

Staff Detection with Stable Paths

Jaime dos Santos Cardoso, Member, IEEE, Artur Capela, Ana Rebelo, Carlos Guedes, and Joaquim Pinto da Costa

Abstract—The preservation of musical works produced in the past requires their digitalization and transformation into a machine-readable format. The processing of handwritten musical scores by computers remains far from ideal. One of the fundamental stages to carry out this task is the staff line detection. We investigate a general-purpose, knowledge-free method for the automatic detection of music staff lines based on a stable path approach. Lines affected by curvature, discontinuities, and inclination are robustly detected. Experimental results show that the proposed technique consistently outperforms well-established algorithms.

Index Terms—Music, optical character recognition, document image processing, image analysis.



1 INTRODUCTION

PRINTED documents and handwritten manuscripts deteriorate over time, thus causing a significant amount of information to be permanently lost. Among such perishable documents, musical scores are especially problematic. Digitization has been commonly used as a possible tool for preservation, offering easy duplications, distribution, and digital processing. However, to transform the paper-based music scores and manuscripts into a machine-readable symbolic format (facilitating operations such as search, retrieval and analysis), an Optical Music Recognition (OMR) system is needed. This justifies the research on reliable OMR algorithms. In these algorithms, the reason for detecting and removing the staff lines lies on the need to isolate the musical symbols for a more efficient and correct detection of each symbol presented on the score. Although their primary application is as a preprocessing step in the recognition of music notation, the line detection problem also occurs in different contexts, for example, the recognition of bank transfer forms.

The detection of staves is complicated due to a variety of reasons. The handwritten staff lines are rarely straight and horizontal and are not parallel to each other. These scores tend to be rather irregular and are determined by a person's own writing style. Moreover, if we consider that most of these works are old, the quality of the paper on which it is written might have degraded over the years, making it more difficult to correctly identify its contents. Although the problem of detecting staff lines is exacerbated with old and handwritten works, it is also present in recent printed scores.

In this paper, a method for the automatic detection of staff lines is presented. The proposed paradigm uses the image as a graph, where the staff lines are considered as connected paths between the two margins of the image. In a preliminary study [1], [2], we proposed a new learning methodology, which is extended and explored in various directions in this paper. First, the concept of stable path is introduced in order to improve the computational performance of the method. Second, the design of the weights on the graph resulting from the music score is generalized to differentiate black pixels belonging to the staff lines from black pixels resulting from the music symbols. Finally, the postprocessing is refined, improving the overall performance. A further development is the study of new staff removal algorithms by incorporating the proposed staff line detection on standard staff removal algorithms. The experimental work reported at the end of the communication includes a thorough testing on synthetic and real scores, with the latter manually groundtruthed.

1.1 Related Works

The problem of staff line detection is often considered simultaneously with the goal of their removal, although exceptions exist [3], [4]. The simplest approach consists of finding local maxima in the horizontal projection of the black pixels of the image [5]. These local maxima represent line positions, assuming straight and horizontal lines. Several horizontal projections can be made with different image rotation angles, thus eliminating the assumption that the lines are always horizontal. An alternative strategy for identifying staff lines is to use vertical scan lines [6], [7]. Recent works present a more or less sophisticated use of a combination of projection techniques to improve the basic approach [8].

Other techniques for finding staff lines include the application of mathematical morphology algorithms [9], rule-based classification of thin horizontal line segments [10], and line tracing [11], [12]. The methods proposed in [3], [4] operate on a set of “staff segments,” with methods for linking two segments horizontally and vertically and merging two segments with overlapping positions into one. In [13], Dalitz et al. improve on these methods.

In spite of the variety of methods available, they all suffer from some limitations. In particular, lines with some curvature or discontinuities are inadequately resolved. The dash detector [14] is one of the few works that try to handle discontinuities. The dash detector is an algorithm that searches the image, pixel by pixel, finding black pixel regions that it classifies as stains or dashes. Then, it tries to unite the dashes to construct lines.

A problem common to all the above-mentioned techniques is that they try to build staff lines from local information, without properly incorporating global information in the detection process. To our knowledge, none of the proposed methods in the literature tries to define a reasonable process from the intrinsic properties of staff lines, namely, the fact that they are the only extensive black objects on the music score. Generally, we argue that the most interesting techniques arise when one defines the detection process as the result of optimizing some global function. In the following, we suggest a graph-theoretic framework where staff lines are the solutions of a global optimization process.

