2014 14th International Conference on Frontiers in Handwriting Recognition

# A* Path Planning for Line Segmentation of Handwritten Documents

Olarik Surinta, Michiel Holtkamp, Faik Karabaa, Jean-Paul van Oosten, Lambert Schomaker and Marco Wiering
Institute of Artificial Intelligence and Cognitive Engineering, University of Groningen
Nijenborgh 9, Groningen, The Netherlands
Email: {o.surinta, m.j.holtkamp, m.f.karabaa, j.p.van.oosten, l.r.b.schomaker, m.a.wiering}@rug.nl

***Abstract*—This paper describes the use of a novel $A^*$ path-planning algorithm for performing line segmentation of handwritten documents. The novelty of the proposed approach lies in the use of a smart combination of simple soft cost functions that allows an artificial agent to compute paths separating the upper and lower text fields. The use of soft cost functions enables the agent to compute near-optimal separating paths even if the upper and lower text parts are overlapping in particular places. We have performed experiments on the Saint Gall and Monk line segmentation (MLS) datasets. The experimental results show that our proposed method performs very well on the Saint Gall dataset, and also demonstrate that our algorithm is able to cope well with the much more complicated MLS dataset.**

***Keywords*—Document analysis, $A^*$ path-planning algorithm, Handwritten historical manuscripts, Line segmentation.**

## I. Introduction

Current search engines are very useful for people to search for information on the Internet. However, there is still a lot of information available on the Internet that cannot be used efficiently, because this information is contained in scanned document images that cannot be read or understood in an effective way by current search engine technology. We are especially interested in making handwritten documents accessible to people by recognizing the contents of these documents and making them searchable with a new generation of search engines tailored to handwritten documents.

The *MONK* system is a historical manuscript recognition system, consisting of many techniques for searching for words in historical manuscript collections. The system consists of different handwriting recognition algorithms, which are trained by crowd sourcing techniques where volunteers can create ground-truth labels for words and lines that occur in the historical documents [1], [2].

In this paper we describe a novel line segmentation algorithm for handwritten documents. Line segmentation [3] is one of the first techniques that needs to be applied to a document, before individual words or characters can be found and (parts of) the handwritten text can be automatically recognized. It should be noted that line segmentation is an intrinsically ill-posed problem that can only be solved using an interaction between classifiers and separation modules. Therefore, there is currently no method that can optimally deal with all difficulties such as curved lines and touching text lines, see Fig. 1, Fig 3(a), and Fig. 7(d) for some complicated examples.

**Related work.** The first step that is performed by a line segmentation algorithm is to find potential candidates for starting points of lines separating upper and lower text fields. Most often this step uses horizontal projection profiles, in which the amount of black ink is summed over the x-axis to obtain a profile indicating text areas having a lot or little to no black ink. Bulacu *et al.* [2] proposed the smoothed horizontal projection profile to more robustly detect peaks and valleys in the binarized document image. In [4], the handwritten document is divided into chunks, and the smoothed projection profile in each chunk is calculated. Then, the valleys in the projection profile are considered as the starting state of the text lines. Also in [5], the baselines of the valleys are used to define the starting states for the separating lines.

After defining the starting states, various methods for finding the line separating upper and lower text ares have been proposed, we refer to [3] for a complete survey on this area. In [2], a droplet method that preserves the ink connectivity is developed. Beginning in the starting state, first an initial straight path is generated. Then, the document image is turned 90 degrees and an artificial water droplet is moved from the top to the bottom of the page. This droplet tries to move around the ink along the straight path with the aim to preserve the ascenders and descenders in the final segmentation. The experimental results on a part of a dataset named "the cabinet of the Dutch queen" containing 32,816 lines (31.6 per page) showed that 99.8% of the lines were correctly segmented.

Saabni and El-Sana [6] proposed the use of the seam carving method to perform language-independent text line extraction. This method finds the extreme points that indicate the layout of text lines by first generating an energy map using the signed distance transform. The minimum energy paths pass along the middle of the text lines. The region of the text line is estimated from a set of intersecting components. Finally, the components between two consecutive lines are extracted and associated to the closest text line.

