Proceedings of the 2017 IEEE
International Conference on Information and Automation (ICIA)
Macau SAR, China, July 2017

Text and Character Recognition on Metal-sheets

Jan Kronenberger
Institut Informatik
Hochschule Ruhr West
Bottrop, Germany
jan.kronenberger@stud.hs-ruhrwest.de

Darius Malysiak
Institut Informatik
Hochschule Ruhr West
Bottrop, Germany
darius.malysiak@hs-ruhrwest.de

Uwe Handmann
Institut Informatik
Hochschule Ruhr West
Bottrop, Germany
uwe.handmann@hs-ruhrwest.de

***Abstract*— To improve the automation of metal sheet production these sheets have to be tracked during the processing steps. This is preferably done by video optical tracking. To recognize the metal sheets each of them has an unique ID imprinted. This paper describes a method to automatically detect the characters on metal sheets, to classify them and to combine them to a string. First the image is preprocessed to detect the metal sheets in the image. After this step every character is located and classified. Finally all characters are combined to one string with its specific certainty. The algorithm has to determine about the alignment and the orientation of the text because it is not known before. The most problematic part is the extraction of the objects, which may be characters, due to difficult lightning situation. There can be different lightning spots visible in the image because of the movement of the sun, weather and factory infrastructure (windows, ceiling lighting) which cause sometimes difficult reflections on the metal sheets.**
The introduced program is completely written in Matlab 2015b. It has been applied to a wide range of images with successful results.

***Index Terms*— Pattern recognition, classification, Matlab**

I. Introduction

In order to read text from an image it is necessary to do some situation regarding preprocessing. Figure 1 shows an example image as it is captured from the cooling bed of a german company producing metal sheets. Each image is greyscale, has an resolution of 1032×1592 pixels and a depth of 12bit.

Because the character printing system uses only one font with fix defined size and space between characters, some assumptions can be made to reduce the complexity of the algorithm. The first assumption can be made regarding the characters due to the fact that each character will not change during time and there will be no new characters added. There will be only small differences due to lightning influences. Another assumption is the limitation of only one line per sheet, which has to be detected. This line is always parallel to the long sheet edge. Sometimes there even is no text printed on the metal-sheets or the text is outside of camera capture area. As there is no tracking implemented between two following images each image is processed without any prior knowledge.
First some State of the Art Papers are presented in Section II. In Section III the sheets are detected using morphological operations. Section IV is about finding the text-area on the sheet. In Section V the chars are classified within the previous found text-area. These Characters are combined into one string, which is described in Section VI. Additionally a certainty for each string is given. Finally Section VII shows the performance of the proposed system.
The system is designed to work with all numbers and every character of the alphabet, but as the given data-set does not contain all possible characters, this will lead to some gaps in the evaluation (See Figure 13). Any difficulties which may appear when all characters would be used are not studied. Figure 2 shows the block diagram of the program. The whole process is done once per image without tracking or remembering. The block diagram shows the splitting part, where the text is processed once at $0°$ and once at $180°$. Except the rotation part both variants do exactly the same. After calculating the scores of both detected texts, the one with the higher score will be selected, validated and passed to the visualisation part.

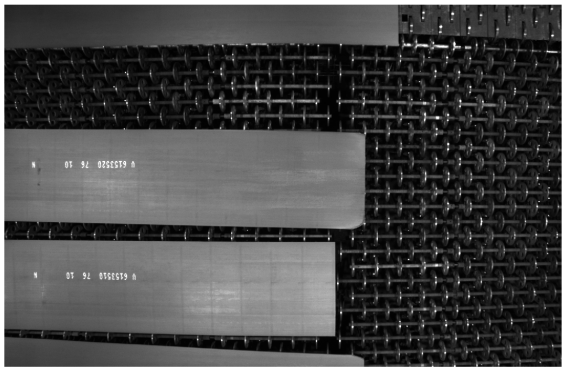

Fig. 1. Metal sheets on conveyor in cooling bed. The camera is mounted above it.

