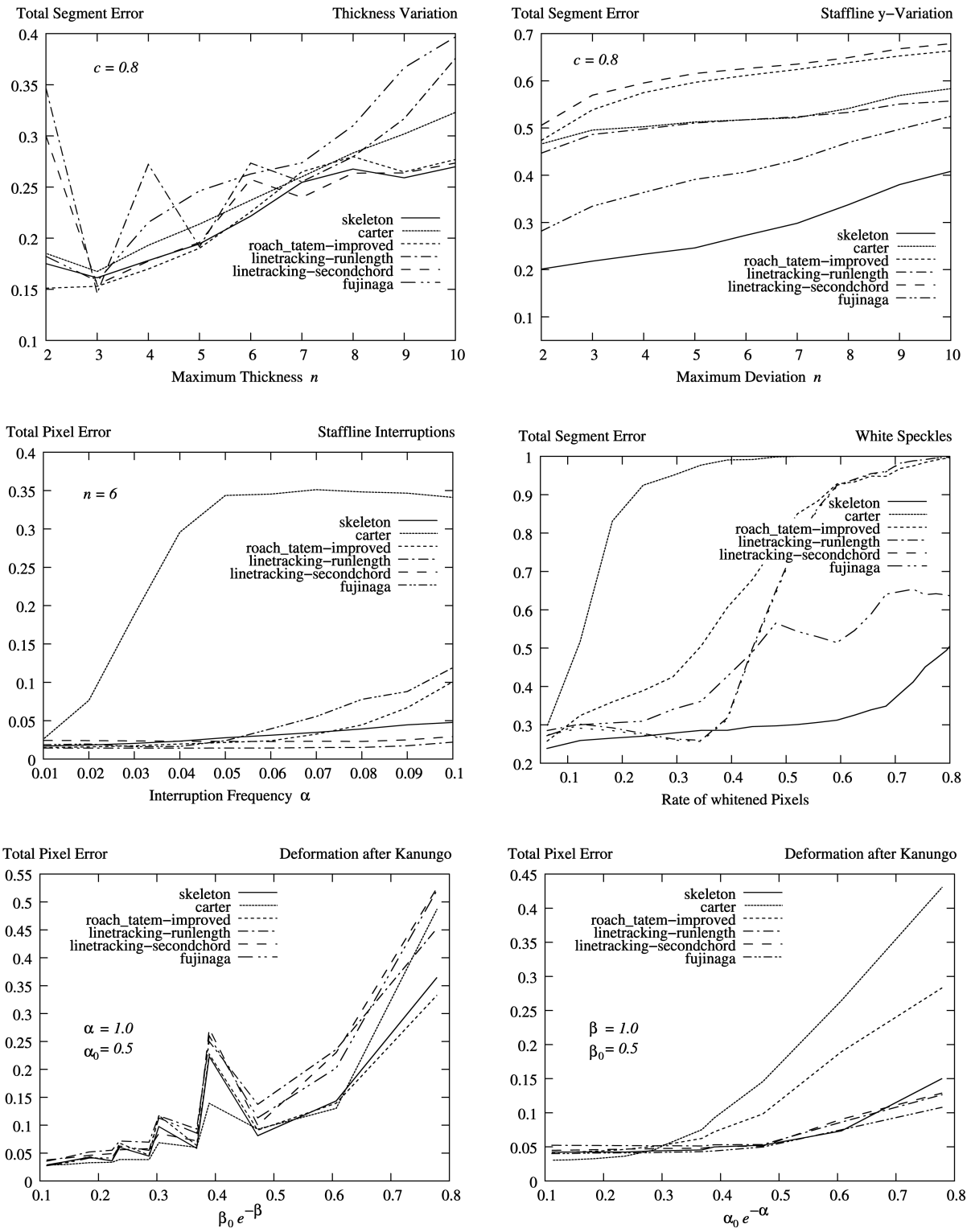


Fig. 13. Effect of different deformations on the overall error rates (continued) (from left to right, top to bottom): staffline thickness variation for fixed inertia factor c, staffline y-position variation for fixed inertia factor c, staffline interruptions for fixed maximum gap width n, white speckles, deformation after Kanungo with fixed foreground parameters (α, α_0) (left) and fixed background parameters (β, β_0) (right).

For the outline deformation after Kanungo et al., two different parameter projections are shown in Fig. 13: The case of fixed foreground flipping probabilities (α, α_0) or fixed background flipping probabilities (β, β_0). With respect to increasing background flipping (left figure), no algorithm is significantly more robust than another. With respect to increasing foreground flipping (right figure), Carter and Bacon's algorithm performs poorest for values $\alpha_0 e^{-\alpha} > 0.4$. This is consistent with the results for speckles and interruptions, in which Carter and Bacon's algorithm is also the most sensitive with respect to whitened foreground pixels.