Garz *et al.* [7] proposed a binarization-free text line segmentation algorithm. First, parts-of-character interest points

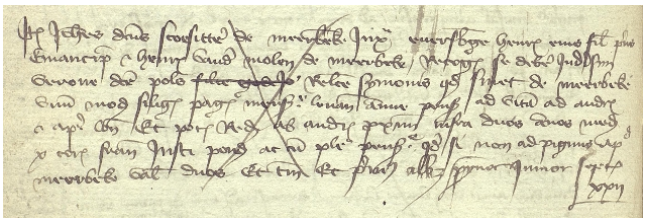


Fig. 1. A complicated historical manuscript example from the *MLS* dataset.

2167-6445/14 $31.00 © 2014 IEEE
DOI 10.1109/ICFHR.2014.37

are located by means of the Difference of Gaussians (DoG) filter after which locations of local minima and maxima are found in the gray-scale image. These detected interest points represent the most significant locations of portions of the text. Then, an energy map is computed around the located text points, and the seam carving technique is used to find a connected path with minimum cost that goes through low energy parts. This technique provides a hit rate of 0.9865 on 1,431 text lines of the *Saint Gall*[1] dataset.

Louloudis *et al.* [8] proposed the use of the Hough transform to perform line segmentation. In this method, the average width and height of connected components in the whole document are computed and used for partitioning the text into sub-areas. The sub-areas are again partitioned into equally sized blocks. After that the ink gravity enter is computed in each block. Finally, the set of all gravity center points is processed by the linear Hough transform to find a straight separating line. This technique provides a line detection rate of 97.4% on the ICDAR 2007 handwriting segmentation contest dataset.

Although these previous methods perform well when the text is structured, they still suffer from inaccurate line segmentations in case characters of subsequent lines are overlapping and even touching. Therefore the aim of this research is to develop a robust method to deal with this overlapping or touching text-lines problem and to obtain accurate line segmentations for different kinds of manuscripts.

**Contributions.** This paper proposes a line segmentation method based on the $A^*$ path-planning algorithm [9]. This well-known path-planning algorithm is combined with a number of cost functions to determine the optimal path separating upper and lower text areas. The cost functions have been designed in order to allow the separating path to go through text areas, although the path incurs a cost if it cuts ink pixels or gets close to them. This makes the proposed method very useful in case upper and lower text fields are overlapping or connected. The $A^*$ path-planning algorithm has been widely used in the field of artificial intelligence, however, this paper shows that this technique can also be very beneficial for line segmentation purposes. We have performed experiments on the *Saint Gall* dataset, and on several historical handwritten documents from the *MONK* line segmentation (*MLS*) dataset[2], which is a selection from the *MONK* collection.

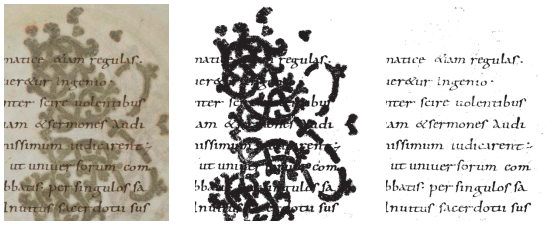
**Paper Outline.** This paper is organized as follows. In Section II, the method for detecting starting states of separating lines is explained and the datasets with handwritten historical manuscripts, which are used in our experiments, are described. Section III describes the $A^*$ path-planning algorithm for text line segmentation. A combination of novel cost functions is described that is used by the $A^*$ path-planning method to find near-optimal separating paths. The experimental results are described in Section IV. Finally, Section V discusses the findings of this paper and describes future work.

[1]The *Saint Gall* dataset is available at http://www.iam.unibe.ch/fki/databases

[2]The *MONK* line segmentation (*MLS*) dataset is available at http://www.ai.rug.nl/∼ mrolarik/MLS/

## II. Text Line Localization

An important aspect of line segmentation is to automatically detect the locations of text lines. In our approach, text line localization is performed in two steps: binarization and projection profile analysis. In the first step, the handwritten document images are binarized using a binarization technique. Such a technique takes into account the diversity of document images, texts, images, mixtures of texts and images, line drawings, and noisy or degraded document images. Bulacu *et al.* [2] and Surinta *et al.* [10] use Otsu's algorithm, a global binarization technique, in their work. Otsu's algorithm uses one threshold value to process an entire document image. This algorithm is not performing well when the background of the document image is complicated as shown in Fig. 2(b). On the other hand, Sauvola's algorithm [11] for local binarization copes effectively with complex backgrounds as shown in Fig. 2(c). Because the contrast between handwriting and background is low (see Fig. 2(a)), the threshold value is calculated by the mean and standard deviation of the local neighborhood of the gray pixel values. This threshold value has to be calculated for each pixel [12], [13].