II. Related Work

The problem with pattern-matching in digitized images with Machine-printed characters was already tried to solve with different approaches. Some algorithms are proposed by

978-1-5386-3154-6/17/$31.00 ©2017 IEEE

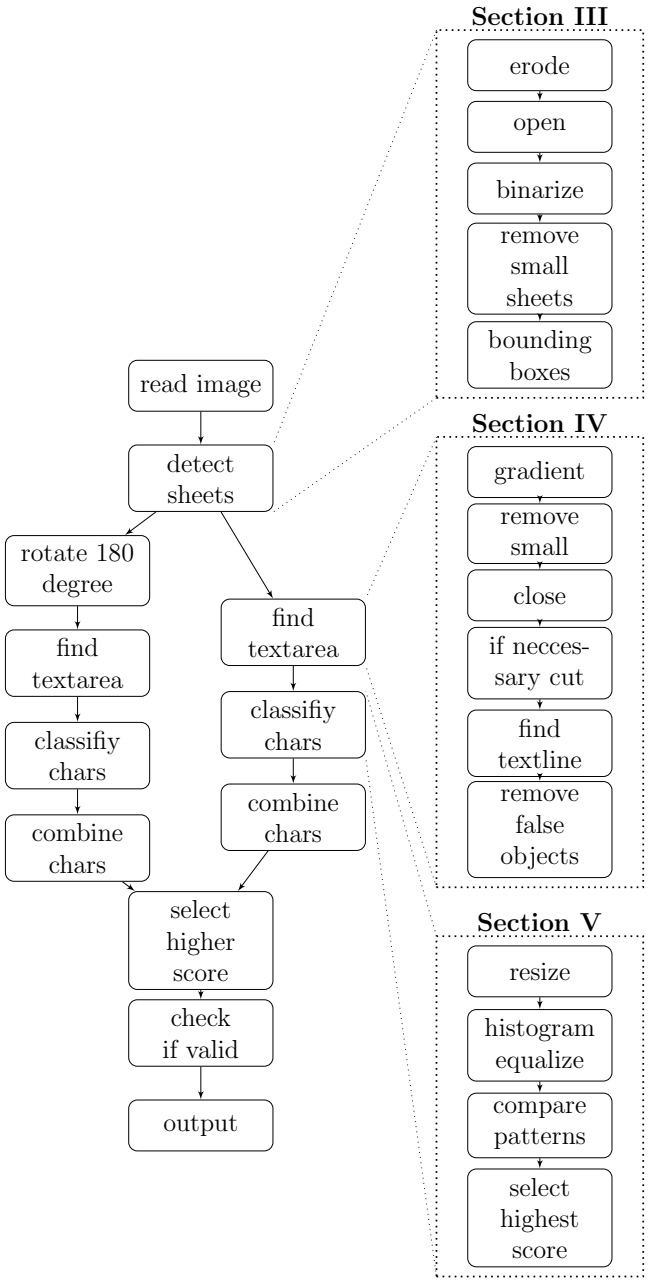

Fig. 2. Block diagram

Safronov et al. [1]. They described the advantages of the different techniques and their application area. To classify broken characters Babu et al. [2] used gradient patterns and their relationship. Hoffman and McCullough [3] divide the characters in vertical strokes or detect their contours to classify them. But this method requires a higher resolution of the characters which is not given here. Others like Li [4] require binary images. This works well for scanned documents with know fonts and with sharp edges on the text. The lightning situation in this project gives us quite unsharp images of the characters. Another advance technique is the graph matching shown by Rocha and Pavlidis [5]. Their performance is quite good, but it requires a well working graph extraction on the input data to get reliable results.
In this case the assumptions which are made for this solution allow to prevent such computationally expensive algorithms.