7 Conclusions

Our results show that there is no clearly best algorithm with respect to all three error metrics. Although our new skeleton-based algorithm is most robust with respect to some defects, it is most susceptible to the typeset emulation of historic prints and does not perform significantly better on the undeformed test set.

When using the discussed algorithms for staff removal outside the realm of music notation, for example, for banking cheques, it should be kept in mind that all algorithms rely on an accurate estimation of *staffline_height* and *staffspace_height*. Although the former is always well defined, the latter only makes sense when there are groups of parallel stafflines. For single stafflines, which can also occur in music notation, for example, for percussion, none of the discussed algorithms works without changes. It seems to us that Carter and Bacon's algorithm might be the best starting point in such a situation because it directly yields filaments as staff segment candidates without a prior determination of staff positions.

Concerning the different performance metrics, it is interesting to observe that the qualitative behavior of the performance under deformations was very similar under all three metrics. This leads us to the conclusion that the exact definition of the used performance metric is of less importance than we had thought initially. We would expect the same to hold for the evaluation of algorithms for different segmentation problems under image degradations.

Our method for the creation of ground-truthing data from a vector image in the Postscript format by first using information only present in this representation for a "perfect" segmentation and then rasterizing and deforming the segmented images has proven to be very useful. It enabled us to fully automate the background/foreground labeling step that can be time consuming in segmentation algorithm comparison studies. It certainly has the potential to be used in other segmentation contexts as well, for example, text/graphics separation or page segmentation.

As we make the full source code of our evaluation framework freely available, it can be utilized to test new staff removal algorithms or to improve existing algorithms. As many algorithms also contain adjustable parameters (for example, thresholds), the framework can also be used to optimize these parameters with respect to different error metrics. Our work can thus help to improve the quality of existing optical music recognition systems. Moreover, it can be used as a starting point for building other segmentation evaluation frameworks.

Acknowledgments

The authors are grateful to Thomas Karsten and Florian Pose for implementing the vector field computation and the "line-tracking" algorithm. Moreover, they thank the Mutopia project (www.mutopiaproject.org), the Choral Public Domain Library (www.cpdl.org), Philip Hazel, Laura Conrad, Johann Tufvesson, Moriwaki Michio, Richard Civiol, and Alain Veylit for providing music samples, either by direct contribution or by posting them on the Internet under a license that allowed their inclusion in the authors' public test set. They would also like to thank the anonymous reviewers for their helpful comments. The authors are also grateful to the US National Science Foundation, Institute for Museum and Library Services, the Lester S. Levy Family, Canadian Foundation for Innovation, and the Social Sciences and Humanities Research Council of Canada for their financial support. M. Droettboom performed this work while at The Digital Knowledge Center of the Johns Hopkins University, Baltimore, Maryland.

References

[1] D. Blostein and H.S. Baird, "A Critical Survey of Music Image Analysis," *Structured Document Image Analysis,* H.S. Baird, H. Bunke, and K. Yamamoto, eds., pp. 405-434, Springer, 1992.

[2] Y.Y. Tang, C.Y. Suen, C. Deyan, and M. Cheriet, "Financial Document Processing Based on Staff Line and Description Language," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 25, no. 5, pp. 738-754, 1995.

[3] D. Bainbridge and T.C. Bell, "Dealing with Superimposed Objects in Optical Music Recognition," *Proc. Sixth Int'l Conf. Image Processing and Its Applications,* pp. 756-760, 1997.

[4] M. Thulke, V. Margner, and A. Dengel, "A General Approach to Quality Evaluation of Document Segmentation Results," *Lecture Notes in Computer Science,* vol. 1655, pp. 43-57, Springer, 1999.

[5] S. Mao and T. Kanungo, "Empirical Performance Evaluation Methodology and Its Application to Page Segmentation Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 3, pp. 242-256, Mar. 2001.

[6] T.K. Ho and H.S. Baird, "Evaluation of OCR Accuracy Using Synthetic Data," *Proc. Fourth Ann. Symp. Document Analysis and Information Retrieval,* pp. 413-422, 1995.

[7] M. Droettboom et al., *The Gamera Project Homepage,* http://gamera.sourceforge.net, 2003-2007.

[8] C. Dalitz et al., *Staff Line Removal Toolkit for Gamera,* http://music-staves.sourceforge.net, 2005-2007.