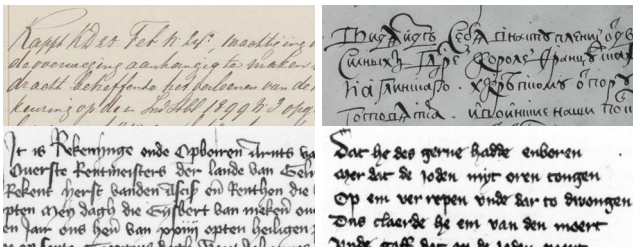
(a) (b) (c)

Fig. 2. Results of the document image after using a binarization technique. (a) The original handwritten document image, (b) background noise is removed by Otsu's algorithm, and (c) the result of Sauvola's algorithm.

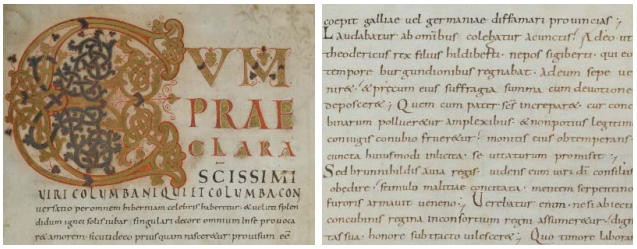
The experimental evaluation of the proposed method is done on two handwritten historical manuscript datasets, namely the *MLS* and the *Saint Gall* datasets (Fig. 3). The *MLS* dataset contains medieval, historical and contemporary manuscripts, and has the purpose of testing line-segmentation algorithms. The collection contains a wide variation of the common problems in handwriting recognition: lines with overlapping ascenders/descenders, slightly rotated scans and curved base lines. The *Saint Gall* dataset is written in the $9^{th}$ century, uses Latin script and contains 60 pages. Each page is written in one column and some pages contain a graphic [7], [14].

In this research, we have used Sauvola's algorithm with a window size of 20 × 20 pixels [14] to convert the typically 300 dpi gray document image to a binary document image, the result of which is shown in Fig. 2(c). The structure of the characters is legible [15] when zooming into the pixel level, see Fig. 4. Unfortunately, Sauvola's algorithm was not able to delete all non-text graphics. For this reason, in case it failed, we manually removed the graphics. We will research automatic removal of all graphics and other issues related to layout analysis in future work.

The second step uses the concept of projection profile analysis [16] for finding the location of the text lines in the handwritten document image. The horizontal ink density histogram of the document image is computed by taking the



(a)



(b)

Fig. 3. A variety of handwritten historical manuscript samples. (a) Four samples from the *MLS* dataset. (b) Two samples from the *Saint Gall* dataset, which is one dataset in the *IAM* historical document database (*IAM-HistDB*). Please note the problem of vertical overlap, especially in Figure (a) up-right.

Fig. 4. Result of Sauvola's algorithm when zooming to the pixel level.

sum of the black pixel values in the corresponding row, and then storing the values into a vector. Subsequently, the maxima are extracted from the vector. In our case, we consider the starting points for line segmentation to fall in between the local maxima of the ink density histogram. For dealing with noise, we make use of a *persistance* threshold, which allows to avoid false local maxima. Therefore, we only consider local maxima if the ink density histogram value at some local maximum is larger than $(\mu_h - \sigma_h)$, where $\mu_h$ is the average of the ink density histogram and $\sigma_h$ is the standard deviation. Then these located local maxima represent the text lines, and the starting and end points for the subsequent line segmentation stage are set in the middle between two subsequent local maxima.

## III. The $A^*$ Path-Planning Algorithm for Text Line Segmentation

Path planning has been applied to different applications in artificial intelligence, such as in robotic systems, route planners, and games. The aim of path planning is to compute the shortest path that allows the agent to reach its destination given its current position. Path-planning methods compute a path for a given environmental representation in the form of a map [17]. The environmental map can contain many obstacles, which the agent is not allowed to pass through.