2 A STABLE PATH APPROACH FOR STAFF LINE DETECTION

In the work to be detailed, the image grid is considered as a graph with pixels as nodes and arcs connecting neighboring pixels. The weight of each arc, $w_{\vec{p}}$, is a function of pixels values and pixels relative positions. A path from vertex (pixel) v_1 to vertex (pixel) v_n

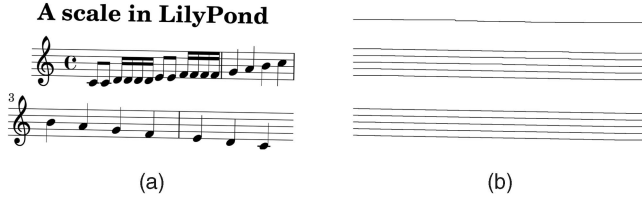


Fig. 1. An illustrative example of the methodology. (a) Skewed staff lines with music. (b) The first 11 shortest paths between left and right margins.

is a list of unique vertices $v_1; v_2; \dots; v_n$, with v_i and v_{i+1} corresponding to neighbor pixels p_i, p_{i+1} . The total cost of a path is the sum of each arc weight in the path $\sum_{i=1}^n w_{v_i, v_{i+1}}^p$.

A path from a source vertex v to a target vertex u is said to be a shortest path if its total cost is minimum among all v -to- u paths. The distance between a source vertex v and a target vertex u on a graph, $d_{\delta v; u}^p$, is the total cost of a shortest path between v and u .

A path from a source vertex v to a subgraph Ω_2 is said to be a shortest path between v and Ω_2 if its total cost is minimum among all v -to- Ω_2 paths. The distance from a node v to a subgraph Ω_2 , $d_{\delta v; \Omega_2}^p$, is the total cost of a shortest path between v and Ω_2 .

$$d_{\delta v; \Omega_2}^p = \min_{u \in \Omega_2} d_{\delta v; u}^p$$

A path from a subgraph Ω_1 to a subgraph Ω_2 is said to be a shortest path between Ω_1 and Ω_2 if its total cost is minimum among

all Ω_1 -to- Ω_2 paths. The distance from a subgraph Ω_1 to a subgraph Ω_2 , $d_{\delta \Omega_1; \Omega_2}^p$, is the total cost of a shortest path between Ω_1 and Ω_2 :

$$d_{\delta \Omega_1; \Omega_2}^p = \min_{v_1 \in \Omega_1, v_2 \in \Omega_2} d_{\delta v_1; v_2}^p$$

2.1 Algorithm Outline

To a first approximation, staff lines can be considered as the only extensive objects made from black pixels in the music score, connected paths of black pixels from the left side to the right side of the music score. Assuming that paths through black pixels are preferred over paths through white pixels, staff lines can then be found among the shortest paths from the left to the right margin of the music score. Staff lines are then best modeled as paths between two regions Ω_1 and Ω_2 , the left and right margins of the score.

One may assume that staff lines do not zigzag back and forth, left and right. Therefore, one may restrict the search among connected paths containing one and only one pixel in each column of the image.¹ Formally, let I be an $N_1 \times N_2$ image and define an admissible staff to be

$$s = \{x_j\}_{j=1}^{N_2}, \text{ s.t. } x_j \in \Omega_1, \text{ s.t. } x_j \neq x_{j+1} \text{ for } j=1, \dots, N_2-1$$

where y is a mapping $y: \Omega_1 \rightarrow \Omega_2$. That is, a staff line is an 8-connected path of pixels in the image from left to right, containing one and only one pixel in each column of the image.

Given the weight function $w_{\delta p; q}^p$, the cost of a staff can be defined as $C_{\delta s}^p = \sum_{j=1}^{N_2} w_{\delta x_j; x_{j+1}}^p$. The optimal staff line that

$$s^* = \arg \min_s C_{\delta s}^p$$

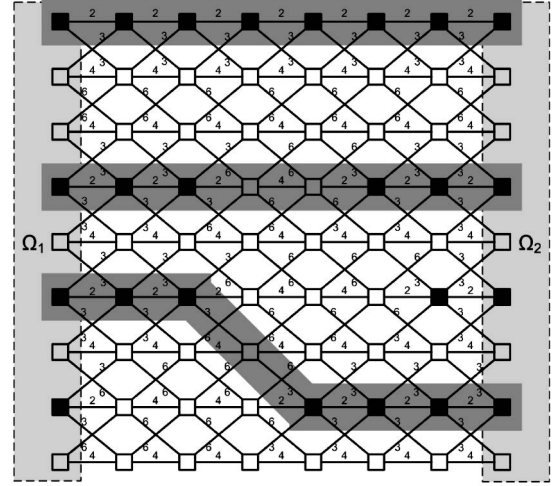


Fig. 2. Stable paths on a toy example.

$$C_{\delta \Omega_1; \Omega_2}^p = \min_{v_1 \in \Omega_1, v_2 \in \Omega_2} C_{\delta v_1; v_2}^p = \min_{v_1 \in \Omega_1, v_2 \in \Omega_2} \sum_{i=1}^n w_{v_i, v_{i+1}}^p$$

where $w_{\delta p; q}^p$ represents the weight of the edge incident with pixels at positions $\delta; j$ and $\delta; m$. At the end of this process,

$$\min_{j \in \{1, \dots, N_2\}} C_{\delta \Omega_1; j}^p$$

indicates the end of the minimal connected staff. Hence, in the second step, one backtrack from this minimum entry on C to find the path of the optimal staff.