III. Sheet Detection

As seen in Figure 1 the image contains a lot of noise which should be removed before the sheets could be detected safely. Therefore two morphological operations are applied. First the image is eroded with an 25×25 pixel diamond pattern (See Figure 3) to shrink high value areas (Top left in Figure 4). This will lead to a more smooth image without high frequencies.

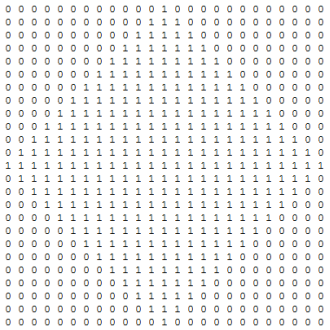

Fig. 3. Diamond Pattern for eroding

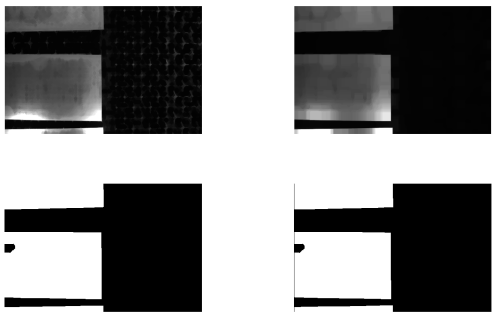

Fig. 4. Steps of the sheet-detection: 1. Erode, 2. Open, 3. Remove small objects, 4. Thresholded

In the next step the image is morphological opened [6] to clean the contours and to remove remaining spots which is shown in the top right sub-figure. On the bottom right the result is thresholded to get a binary image. In the final step all connected pixel are combined into objects. Objects having a smaller area as a given threshold are removed. This result is displayed at the bottom right.

At this point the detected objects are cut out for faster computing. They are rotated as well to make the edges of the sheet be parallel to the x- and y-axis of the image. This will remove the problem of only having invariant patterns in section V.
Because the orientation of the text is unknown, the image is processed two times. One time as it is cut out and the second time rotated at $180°$.

IV. Text-area Detection

To localize the chars an edge detector is applied and the image is binarized. Now all connected components are selected as one object. If one object contains less than 40 pixels it is removed. To close small gaps and prevent false grouping the image is morphologically closed with $mask = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$.
This will lead to a list of objects. To find the text-line, false objects have to be removed. Fitting an horizontal line through all objects gives the ability to remove objects with a large offset to this line. This is possible because the sheet was rotated before and we assume there is only one text-line. Figure IV shows an example of the text-line fitting.

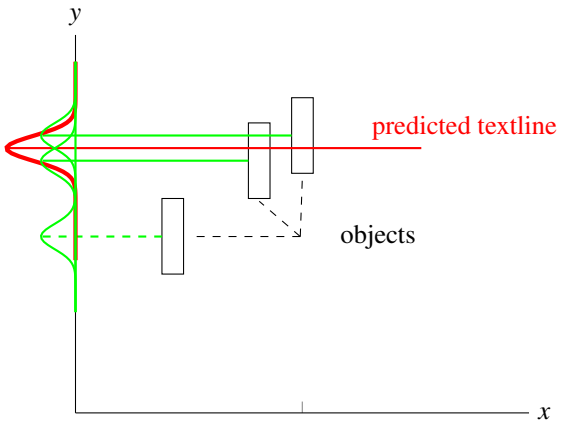

Fig. 5. Finding the text-line. As the bottom char object is too far away from the text-line, it will be removed

Therefore the center-point of every object is calculated and the distribution is plotted along the y-axis. The text-line is now determinated at the point with the highest distribution. The objects are now filtered regarding to their dimensions. If the height is less than 14 pixels and the width is less than 5 pixels, the object is removed. This filtering is possible due to the assumption that the font will not change and the distance of the camera to the sheets will always be the same (Appearance on the image will not change). But as the transition between two or more adjacent characters may not be clean, multiple characters could be packed into one object. Because we know the aspect ratio of the printed characters (≈ 2), the amount of characters in this object can be calculated by the dimensions of the object with

$$count = round(\frac{width}{height/2}) \tag{1}$$

The object is then divided every $\frac{width}{count}$ pixels and stored in $count$ new objects. Figure 6 shows the steps of the object detection. On the top left the sheet-image is displayed. The top right image shows the edge detection. In the next step the objects are extracted and finally in the fourth step the false objects are removed.