The goal of the $A^*$ path-planning algorithm is to minimize the sum of costs on the path between the starting state $s_1$ and the goal-state $s_n$. If we denote $s_1^a, s_2^a, \dots s_n^a$ as the sequence of states traversed by path $p^a$, then the goal is to compute the optimal path $p^*$ with the lowest total traveling cost:

$$p^* = \arg\min_{p^a} \sum_{i=1}^{n^a-1} C(s_i^a, s_{i+1}^a) \tag{1}$$

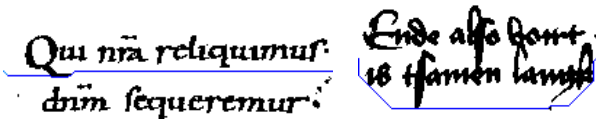
where $C(s_i, s_j)$ is the cost to go from state $s_i$ to state $s_j$.

The $A^*$ path-planning algorithm uses some heuristic function to speed up computing the optimal solution to reach the goal state. The algorithm combines the cost of the current path between the starting state and the current state with an admissible heuristic function that estimates the shortest possible cost from the current state to the goal state. The heuristic function uses the Euclidean distance to compute a lower bound on the expected cost to travel to the goal state from the current state.

### A. The Problem with Unreachable Goal States

The standard $A^*$ path-planning algorithm cannot address the problem of an unreachable goal state. For example, it cannot find a goal state when it is enclosed from all sides by obstacles. This is because in the standard algorithm the agent is not allowed to move through obstacles and then in this case it can never compute a path to reach the goal state.

We are interested in line segmentation of handwritten historical manuscripts. In our problem, black ink pixels are converted to obstacles and the start and goal states are determined based on the text line localization method explained in Section II. When using the original A* path-planning algorithm for this problem, it is effective when the components of two lines do not overlap. On the other hand, the output of the algorithm is incorrect when the handwritten text of two subsequent lines is overlapping. Some results of the standard $A^*$ path-planning algorithm are shown in Fig. 5(a) and Fig. 5(b). The result in Fig. 5(a) is good and shows that for easy cases the standard algorithm can separate the lines well with an almost straight line. The result in Fig. 5(b) clearly shows the problem of the standard A* algorithm when the text parts of two lines are overlapping.



(a) (b)

Fig. 5. Illustrations of the standard $A^*$ path-planning algorithm. (a) The agent correctly separates two character lines. (b) The agent cannot divide the two touching text lines, because it cannot move through obstacles.

Our proposed $A^*$ path-planning algorithm operates quite differently from the standard method. Most importantly, our $A^*$ algorithm allows the agent to pass through obstacles. However, going through them incurs some cost and therefore it is better for the agent not to traverse these obstacles, if possible. Consequently, the problem of the touching text lines is solved at the algorithm level.

### B. Cost Functions in our $A^*$ Path-Planning Algorithm

Our $A^*$ path-planning algorithm uses five cost functions. These cost functions are combined to compute the traveling cost from a state until the goal state is reached. We will now explain the used cost functions in our path-planning algorithm.

*1) The Ink Distance Cost Functions $D(n)$ and $D(n)^2$:* These two cost functions control the agent to stay more or less in between ink pixels above and below. They use the closest distance $d$ of the agent to possible obstacles in the upward and downward directions. The ink distance cost function $D(n)$ for traveling through state $n$ is:

$$D(n) = \frac{1}{1 + min\left(d\left(n, n_{y_u}\right), d\left(n, n_{y_d}\right)\right)} \tag{2}$$

where $d(n, n_{y_u})$ and $d(n, n_{y_d})$ are the distances between the state $n$ and the closest obstacle (ink pixel) in the upward and downward direction, respectively. The distance is set to a maximum value if no obstacle is found in that direction. Note that the cost is highest (with a value of 1) if the agent passes an obstacle. Figure 6 illustrates this cost function. We also make use of $D(n)^2$ which attributes a much higher cost to getting close to pixel values compared to staying further away from the black pixels. $D(n)^2$ is defined as follows:

$$D(n)^2 = \frac{1}{1 + min\left(d\left(n, n_{y_u}\right), d\left(n, n_{y_d}\right)\right)^2} \tag{3}$$

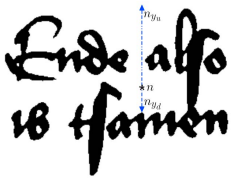



Fig. 6. Illustration of the ink distance cost function. The minimum cost is computed from the distance between state $n$ and the closest obstacles in the upward direction $y_u$ and the downward direction $y_d$.