Assume that one wants to find all staff lines present in a score. This can be approached by successively finding and erasing the shortest path from the left to the right margin of the score. The erase operation is required to ensure that a staff is not detected

multiple times.

Consider the music score presented in Fig. 1a; in Fig. 1b, the first 11 shortest paths are traced. This example shows that music symbols placed on top of staff lines do not interfere with the detection of the staff lines. Moreover, the example also makes clear that slightly skewed scores do not pose any problem to the proposed approach.

Before presenting the complete algorithm, we introduce here for the first time the concept of a stable path in a graph, which will allow computing multiple staff lines in a single iteration instead of sequentially computing them one at a time.

2.2 Stable Paths on a Graph

Before moving to the formal definition of a stable path on a graph, it is instructive to motivate the concept by considering a hypothetical, simplified music score, with four staff lines (rows 1,

4, 6, and 8) in a 8×9 image. The staff lines comprise mostly black

elements; the discontinuities on some of the staff lines try to simulate the presence of noise. The graph corresponding to such score is represented in Fig. 2. The design of the weight function will be considered next; for now, it suffices to know that it was constructed to favour paths through black pixels.

The shortest path between the left and right margins (sub-

minimizes this cost can be found using dynamic programming. The first step is to traverse the image from the second column to the last column and compute the cumulative minimum cost C for all possible connected staff lines for each entry $\delta_i; j$:

1. These assumptions, 8-connectivity and one pixel per column, impose a maximum detectable 45 rotation degrees.

graphs 1 and 2) is the path corresponding to the first row, entirely through black pixels. By following the strategy just delineated, one could find the four staff lines in four iterations, sequentially. Nonetheless, although only one staff line corresponds to the shortest path, they all constitute a sort of (almost) optimal paths. The stable path concept provides a means to find all of such paths simultaneously.

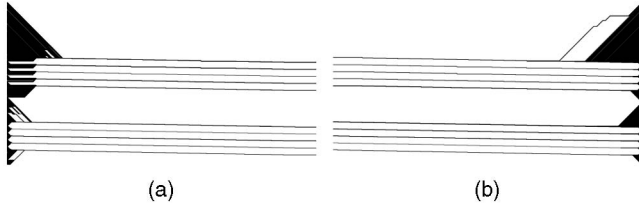


Fig. 3. Illustration of stable paths for Fig. 1a. (a) Shortest paths from each pixel in the left column and the whole right column, superimposed on the original image. (b) Shortest paths from each pixel in the right column and the whole left column, superimposed on the original image.

Definition. A path $P_{s,t}$ is a stable path between regions r_1 and r_2 if $P_{s,t}$ is the shortest path between $s \in r_1$ and the whole region r_2 , and $P_{s,t}$ is the shortest path between $t \in r_2$ and the whole region r_1 .

The naming of stable path has its roots in dynamical systems, as it resembles stable fixed points. If one considers the function $F_{r_1, r_2}(\delta P)$, mapping a node $s \in r_1$ to a node $t \in r_2$ by finding the shortest path $P_{s,t}$ between $s \in r_1$ and r_2 , with $t \in r_2$ as the end node of such shortest path, then

$$G_{r_1, r_2}(\delta P) = F_{r_1, r_2}(\delta P) \text{ if and only if } P_{s,t} \text{ is a stable path.}$$

Note that the concept of stable path is valid for any graph and any two subgraphs in general. The computation of the stable paths on the toy example of Fig. 2 provides the three paths yellow highlighted in the figure.

As a second example, in Fig. 3a the shortest paths between each point on the left margin and the whole right margin are traced for the score in Fig. 1a. As seen, the paths got attracted by the staff lines. Likewise, Fig. 3b shows the shortest paths between each point on the right margin and the whole left margin. The set of stable paths between both margins result as the set of paths present in both figures.

Although the computation of the stable paths may be expensive in general graphs, the computation in the graph derived from an image under the setting adopted in Section 2.1 has only roughly twice the complexity of the shortest path computation presented in the same section. Noticing that the procedure delineated in Section 2.1 actually gives the shortest path between the whole left margin r_1 and each point on the right margin r_2 , the first step on the computation of the stable path corresponds verbatim to the computation of the shortest path presented on Section 2.1. In a second step, one repeats the same procedure, now traversing the graph from the right column to the left. At the end of this process, if the two endpoints of a direct and reverse path coincide, we are in the presence of a stable path.