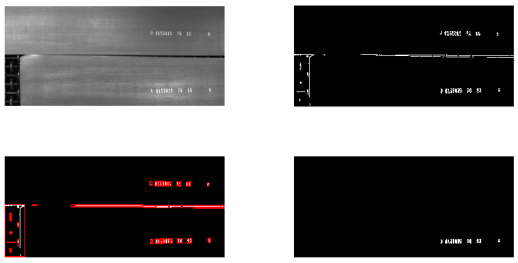

Fig. 6. Steps of textarea detection: 1. Croped sheet-image, 2: Result of the edge detection, 3. Possible objects are selected, 4. Removal of false objects

As this list of objects could still contain non-characters each object now has to be classified

V. Character Classification

At first the templates for the classification have to be generated. Therefore all labeled images of one character are resized to 20×10 pixels, summed up and normalized (Figure 7) into one pattern. To remove the background of one character image, the lowest pixel value of the image is subtracted at each pixel. As the background pixel value is very uniform, this simple subtraction removes nearly everything unnecessary.

To get the character for one detected object the metal-sheet first has to be processed in the same way as the pattern was while learning. To get the same result first the background is subtracted, then the image is normalized and finally it is re-sized to the size of the patterns. The formula

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}} \tag{2}$$

[7] calculates the difference between an image A and an image B, where $\bar{A}$ and $\bar{B}$ are the mean values of A and B. The result describes the similarity of the images between 1.0 (100%) and 0.0 (0%). Figure 8 shows the similarities between the input image and all patterns.

Fig. 7. Normalized averages of different patterns

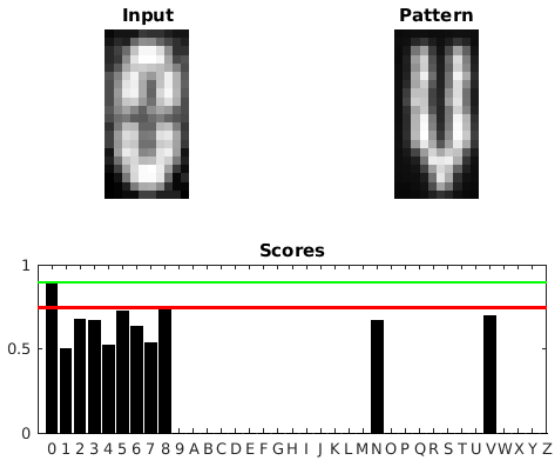

Fig. 8. Matching against all patterns.
green line: highest value
red line: second highest value

As greyscale patterns are used, there are many advantages to binary patters like they are used in [8]. While binary patterns would produce worse results in matching when they are not perfectly arranged to the input image, our greyscale patters will allow small invariances in scale, rotation and location due to its smoothness.

To measure the confidence of the classification two parts are considered.

1) Total score (*T1*)
2) Difference to second best score (*T2*)

A good value for the total score is *T1* = 0.6 and for the difference *T2* = 0.01. The difference value *T2* has to be that low because of similar pattern like 6 and 8. If it is too high these patterns can not be stably kept apart. For further explanation see Section VII-A.

VI. Textbuilding

At this point there are two object arrays. One for the sheet rotated at $0°$ and one at $180°$. Each array contains a list of characters with a score and a confidence per character.