*2) The Map-Obstacle Cost Function $M(n)$:* To enforce the agent from trying not to traverse obstacles in the map so that it does not cut through black ink, we created another cost function that gives a penalty only if the agent passes through an obstacle. In this cost function $M(n)$ returns 1, if the state $n$ lies on a black pixel, and otherwise $M(n)$ returns 0.

*3) The Vertical Cost Function $V(n)$:* The vertical cost function $V(n)$ is used so that the path does not deviate too much from the $y$-position of the starting and end position. This prevents the agent from going up or down an entire line. The cost function $V(n)$ is defined as:

$$V(n) = abs(n_y - n_y^{start})$$

where $n_y$ is the $y$-position of the current state and $n_y^{start}$ is the $y$-position of the starting point.

*4) The Neighbor Cost Function $N(s_i, s_j)$:* The neighbor cost $N(s_i, s_j)$ is used to compute the shortest path between the start and goal state, and is the same as also used in the standard A* path-planning algorithm. When making a move to a new state, the cost $N(s_i, s_j)$ is 10 for a vertical and horizontal step and the cost is 14 for diagonal directions. In each state the agent can make use of 8-directional movements.

The proposed $A^*$ path-planning algorithm now uses the following combined cost-function $C(s_i, s_j)$:

$$\begin{aligned} C(s_i, s_j) \quad &= \quad c_d D(s_i) + c_{d2} D(s_i)^2 + c_m M(s_i) \\ &\quad + c_v V(s_i) + c_n N(s_i, s_j) \end{aligned} \tag{4}$$

The parameters $c_d$, $c_{d2}$, $c_m$, $c_v$ and $c_n$ have been tuned using some images during preliminary experiments. This cost function is used to compute the path cost. After each move the state with the lowest value is used to expand its current path. Once the goal-state is to be expanded, the optimal path based on the used cost functions has been found.

## IV. Experimental Evaluation

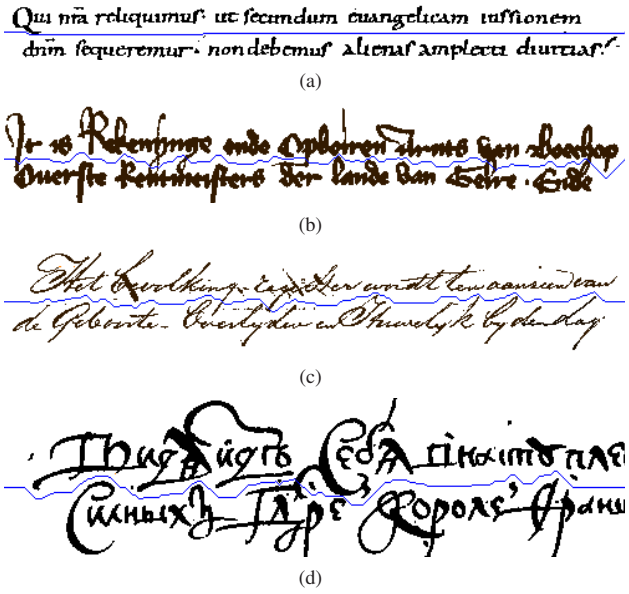
The whole algorithm now works as follows. In the first step the handwritten historical manuscripts are binarized using Sauvola's algorithm. In the second step the smoothed horizontal ink density histogram of the binary image is calculated. Then peaks of the horizontal ink density histogram are detected by the local maxima method. The starting state of each line is set between subsequent peaks. Finally, the $A^*$ path-planning algorithm is applied.

Our line segmentation system has been applied to handwritten historical manuscripts from the *MLS* and the *Saint Gall* dataset. The values we used for the experiments on the *Saint Gall* dataset are: $c_d = 150$, $c_{d2} = 50$, $c_m = 50$, $c_v = 3$, and $c_n = 1$. The values we used for the experiments on the heterogeneous *MLS* dataset are: $c_d = 130$, $c_{d2} = 0$, $c_m = 50$, $c_v = 2.5$, and $c_n = 1$.