The stable paths on the toy example of Fig. 2 provide only three out of the four staff lines, with the last stable path following partially through a segment of the third staff line and a segment of the fourth staff line.² Therefore, the computation of the stable paths is not guaranteed to find all paths of interest. The application of the stable path procedure a second time, after erasing the paths found on the first iteration, yields a new set of paths containing a path joining the remaining segments of the third and fourth staff lines. The advantage of this search based on stable paths over the sequential search of the shortest paths depends on the number of stable paths found simultaneously. While a score with 60 staff lines would require 60 iterations of the shortest path algorithm, it requires a few (typically between four and six) iterations with the stable path method. Next, the complete proposed algorithm for staff line detection is detailed.

2. Note that the sequential search of the shortest path would suffer from the same limitations.

2.3 Proposed Algorithm

The proposed algorithm can be implemented as a sequence of a few high-level operations, as presented in Listing 1.

Listing 1. Main operations of the proposed method.

Preprocessing:

compute staffspaceheight and stafflineheight
compute weights of the graph

Main Cycle:

compute stable paths
validate paths with blackness and shape
erase valid paths from image
add valid paths to list of stafflines
end of cycle if no valid path was found

Postprocessing:

uncross stafflines
organize stafflines in staves
smooth and trim stafflines

2.3.1 Preprocessing

To detect the staff lines, the proposed algorithm starts by estimating the staff space height, staffspaceheight, and staff line height, stafflineheight. These lengths are used as reference lengths on subsequent operations. Robust estimators are already in common use: the technique starts by computing the vertical run-lengths representation of the image. If a bit-mapped page of music is converted to vertical run-lengths coding, the most common black runs represents the staff line height and the most common white runs represents the staff space height [8]. After estimating the reference lengths, the edges' weights are estimated as explained in the next section.

2.3.2 Main Cycle

The preprocessing is followed by the main cycle of the methodology, by successively finding the stable paths between the left and right margins, adding the paths found to the list of staff lines, and erasing them from the image. The erase operation sets to white the pixels on a vertical strip centred on the detected staff line. The erase operation is necessary to ensure that a line is not detected multiple times, even if its height is higher than one pixel. The height of the strip was empirically fixed at staffspaceheight.

2.3.3 Stopping Rule

To stop the iterative staff line search, a sequence of (arguably) sensible rules is used to validate the stable paths found; if none of them passes the checking, the iterative search is stopped. Two validation rules were applied, both assessing features with respect to the median values obtained during the first iteration. A path is discarded if it does not have a percentage of black pixels above a fixed threshold. The median percentage of blackness of all lines found in the first iteration of the main cycle provides the necessary reference (a threshold of 80% of the median value was empirically selected). Likewise, a path is discarded if its shape differs too much from the shape of the line with median blackness. A dissimilarity—measured as the average y-distance between both paths, after removing the means—above $\text{shapediff} \geq 4 \times \text{staffspaceheight}$ was selected as threshold.

2.3.4 Postprocessing

After the main search step, valid staff lines are postprocessed. Although true staff lines never intersect, the above algorithm may occasionally create intersecting lines, detected on different iterations. Local discontinuities can induce a stable path to zigzag back

and forth between consecutive lines; on the next iteration, the detected path is likely to connect the remaining segments, and therefore, intersect with the path detected in first place. To preclude such final, undesired state, lines are postprocessed to remove intersections: for each image column, sort on y the pixels of the detected lines and assign the i -pixel to the i -line. After this simple process, lines may touch but they do not intersect.

It is now possible to eliminate spurious lines and cluster them in staves. Because lines are ordered, these operations require only iterating through the list of lines and starting a new staff whenever the distance between two consecutive lines is above a fixed threshold ($\frac{1}{4} 2$ staffspaceheight). Next, spurious staves can be eliminated. Although more robust rules can be designed, we are simply discarding sets with a single line.

Finally, lines are smoothed and trimmed. As is noticeable in the example of Fig. 1b, before meeting with a staff line, a path travels through a sequence of white pixels. Likewise, after the end of the staff line, the path goes again through a sequence of white pixels until it meets the right margin of the image. In order to eliminate the undesirable segments, the trimming operation works per staff. For each staff, a sequence of median colours is computed as follows: for each column, the median of the colours (black and white values, as we are working with binary images) of the lines is added to the sequence. Next, the trimming points are found in this sequence: Starting on the center, we traverse the sequence to the left and right until a run of $\frac{1}{4} 2$ staffspaceheight white pixels is found. The pixels between the left and right runs are kept in the staff lines. At the end, lines are smoothed with a standard average low-pass filter. Considering a staff line as a sequence $y \times P$ of y -positions, a one-dimensional averaging filter is applied. A window size of 2 staffspaceheight was selected empirically.

2.4 Design of the Weight Function

An immediate approach is to support the design of the weight function solely on the values of the incident nodes: if any of the corresponding pixels are black, then a low cost is assigned to the edge; otherwise the edge assumes a high cost. In [1], we followed this straightforward approach, with already significant results. We call this the `baseWeight` in Listing 2. Now the weight function is generalized to account to other factors. To incorporate some prior knowledge about a music score into the shortest path process, we consider different alternatives to modify the weight function of the graph.

