British Journal of Mathematics & Computer Science
4(7): 933-946, 2014

SCIENCEDOMAIN *international*
www.sciencedomain.org

Reading Skewed Images without Image Rotation

Wassim Al-Khawand[1*], Seifedine Kadry[2], Riccardo Bozzo[3] and Khaled Smaili[4]

[1]School of Engineering Sciences and Technologies, University of Genoa–UNIGE, Genoa, Italy.
[2]School of Engineering, American University of the Middle East, Kuwait.
[3]DITEN-Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture University of Genoa, Genoa, Italy.
[4]Faculty of Sciences, Lebanese University, Lebanon.

Authors' contributions

This work was carried out in collaboration between all authors. Author WAK analyzed developed and wrote this manuscript, the author SK closely directed, greatly supported and deeply reviewed this study, while the authors RB and KS directed, supported and reviewed this work. All authors read and approved the final manuscript.

Original Research Article

Received: 19 November2013
Accepted: 03 January 2014
Published: 25 January 2014

Abstract

Aims: To extract characters from skewed images without image rotation.
Study Design: This study is designed to be implemented at the gates of Customs, Port Authorities, Terminal Operators and it can also be implemented for vehicles traffic management.
Place and Duration of Study: Lebanon, between September and November 2013.
Methodology: The proposed method consists of sorting the segmented characters according to the X axis, then assigning a Program Line Number to each character based on the skew angle and finally sorting the Program Line Numbers according to their intersection with the Y axis.
Results: Our approach is capable of handling any font and size of characters and it is robust and efficient; regarding its complexity for an image having N lines and M characters, the worst CPU time usage and the worst memory usage is equal to O (NxM) while the network usage and disk usage for one image is O (1) which led to a 0.11 milliseconds response time to extract all container number digits.
Conclusion: Acceleration of segments' extraction from skewed images by avoiding image rotation in order to acquire a faster and more accurate OCR process.

Keywords: Skewed image, rotation angle, line slope, container number.

**Corresponding author: 3854303@studenti.unige.it;*

1. Introduction

Due to the incurred problems related to rotated images taken from real-life scenarios, many papers tackled this issue in order to detect and rotate the skewed images. The existing algorithms consist of rotating the image after calculating its inclination angle [1-3] and before proceeding into the next steps of the Optical Character Recognition (OCR) process.

On the one hand, image rotation can never add details to the image and thus, the result of an adjustment of a skewed image can easily deteriorate the quality of the image (colors less intense, haloes creation around the characters) as shown in figure 1 and figure 2 and may also result in losing some of the surrounding pixels and/or characters as shown in figure 3 and figure 4.

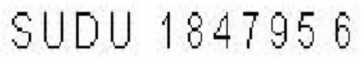

Figure 1. Container number original image

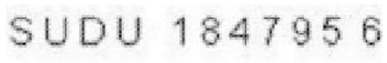

Figure 2. Quality deterioration plus haloes after two 25º rotation

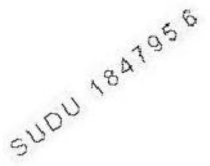

Figure 3. Container number skewed image

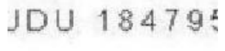

Figure 4. Lost of pixels/characters after 25º rotation

On the other hand, image rotation is time consuming due to the complexity of its computation which is considered as a big problem in real-time systems because it affects the overall performance of the character recognition process and therefore it will be better if it can be avoided.

2. Proposed Method

Based on the above described drawbacks of image rotation, our proposed method consists of extracting the characters of an image subject to a later Optical Character Recognition (OCR) without rotating the image. Our approach is very simple and consists of the following two phases: preparatory and execution phases.

2.1 Preparatory Phase

The preparatory phase consists of relying on any existing characters' segmentation and on any image rotation angle determination.

2.1.1 Characters' Segmentation

Any segmentation approach can be used in order to detect the useful segments from a rotated image [4-11] and all that we need is the upper-left coordinates represented by (X_{min}, Y_{min}) and the lower-right coordinates represented by (X_{max}, Y_{max}) of each segment.

It is worthy to note that in digital image processing, images are read from top to bottom and from left to right, so as shown in figure 5, the X and Y axes are a little bit different than the normal used ones.

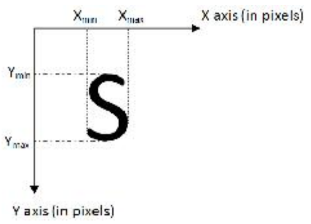

Figure 5. Letter S coordinates

2.1.2 Image Rotation Angle Determination

Any skew detection approach can be used in order to determine the rotation angle of the image (represented hereafter by θ), worthwhile mentioning that such algorithms are classified into the following four main categories [12]: Hough transform [13,14], projection profile [15,16], nearest neighbor clustering [17,18] and interline cross correlation [19-21].

All we need from this sub-phase is to know the image rotation angle in order to start the execution of our proposed approach.

2.2 Execution Phase

The execution phase is the kernel of our approach and is divided into Segments Line Determination and Program Lines Ordering.

2.2.1 Segments Line Determination

In order to acquire the best response time, our approach consists on working all the time in the memory instead of going to any other computer peripheral.

First, we start by determining the coordinates (X, Y) of the center of each segmented character retrieved in the preparatory phase, in the following way:

$$X = (X_{min} + X_{max)}) / 2$$

$Y = (Y_{min} + Y_{max}) / 2$

where (X_{min},Y_{min}) represents the upper-left corner of the segmented character and (X_{max},Y_{max}) represents the lower-right corner of the segmented character.