Some results of the $A^*$ path-planning algorithm are shown in Fig. 7. The text lines are separated by the optimal path (i.e. the path with the lowest cost). Some line segmentation results on whole document parts of our method are shown in Fig. 8.

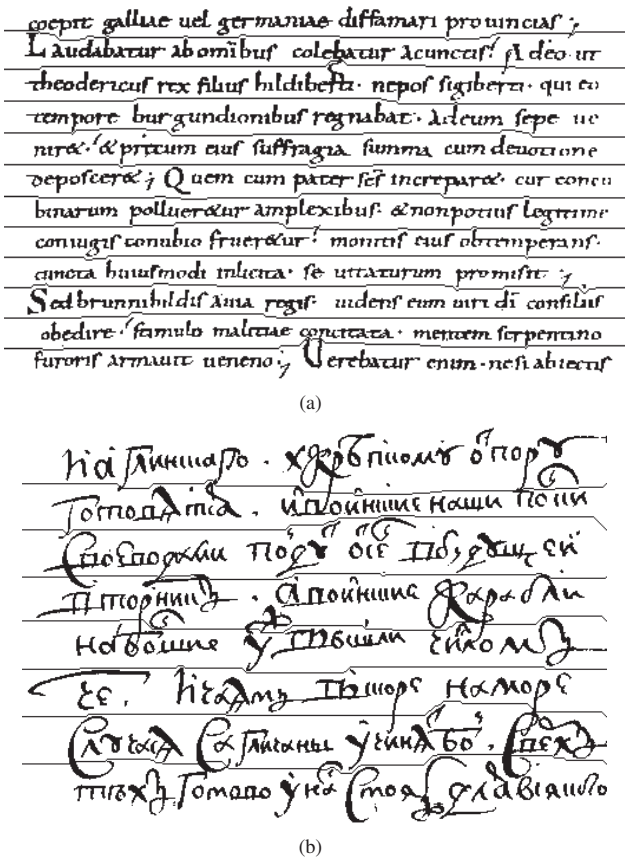
The ground-truth line segmentation for these datasets is acquired manually with the help of a tool developed for this task. This tool presents a scanned document in a web page, and allows the human user to mark handwritten text with the mouse pointer by selecting a vertical area (denoted by two y-values in the image) using JavaScript (see Fig. 9). After annotating these text lines, the page image is split as follows: for each line annotation the area above and below the line area are whitened in a copy of the original image. This preserves the original image size. Descenders and ascenders from respectively the lines above and below are not removed by this process; they are removed manually by whitening them with a simple paint program (e.g., xpaint). This procedure yields the ground truth for the target line.

Both the input images and output images are losslessly compressed to prevent any influence from lossy compression artefacts. Each line annotation image is named as the original



(a) (b) (c) (d)

Fig. 7. Some results of our $A^*$ path-planning algorithm.



(a) (b)

Fig. 8. Line segmentation results on handwritten documents from (a) the *Saint Gall* dataset and (b) the *MLS* dataset.

image, appended with a line number. This allows the matching of the ground-truth images with the output images of our $A^*$ path-planning algorithm.

For evaluating the performance of the line segmentation algorithm, we will use the pixel-level hit rate and the line detection accuracy measure on the binary document image, as proposed by Li *et al.* [18] and Garz *et al.* [7].

After the line segmentation algorithm is finished, we will have $M$ ground-truth lines and $N$ detected lines. Now a matrix $P$ of size $M \times N$ is computed, where $P_{ij}$ is the number of shared black pixels between the $i^{th}$ ground-truth line and the $j^{th}$ detected line [7], [19]. The goodness of assigning particular ground-truth lines to particular detected lines is given by the total number of shared black pixels given this assignment. From all the possible assignments the optimal one is selected, resulting in the value $G(S_{max})$ indicating the total number of shared pixels for the optimal assignment $S_{max}$. The pixel-level hit rate is now defined as follows:

$$Hr = \frac{G(S_{max})}{|GT \cup R|} \tag{5}$$

where $GT$ is the set of black pixels in the ground-truth line, and $R$ is the total number of black pixels found by our algorithm including pixels which are not in the ground-truth.