We considered two more attributes of a pixel to influence the main contribution to the edge's weight resulting from the values of the incident pixels. The prime intention of these additional features is to discriminate black pixels in the staff lines from black pixels in the music symbols, penalizing the latter and favouring the former.

If a black pixel is part of a short vertical run of black pixels, then it is more likely to be part of a staff line rather than of a symbol. Therefore, a term benefiting such edges is included in the weight function. Another prior knowledge is that a staff line is likely to have another staff line at roughly staffspaceheight pixels (assuming that staves have at least two lines). As such, if the nearest vertical run of black pixels on the same column is excessively far from the vertical run of black pixels containing the current black pixel, then this pixel is more likely to belong to a symbol (probably a ligature) rather than to staff line. Consequently, a penalizing term is incorporated in the weight function for these cases. The pseudocode for the weight function is provided in Listing 2.

Listing 2. Pseudocode for the weight function. The base weight was set to 4 on black pixels and 8 on white pixels for 4-neighborhoods, and 6 and 12 on for 8-neighborhoods. The delta penalizing term in the weight function was set to 1. For efficiency, weights were designed with integer values.

```
WeightFunction(pixelValue1, pixelValue2, vRun1, vRun2,
    nearestVRun1, nearestVRun2, NeighbourhoodType)
{
    value = min(pixelValue1, pixelValue2);
    weight = baseWeight (value, NeighbourhoodType);
    if((vRun1 <= STAFFLINEHEIGHT)
        OR (vRun2 <= STAFFLINEHEIGHT))
        weight = weight - delta;
    if((nearestVRun1 >= STAFFSPACEHEIGHT +
        STAFFLINEHEIGHT)
        OR (nearestVRun2 >= STAFFSPACEHEIGHT +
        STAFFLINEHEIGHT))
        weight = weight + delta;
    return weight;
}
```

3 RESULTS

Although the assessment of a new staff detection algorithm can be done by visually inspecting the output on a set of scores—as adopted in [2]—here, the comparison is supported on quantitative measures. The test set adopted for the qualitative evaluation of the proposed method is the one presented in [13]. The test set consists of 32 ideal scores to which known deformations can be applied. The distortions range from rotation and curvature to typeset emulation and staff line thickness variation—see [13], [15] for more details. In total, 2,688 images were generated from 32 perfect scores. To conveniently measure the performance of a staff line detection algorithm, two different error metrics are considered: the percentage of staff lines falsely detected and the percentage of staff lines missed to detect.

To evaluate these metrics, we start by computing the average euclidian distance between each reference staff line and each actually detected staff line; then we solve the matching problem on the resulting bipartite graph by minimizing the assignment cost (= distance). Only pairs with average error-distance below stafflineheight were assumed correctly matched (the other pairs were assumed to originate from a false positive staff line being matched to an undetected true staff line and were therefore unmatched). Now the two metrics result as the number of unmatched detected staff lines (false positive) and unmatched reference staff lines (missed to detect). It should be noted that these metrics only measure whether staff lines are found, not how good the match is.

The proposed algorithm was compared with the three methods considered in [13] for staff line detection.³ As Dalitz's algorithm performed significantly better than the two others evaluated in [13], we have only included Dalitz's results in subsequent figures. This relative performance of the algorithms for staff line detection agrees with the relative performance for staff line removal reported in [13]. The parameters of the stable path algorithm were preliminary tuned with an independent set of images, yielding the thresholds already presented in the description of the method.

The effects of the different deformations over the respective parameter ranges are shown in Table 1; the shortest path method relates to the version presented in [1], including the weight function and postprocessing. The curvature deformation is done with a half sine wave; the line y -variation generates random defects on the y -position of the staff line; the typeset

3. The source code is available upon request to the authors.

TABLE 1
Effect of Different Deformations on the Overall Staff Detection Error Rates in Percentage:
Average (Standard Deviation) of the False Detection Rate and Miss Detection Rate. See [13] for Parameter Details