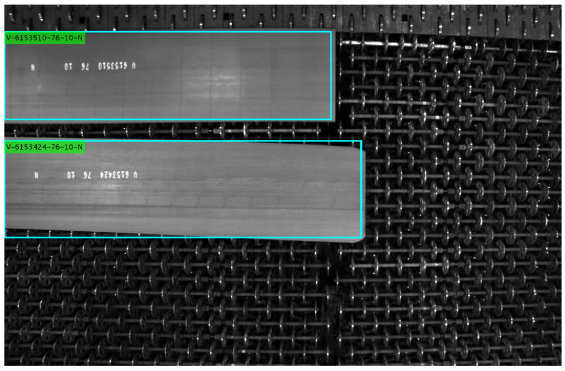

Fig. 9. Final output

To figure out which array should be taken the scores of each array are summed up and compared. The one with the higher score will be selected. This method works only if there are some rotation invariant characters in the array. Some characters which are not rotation invariant are 0, 8, *O* and *H*. If the x-Coordinate of the objects containing two adjacent characters is greater than a threshold, a space is inserted between them. Figure 9 shows an example output image, where the sheets are marked with a bounding-box and the text is displayed. Notice every space is written as "-". If any character in the selected array has a low score or confidence, the text-string is still build, but with a wild card on the characters position. Alternative the strings with the highest scores could be printed.

VII. Evaluation

The evaluation is divided into two parts. Part VII-A describes the certainty of the trained dataset. Part VII-B evaluates the trained dataset against a test-set of characters. As the data only contains following characters *012345678VN* there will be some gaps for the other characters in the evaluation (Figure 10 and 13). Remaining problems in this solution are discussed in part VII-C.

A. Patternconfidence

As mentioned in Section V some patterns are very equal which makes a certain detection difficult. To get a score for the trained patterns each pattern is compared to each other. The comparison is done like in section V [7]. Figure 10 shows the result of the comparison in the form of a confusion matrix.

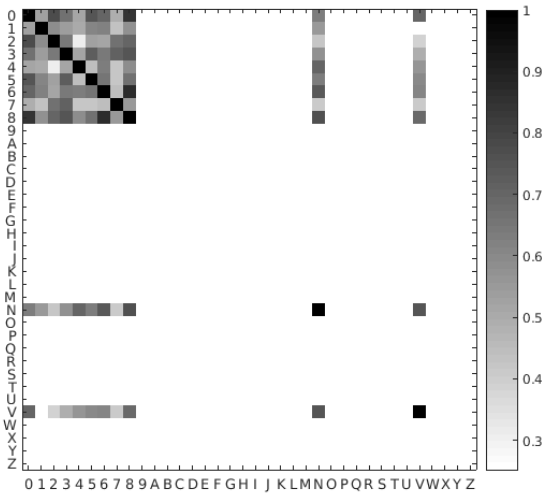

Fig. 10. Confidence between patterns. The higher the more equal

Most of the patterns have a similarity of about 40 to 60%. The resulting difference between the character 6 and 8 is only $13,63\%$ (Shown in Table 11). If there is less similarity between the patterns, the character classification will be more certain, but it might happen that a character is not classified because the difference to the pattern is too high (overfitting). This happens when there is too much training. To check if there is overfitting the trained data is tested against a set of images which were not used in the training in section VII-B.

Pattern 1	Pattern 2	Score
6	8	0.8637
0	8	0.8366
...	...	...
2	4	0.3015
1	V	0.2520

Fig. 11. Lowest and highest similarity between patterns. 1 = equal

B. Character classification

To evaluate the char-detection a test-set with 513 labeled chars is created which is about 40 - 50 images per char. This set does not contain any char on which the training-set was trained on.

Training	513
Test	123
Total	636

Fig. 12. Images in Testset

To measure the performance a confusion matrix (Figure 13) is created. It shows how many images of e.g. class 7 were correctly classified. Any false classification is marked outside the diagonal. If any character could not be classified with a high score or good confidence it is categorized as *?*. The perfect confusion matrix would be an identity matrix, where 100% of each class is classified correct.