We also use the text-line detection accuracy measure [18], [19]. A line $i$ is correctly detected if $\frac{G_{ij}(S_{max})}{|GT_i|} \geq 0.9$ and $\frac{G_{ij}(S_{max})}{|R_j|} \geq 0.9$. where $GT_i$ is the set of black pixels in the ground-truth line $i$, $R_j$ is the set of black pixels in the matching $j$-th line found by our algorithm and $G_{ij}$ is the number of shared black pixels between line $i$ of the ground-truth and the $j$-th detected line. This measure is affected by missing parts of the text-line and additional noise in the detected line.

We computed results on the *Saint Gall* dataset using a manuscript containing a total of 1,429 lines and on different manuscripts from the *MLS* dataset containing in total 995 lines. For the *Saint Gall* dataset, our method obtains a near-perfect pixel-level hit rate of 0.998 and a line-detection accuracy of 0.999. The computation time for the 1,429 lines is around 8 minutes. On the much more complicated *MLS* dataset our method obtains a pixel-level hit rate of 0.928 and a line-detection accuracy of 0.9. The computation time for this dataset for the 995 lines is around 26 minutes. Most errors on the *MLS* dataset are caused by our method for detecting starting points of lines, because this dataset contains some very short lines, with which our line detection algorithm cannot cope well. Other errors on the *MLS* dataset are caused by lines which are not horizontal, but slanted.

Table I presents the hit rate and the line accuracy for the *Saint Gall* dataset for several methods. In this table the performance of our method is compared with two systems including the method by Garz *et al.* [7] and the method by Baechler *et al.* [19]. The results show that our method outperforms previous methods on this dataset.

TABLE I. LINE ACCURACY AND HIT RATE OF LINE SEGMENTATION ON THE SAINT GALL DATASET

| | Hit rate | Line Accuracy |
|---|---|---|
| Baechler *et al.* | 0.9600 | 0.9540 |
| Garz *et al.* | 0.9865 | 0.9797 |
| Our method | 0.9980 | 0.9990 |

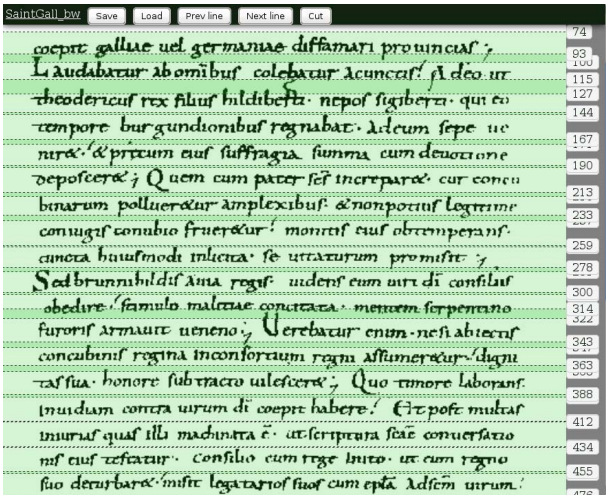



Fig. 9. Illustration of the ground-truth tool. The text line area is marked by the human user.

## V. Conclusion

In this paper an $A^*$ path-planning algorithm is described that uses a combination of cost functions to control a line segmentation agent. The algorithm uses a binarized image obtained from a handwritten text image using Sauvola's algorithm. The start and end points of a line are detected using the smoothed horizontal ink density histogram. The algorithm uses different cost functions to stay as far away as possible from ink pixels and at the same time tries to compute the shortest path. The results on two historical line-segmentation datasets show that our $A^*$ path-planning algorithm successfully separates subsequent text lines, even when they partially overlap. The advantages of the proposed method are that our method is fairly simple to implement, quite fast, and robust for different kinds of handwritten documents. The disadvantage is that sometimes the method prefers to cut some text instead of going up with a curved character. This is a consequence of the trade-off between staying away from ink pixels and computing the shortest path. As stated in the introduction a synergy between bottom-up and top-down processing using text classification likelihoods can solve these problems in the future. Also Sauvola's algorithm needs to be applied iteratively using several window sizes in a real system.

In future work, layout analysis [2], [19] will be used to handle the document image before applying the line segmentation technique. We will use an object detection technique to detect non-text graphics in the handwritten images. Furthermore, we plan to use an energy function [7] that can be computed on gray images instead of the black/white images we used in this paper. Given the current system, it would be interesting to see if the parameters can be automatically optimized using an adaptive framework. Finally, we will apply the $A^*$ path-planning algorithm to a handwritten Thai dataset, which we recently collected, and combine this with handwritten character recognition algorithms.