Error	Angle	ROTATION					Runtime
		-5	-2.5	0	2.5	5	
Error	Stable path	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.6 (3.5); 0.6 (3.5)	0.7 (3.5); 0.7 (3.5)	1.2 (4.0); 1.2 (4.0)	858 sec.
	Sortest Path	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.6 (3.5); 0.6 (3.5)	0.7 (3.5); 0.7 (3.5)	1.2 (4.0); 1.2 (4.0)	6006 sec.
	Dalitz	17.8 (22.0); 51.7 (38.1)	8.6 (14.0); 15.5 (28.7)	0.0 (0.0); 0.0 (0.0)	4.2 (19.6); 9.8 (29.0)	5.5 (9.3); 37.5 (41.9)	612 sec.
Error	Amplitude/staffwidth	CURVATURE					Runtime
		0.02	0.04	0.06	0.08	0.10	
Error	Stable path	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	1.2 (4.0); 1.2 (4.0)	822 sec.
	Sortest Path	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	1.2 (4.0); 1.2 (4.0)	5554 sec.
	Dalitz	12.7 (29.0); 12.6 (28.7)	53.8 (44.5); 55.7 (45.2)	83.3 (30.6); 83.9 (30.4)	95.8 (17.5); 100.0 (0.0)	96.0 (17.5); 100.0 (0.0)	639 sec.
Error	Rate whitened pixels	WHITE SPECKLE					Runtime
		0.03	0.05	0.07	0.09	0.11	
Error	Stable path	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	0.9 (3.7); 0.9 (3.7)	1.2 (3.8); 1.2 (3.8)	2.1 (4.6); 2.3 (4.8)	809 sec.
	Sortest Path	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	0.9 (3.7); 0.9 (3.7)	1.7 (4.0); 1.9 (4.3)	5.3 (7.4); 7.0 (9.6)	5122 sec.
	Dalitz	0.0 (0.0); 0.0 (0.0)	0.3 (1.4); 0.3 (1.4)	26.7 (25.3); 29.9 (27.2)	89.3 (54.6); 86.9 (25.6)	54.5 (55.9); 95.2 (17.0)	872 sec.
Error	Max deviation, n	LINE Y-VARIATION					Runtime
		2	3	4	5	6	
Error	Stable path	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	1.1 (3.8); 1.1 (3.8)	767 sec.
	Sortest Path	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.7 (3.5); 0.7 (3.5)	0.8 (3.6); 0.8 (3.6)	1.1 (3.8); 1.1 (3.8)	5122 sec.
	Dalitz	31.0 (50.7); 31.7 (46.1)	22.1 (38.9); 32.6 (45.6)	15.7 (27.2); 33.7 (45.0)	13.0 (20.1); 33.7 (45.0)	12.8 (18.6); 34.2 (44.7)	768 sec.
Error	Max gap width, n_{gap}	TYPESET EMULATION I					Runtime
		1	4	7	10	13	
Error	Stable path	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	739 sec.
	Sortest Path	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	0.6 (3.5); 0.6 (3.5)	5085 sec.
	Dalitz	19.7 (34.1); 13.7 (21.6)	21.4 (27.4); 15.4 (17.3)	22.3 (30.0); 17.4 (19.0)	24.2 (38.9); 16.7 (22.0)	31.4 (42.3); 19.2 (20.3)	703 sec.
Error	Max vertical shift, n_{shift}	TYPESET EMULATION II					Runtime
		1	4	7	10	13	
Error	Stable path	0.6 (3.5); 0.6 (3.5)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	886 sec.
	Sortest Path	0.6 (3.5); 0.6 (3.5)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	0.7 (3.8); 0.7 (3.8)	6106 sec.
	Dalitz	3.3 (10.4); 2.3 (7.3)	15.9 (27.1); 12.0 (17.1)	39.0 (48.7); 24.6 (27.3)	48.7 (60.8); 29.3 (30.7)	58.7 (54.9); 37.3 (29.0)	842 sec.

emulation tries to imitate sixteenth century prints with staff lines interruptions between symbols and random vertical shifts [13]. With respect to the distortions considered, the stable path based-approach (and the shortest path approach) outperforms the Dalitz algorithm.⁴ In fact, the performance of our approach is almost independent of intensity of the deformation, for the range of values considered. This performance gain is even more noteworthy as the Dalitz algorithm is receiving as input the correct number of lines per staff. Had not this been the case, the difference between both would have been much larger. The current implementation of the stable path algorithm runs as fast as the Dalitz algorithm (and about five times faster than the shortest path version), as available with the MusicStaves Toolkit [15]. By generalizing the design of the weight function and enhancing the postprocessing, we were able to improve the detection performance on our initial results in [1]; the use of the stable paths allowed us to improve the detection speed.

In a simple analysis of the sensitivity of the approach to the selected parameters, we repeated the experiments with values for stafflineheight and staffspaceheight one pixel above and below the true estimated values. Note that all of the parameters of the method are scaled by one of these two estimates. In both cases, the stable path method presented a good performance and continued to yield the best results (the strongest degradation occurred for the rotation degradation, with angle = 5 degree, where both error rates increased to 1.7 percent).

In a second experiment, we evaluated the staff line detection methods on a set of 40 real music scores, for which reference staff lines were manually outlined. Images were previously binarized with the Otsu threshold algorithm, as implemented in the Gamera project.⁵ Two examples of this set of real scores are presented in Fig. 4. The evaluation of the detection algorithms yielded the results presented in Table 2. Again, although the correct number of staff lines per staff was inputted to the Dalitz' algorithm, the stable path approach obtained the best performance.

4. For the deformations not shown, the stable path is not significantly better than Dalitz.

5. <http://gamera.sourceforge.net>.