[9] M. Vuilleumier Stückelberg and D. Doermann, "On Musical Score Recognition Using Probabilistic Reasoning," *Proc. Fifth Int'l Conf. Document Analysis and Recognition,* p. 115, 1999.

[10] J.R. McPherson, "Introducing Feedback into an Optical Music Recognition System," *Proc. Third Int'l Conf. Music Information Retrieval,* pp. 259-260, 2002.

[11] H. Miyao and M. Okamoto, "Stave Extraction for Printed Music Scores Using DP Matching," *J. Advanced Computational Intelligence and Intelligent Informatics,* vol. 8, no. 2, pp. 208-215, 2004.

[12] M. Szwoch, "A Robust Detector for Distorted Music Staves," *Lecture Notes in Computer Science,* vol. 3691, pp. 701-708, Springer, 2005.

[13] L. Pugin, "Optical Music Recognition of Early Typographic Prints using Hidden Markov Models," *Proc. Seventh Int'l Conf. Music Information Retrieval,* pp. 53-56, 2006.

[14] I. Fujinaga, "Staff Detection and Removal," *Visual Perception of Music Notation,* S. George, ed., pp. 1-39, Idea Group, 2004.

[15] J.W. Roach and J.E. Tatem, "Using Domain Knowledge in Low-Level Visual Processing to Interpret Handwritten Music: An Experiment," *Pattern Recognition,* vol. 21, pp. 33-44, 1988.

[16] P. Martin and C. Bellissant, "Low-Level Analysis of Music Drawing Images," *Proc. First Int'l Conf. Document Analysis and Recognition,* pp. 417-425, 1991.

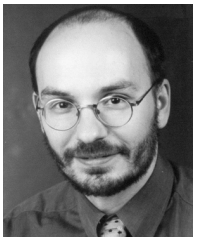
[17] N.P. Carter and R.A. Bacon, "Automatic Recognition of Printed Music," *Structured Document Image Analysis,* H.S. Baird, H. Bunke, and K. Yamamoto, eds., pp. 454-465, Springer, 1992.

[18] K. Ng, "Music Manuscript Tracing," *Lecture Notes in Computer Science,* vol. 2390, D. Blostein, and Y.B. Kwon, eds., pp. 330-342, Springer 2002.

[19] R. Randriamahefa, J.P. Cocquerez, F. Pépin, and S. Philipp, "Printed Music Recognition," *Proc. Second Int'l Conf. Document Analysis and Recognition,* pp. 898-901, 1993.

[20] W.K. Hastings, "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika,* vol. 57, no. 1, pp. 97-109, 1970.

[21] T. Kanungo, R.M. Haralick, and H.S. Baird et al., "A Statistical, Nonparametric Methodology for Document Degradation Model Validation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1209-1223, Nov. 2000.
[22] J. Kanai, "Automated Performance Evaluation of Document Image Analysis Systems: Issues and Practice," *Int'l J. Imaging Systems and Technology*, vol. 7, pp. 363-369, 1996.
[23] Z. Galil, "Efficient Algorithms for Finding Maximum Matchings in Graphs," *ACM Computing Surveys*, vol. 18, no. 1, pp. 23-38, 1986.



Christoph Dalitz received the diploma and the PhD degree in physics from the University of Bielefeld, Germany, in 1993 and 1997, respectively. From 1997 to 1999, he designed and developed optical archiving systems for the Comline Company in Dortmund, Germany, and from 2000 to 2001, he developed information systems for the company Infotech in Recklinghausen, Germany. Since 2002, he has been a professor at the University of Applied Sciences, Krefeld, Germany, where he teaches mathematics and computer science.

Michael Droettboom received the master's degree in computer music research from the Johns Hopkins University. He is the lead developer of the Gamera framework for document image recognition (http://gamera.sourceforge.net/). He currently works as a software engineer at the Space Telescope Science Institute, improving their research software tools.

Bastian Pranzas (formerly Czerwinski) received the diploma in computer engineering from the University of Applied Sciences in Krefeld, Germany, recently.

Ichiro Fujinaga received the BSc degree in mathematics and the BMus degree in percussion from the University of Alberta in 1979 and 1982, respectively, and the MA degree in music theory and the PhD degree in music technology from McGill University in 1988 and 1997, respectively. He was a faculty member of the Computer Music Department, Peabody Conservatory of Music, Johns Hopkins University. From 2002 to 2003, he was the chair of the music technology area at the School of Music. From 2003 to 2004, he was the acting director of the Center for Interdisciplinary Research in Music Media and Technology (CIRMMT), McGill University. He is currently an associate professor in music technology area at the Schulich School of Music, McGill University.