## References

[1] M. Bulacu, A. Brink, T. van der Zant, and L. Schomaker, "Recognition of handwritten numerical fields in a large single-writer historical collection," in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, 2009, pp. 808–812.

[2] M. Bulacu, R. van Koert, L. Schomaker, and T. van der Zant, "Layout analysis of handwritten historical documents for searching the archive of the cabinet of the Dutch queen," in *Document Analysis and Recognition, 2007. ICDAR '07. 9th International Conference on*, vol. 1, 2007, pp. 357–361.

[3] L. Likforman-Sulem, A. Zahour, and B. Taconet, "Text line segmentation of historical documents: a survey," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 9, no. 2-4, pp. 123–138, 2007.

[4] M. Arivazhagan, H. Srinivasan, and S. Srihari, "A statistical approach to line segmentation in handwritten documents," in *Document Recognition and Retrieval XIV, 2007. DDR '07. 14th Conference on*, vol. 6500, 2007, pp. 65 000T–65 000T–11.

[5] R. Chamchong and C. C. Fung, "Text line extraction using adaptive partial projection for palm leaf manuscripts from Thailand," in *Frontiers in Handwriting Recognition, 2012. ICFHR '12, 13th International Conference on*, 2012, pp. 588–593.

[6] R. Saabni and J. El-Sana, "Language-independent text lines extraction using seam carving," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, Sept 2011, pp. 563–568.

[7] A. Garz, A. Fischer, R. Sablatnig, and H. Bunke, "Binarization-free text line segmentation for historical documents based on interest point clustering," in *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, 2012, pp. 95–99.

[8] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis, "Text line and word segmentation of handwritten documents," *Pattern Recognition*, vol. 42, no. 12, pp. 3169–3183, 2009.

[9] N. Nilsson, *Principles of Artificial Intelligence*, ser. Symbolic Computation / Artificial Intelligence. Springer, 1982.

[10] O. Surinta, L. Schomaker, and M. Wiering, "A comparison of feature extraction and pixel-based methods for recognizing handwritten Bangla digits," in *Document Analysis and Recognition, 2013. ICDAR '13. 12th International Conference on*, 2013, pp. 165–169.

[11] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[12] T. R. Singh, S. Roy, O. I. Singh, T. Sinam, and K. M. Singh, "A new local adaptive thresholding technique in binarization," *International Journal of Computer Science Issues*, vol. 8, pp. 271–277, 2011.

[13] F. Shafait, D. Keysers, and T. M. Breuel, "Efficient implementation of local adaptive thresholding techniques using integral images," in *Document Recognition and Retrieval XV, 2008. DDR '08. 15th Conference on*, 2008, pp. 681 510–681 510–6.

[14] A. Fischer, V. Frinken, A. Fornés, and H. Bunke, "Transcription alignment of Latin manuscripts using hidden Markov models," in *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing. HIP '11*, 2011, pp. 29–36.

[15] E. Badekas and N. Papamarkos, "Optimal combination of document binarization techniques using a self-organizing map neural network," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 1, pp. 11–24, 2007.

[16] R. Ghosh, D. Bhattacharyya, T.-h. Kim, and G.-s. Lee, "New algorithm for skewing detection of handwritten Bangla words," in *Signal Processing, Image Processing and Pattern Recognition*, ser. Communications in Computer and Information Science, T.-h. Kim, H. Adeli, C. Ramos, and B.-H. Kang, Eds. Springer Berlin Heidelberg, 2011, vol. 260, pp. 153–159.

[17] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Robotics and Automation, 1994. ICRA 1994. IEEE International Conference on*, 1994, pp. 3310–3317 vol.4.

[18] Y. Li, Y. Zheng, D. Doermann, S. Jaeger, and Y. Li, "Script-independent text line segmentation in freestyle handwritten documents," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 8, pp. 1313–1329, 2008.

[19] M. Baechler, M. Liwicki, and R. Ingold, "Text line extraction using DMLP classifiers for historical manuscripts," in *Document Analysis and Recognition, 2013. ICDAR '13. 12th International Conference on*, 2013, pp. 1029–1033.