3.1 Staff Line Removal

Staff line detection algorithms can be used as a first step in many staff removal algorithms. To understand the potential of our algorithm to leverage the performance of existing staff removal algorithms, we conducted a series of experiments, comparing existing versions of staff line removal algorithms with modified versions of them, making use of the stable path algorithm at the staff line detection step. The quantitative comparison of the different algorithms is totally in line with the comparison presented in [13]. Adopting the naming convention from [13], the following algorithms were adapted: LineTrack Height, LineTrack Chord, Roach/ Tatem. The original version of the algorithms were considered as available in [15], making use of the Dalitz algorithm in the detection phase; the modified versions use instead the stable path for detecting lines. We also adopted the same error metrics (individual pixels, staff-segment regions, and staff interruption location) and conducted the comparative study on the same test set. It turned out that the effects of the deformations and the insertion of the stable path as the detection algorithm are similar for all three error metrics. Hence, in Tables 3 and 4, we just present the results for the pixel error rate and the Line Track Height algorithm (original and modified), plus the Skeleton algorithm, which exhibited a competitive performance in [13]. The Line Track Height (LTH) checks whether the vertical black run through the staff line point is longer than ($\frac{1}{4} 2 \cdot \text{stafflineheight}$).

A first observation is that, overall, the replacement of the Dalitz method by the stable path approach as the staff detection step improved the results in the algorithms under comparison. Additionally, the LineTrack Height algorithm with the stable path consistently outperformed the other algorithms. Nevertheless, the skeleton method [13], which does not have a clear line detection step, continues to present a competitive performance. It is worthwhile to finalize by noticing that the skeleton algorithm is

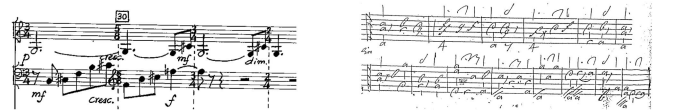


Fig. 4. Two music scores from the set of 40 real music scores used on the experimental evaluation (detail).

TABLE 2
Detection Performance on Real Music Scores in Percentage:
Average (Standard Deviation)

	False detection rate	miss detection rate	Runtime
Dalitz	5.2% (10.4)	5.9% (11.3)	112 sec.
Shortest path	1.4% (3.5)	2.5% (7.3)	612 sec.
Stable path	1.3% (3.7)	1.4% (6.4)	115 sec.

TABLE 3
Effect of Different Deformations on the Overall Staff Removal
Error Rates in Percentage: Average (Standard Deviation)

	ROTATION				
	-5	-2.5	0	2.5	5
Angle					
Stable path + LTH	1.7 (0.7)	1.5 (0.7)	1.4 (0.7)	1.4 (0.7)	1.6 (0.7)
Dalitz + LTH	19.4 (18.4)	5.2 (8.7)	1.4 (0.8)	4.4 (8.8)	17.5 (18.9)
Skeleton	1.9 (0.9)	1.7 (0.8)	1.5 (0.7)	1.6 (0.7)	1.7 (0.8)
	CURVATURE				
	0.02	0.04	0.06	0.08	0.10
Amplitude/staffwidth					
Stable path + LTH	1.4 (0.7)	1.4 (0.7)	1.4 (0.7)	1.5 (0.7)	1.6 (0.7)
Dalitz + LTH	3.8 (5.8)	14.0 (12.2)	22.8 (13.7)	31.1 (11.0)	35.0 (10.6)
Skeleton	2.6 (2.4)	5.2 (5.1)	8.1 (7.2)	11.9 (8.6)	15.4 (10.4)
	WHITE SPECKLE				
	0.03	0.05	0.07	0.09	0.11
Rate whitened pixels					
Stable path + LTH	11.9 (3.1)	17.2 (4.9)	21.1 (5.9)	24.0 (6.7)	26.1 (7.2)
Dalitz + LTH	11.5 (3.2)	16.8 (4.9)	26.7 (8.0)	53.3 (14.9)	73.3 (14.6)
Skeleton	14.6 (3.2)	21.5 (4.6)	27.1 (5.6)	35.2 (12.8)	46.9 (18.7)
	LINE Y-VARIATION				
	2	3	4	5	6
Max deviation, n					
Stable path + LTH	1.2 (0.7)	1.3 (0.7)	1.3 (0.6)	1.4 (0.6)	1.4 (0.6)
Dalitz + LTH	9.0 (13.2)	10.4 (14.1)	10.9 (14.5)	10.9 (14.5)	11.0 (14.6)
Skeleton	1.5 (0.8)	1.7 (0.8)	2.2 (0.9)	3.7 (1.7)	5.2 (2.2)
	TYPESET EMULATION I				
	1	4	7	10	13
Max gap width, n_{gap}					
Stable path + LTH	1.4 (0.7)	1.4 (0.7)	1.4 (0.7)	1.4 (0.7)	1.4 (0.7)
Dalitz + LTH	2.6 (1.8)	2.9 (2.0)	3.2 (1.7)	2.9 (1.7)	3.0 (1.8)
Skeleton	26.4 (9.8)	27.3 (10.1)	27.2 (11.3)	25.5 (9.8)	26.4 (10.3)
	TYPESET EMULATION II				
	1	4	7	10	13
Max vert. shift, n_{shift}					
Stable path + LTH	1.4 (0.7)	1.4 (0.7)	1.4 (0.7)	1.5 (0.7)	1.6 (0.7)
Dalitz + LTH	1.5 (0.8)	2.8 (1.6)	3.3 (2.5)	3.8 (2.4)	4.7 (3.7)
Skeleton	7.9 (8.9)	24.1 (9.1)	26.7 (11.0)	26.1 (9.6)	29.1 (10.7)