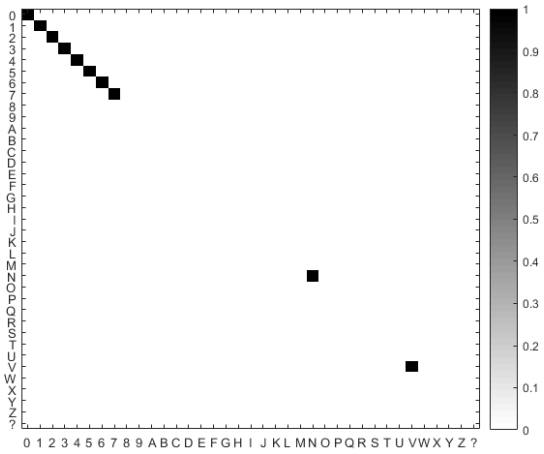

Fig. 13. Confusion matrix of all trained Characters

Character	Total Images	Right	False	Unsure
0	74	71	0	3
1	7	7	0	0
2	16	16	0	0
3	2	2	0	0
4	10	10	0	0
5	4	4	0	0
6	2	2	0	0
7	2	2	0	0
8	2	2	0	0
V	2	2	0	0
N	2	2	0	0

Fig. 14. Classification Results

The confusion matrix shows that the characters are classified correct at a rate of $97,56\%$. 3 images of the character *0* could not be classified with confidence and where categorized as *?*. As suspected in section VII-A the classification for the characters 0 and 8 is very difficult and may be wrong if there are any misleading reflections.

C. Remaining problems

The performance of the system depends on many steps. If the first steps fail or produce bad results the final result will be bad as well. The first big problem is the sheet detection. If a sheet is not or false detected, there would be no character detection. This mostly occurs when there are multiple sheets very close or on top of each other.

The second problem concerns the detection of the characters. Due to the lightning many false-positives are selected. Most of them are deselected when the textarea is found, but they can disturb the textarea detection.

VIII. Conclusion

In this article a fast text recognition system is introduced. The comparison on such small templates is very fast because of the few pixels which have to be compared and because there is no need to extract features like lines or shapes. The process could be speeded up, if there would be some tracking implemented. Right now, the orientation of the sheets has to be calculated every time again. This can be skipped if the orientation would be know from the prior image. The accuracy would increase, too.

Overall this method works well if there are no light spots which outshine parts of the image. Especially when parts of the text are outshine the text-detection will not work. Additionally the pattern matching works only if the font of the chars do not change. This would break the assumptions. This system shows an overall performance of $97,56\%$ for the character classification.

References

[1] K. Safronov, I. Tchouchenkov, and H. Wörn, "Optical character recognition using optimisation algorithms," University of Karlsruhe, Tech. Rep., 2007.

[2] R. Babu and M. Ravishankar, "Recognition of machine printed broken characters based on gradient patterns and its spatial relationship," Dayananda Sagar College of Engineering, Tech. Rep., 2010.

[3] R. L. Hoffman and J. W. McCullough, "Segmentation methods for recognition of machine-printed characters," IBM General Systems, Tech. Rep., 1970.

[4] N. Li, "An implementation of ocr system based on skeleton matching," University of Kent at Canterbury, Tech. Rep., 1991.

[5] J. Rocha and T. Pavlidis, "A shape analysis model with applications to a character recognition system," IEEE, Tech. Rep., 1994.

[6] MathWorks. Gradient magnitude and direction of an image. [Online]. Available: https://de.mathworks.com/help/images/ref/imgradient.html

[7] ——. 2-d correlation coefficient. [Online]. Available: https://de.mathworks.com/help/images/ref/corr2.html

[8] F. Mohammad, J. Anarase, M. Shingote, and P. Ghanwat, "Optical character recognition implementation using pattern matching," ISB M School of Technology, Tech. Rep., 2014.