TABLE 4
Removal Performance on Real Music Scores (in Percentage):
Average (Standard Deviation)

	Stable path + LTH	Dalitz + LTH	Skeleton
Pixel Error Rate	2.8 (1.2)	3.8 (2.6)	6.5 (8.2)

about two times slower than the modified Line Tracking Height algorithm.

4 CONCLUSION

This paper presented a robust algorithm for the automatic detection of staff lines in music scores. The proposed method uses a very simple but fundamental principle to assist detection and avoid the difficulties typically posed by symbols superimposed on staff lines. The stable path approach for staff line detection algorithm is adaptable to a wide range of image conditions, thanks to its intrinsic robustness to skewed images, discontinuities, and curved staff lines. The proposed approach is robust to discontinuities in staff lines (due to low-quality digitalization or low-quality originals) or staff lines as thin as one pixel.

In order to take full advantage of the method, existing staff line removal algorithms were enhanced by using the stable path method as their first processing step. The integration of the stable path algorithm in the MusicStaves Toolkit has already been reported in [16]. The encouraging results lead us to consider now investigating the detection of music symbols benefiting from the improved staff line detection and removal.

REFERENCES

- [1] J.S. Cardoso, A. Capela, A. Rebelo, and C. Guedes, "A Connected Path Approach for Staff Line Detection," Proc. Int'l Conf. Image Processing, to appear.
- [2] A. Rebelo, A. Capela, J.F.P. da Costa, C. Guedes, E. Carrapatoso, and J.S. Cardoso, "A Shortest Path Approach for Staff Line Detection," Proc. Third Int'l Conf. Automated Production of Cross Media Content for Multichannel Distribution, pp. 79-85, 2007.
- [3] H. Miyao and M. Okamoto, "Stave Extraction for Printed Music Scores Using DP Matching," J. Advanced Computational Intelligence and Intelligent Informatics, vol. 8, pp. 208-215, 2004.
- [4] M. Szwoch, "A Robust Detector for Distorted Music Staves," Computer Analysis of Images and Patterns, pp. 701-708, Springer-Verlag, 2005.
- [5] 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-Verlag, 1992.
- [6] N.P. Carter, "Automatic Recognition of Printed Music in the Context of Electronic Publishing," PhD thesis, Depts. of Physics and Music, Univ. of Surrey, 1989.
- [7] D. Bainbridge, "Extensible Optical Music Recognition," PhD thesis, Dept. of Computer Science, Univ. of Canterbury, Christchurch, New Zealand, 1997. [8] I. Fujinaga, "Staff Detection and Removal," Visual Perception of Music Notation: On-Line and Off-Line Recognition, S. George, ed., pp. 1-39, Idea Group Inc., 2004.
- [9] I. Gawdzki, "Optical Music Scores Recognition," technical report, Laboratoire de Recherche et de Développement de l'Epita, 2002.
- [10] J.V. Mahoney, "Automatic Analysis of Music Score Images," BSc thesis, Dept. of Computer Science and Eng., Massachusetts Inst. of Technology, 1982.
- [11] D. Prerau, "Computer Pattern Recognition of Standard Engraved Music Notation," PhD thesis, Dept. of Computer Science and Eng., Massachusetts Inst. of Technology, 1970.
- [12] 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, no. 1, pp. 33-44, 1988.
- [13] C. Dalitz, M. Droettboom, B. Czerwinski, and I. Fujigana, "A Comparative Study of Staff Removal Algorithms," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 30, no. 5, pp. 753-766, May 2008.
- [14] I. Leplumey, J. Camillerapp, and G. Lorette, "A Robust Detector for Music Staves," Proc. Int'l Conf. Document Analysis and Recognition, pp. 902-905, 1993.
- [15] C. Dalitz, M. Droettboom, B. Czerwinski, and I. Fujigana, "Staff Removal Toolkit for Gamera, 2005-2007," <http://music-staves.sourceforge.net>, 2009. [16] A. Capela, A. Rebelo, J.S. Cardoso, and C. Guedes, "Staff Line Detection and Removal with Stable Paths," Proc. Int'l Conf. Signal Processing and Multimedia Applications, pp. 263-270, 2008.