Second and because Latin characters are written from left to right, we will sort the segmented characters according to their X_{min} value and thus there will be no problem if the segmented characters which were retrieved in the preparatory phase are scrambled (i.e. retrieved in any order).

Third, we initialize the Program Line Number (PLN) variable by zero then we enter the first loop where the variable "I" will hold the first till the last segment of the previously sorted array, where we will:

(a) Check if the segment I having –for example- the center (X_i,Y_i), was previously assigned with a Program Line Number; if the answer is positive we will pass to the next step of the first loop (i.e. we put in I the next segment of the sorted array and we go back to step a) otherwise, we will increment the PLN by one and we assign it to I and then we will enter a second loop where the variable "J" will hold the segment which exists after I till the last segment of the sorted array, where we will:

(b) Check whether the segment J was previously assigned with a Program Line Number; if the answer is positive we will pass to the next step of the second loop (i.e. we put in J the next segment of the sorted array and we go back to step b) otherwise, we check whether the Imaginary Line passing by (X_i, Y_i) crosses the segment J; if the answer is positive, PLN is assigned to J otherwise nothing will be done. Finally, we will pass to the next step of the second loop (i.e. we put in J the next segment of the sorted array then we go back to step b).

It is important to note that:

1. All the segments which are crossed by the Imaginary Line passing by (X_i,Y_i) and having the slope S will have the same PLN because they are in the same line of the rotated image.
2. If an image is rotated by θ, all the lines inside the image will be rotated by the same value.
3. As illustrated in figure 6, The slope of the line passing by the two points (X_1,Y_1) and (X_2,Y_2) is equal to $(Y_2-Y_1)/(X_2-X_1)$ which is equal to the tangent of θ.

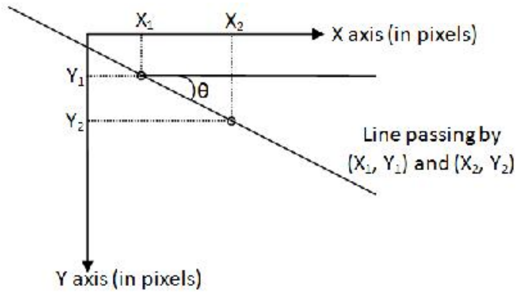

Fig. 6. Relation between the angle rotation and the line slope

Figure 7 shows a line having a positive slope while figure 8 shows a line having a negative slope; it is worthy to note that a line parallel to the X axis has a slope equal to zero.

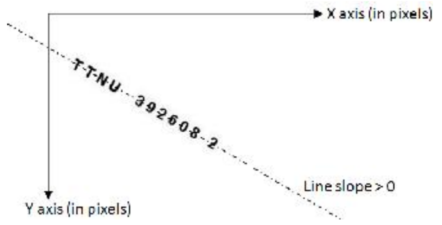

Figure 7. Line having a positive slope.

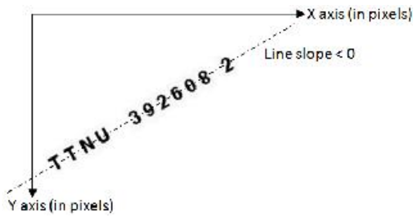

Figure 8. Line having a negative slope.

4. The equation of the imaginary line passing by (X_1,Y_1) and having the slope "S" is equal to:

   $Y= S(X-X_1)+Y_1$

5. The Imaginary Line passing by (X_1,Y_1) and having the slope S crosses a segment if:
   - In case the slope is positive, the Imaginary Line passes by/below its upper-right coordinates (X_{max}, Y_{min}) and by/above its lower-left coordinates (X_{min}, Y_{max}), which means:

   $Y_{min} <= S(X_{max}- X_1) + Y_1$ and
   $Y_{max} >= S(X_{min}- X_1) + Y_1$

Figure 9 illustrates this case.

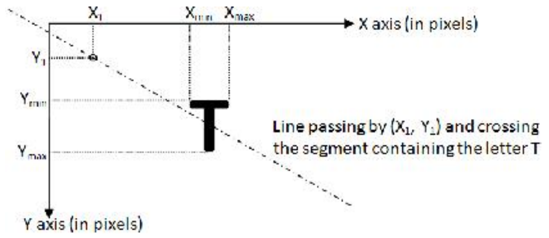

Figure 9. Example of a line crossing a segment

- In case the slope is negative or equal to zero, the Imaginary Line passes by/below its upper-left coordinates (X_{min}, Y_{min}) and by/above its lower-right coordinates (X_{max}, Y_{max}), which means:

$Y_{min} <= S(X_{min} - X_1) + Y_1$ and
$Y_{max} >= S(X_{max} - X_1) + Y_1$

6. At the end of the B1 phase, all the segments will be assigned by a PLN but the PLNs will not reflect the real ordering of the image lines.

2.2.2 Program Lines Ordering

At this stage, all the segments are classified according to their PLNs but which may not be similar to the real order of the image lines because our approach can deal with scrambled segments. To solve this issue, the PLNs will be sorted according to their Imaginary Line intersection with the Y axis. Below, the figures. 10, 11 and 12 will illustrate the problem and its solution.

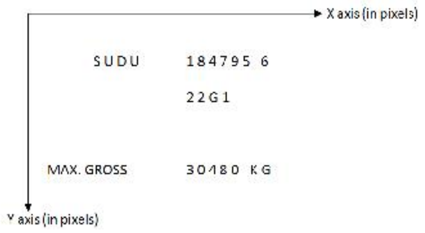

Figure 10. Original binarized upper-right back container image

Figure 11 shows the PLNs state before the Program Line Ordering phase (i.e. this problem occurred due to sorting the segments according to their X_{min} value because our approach can deal with scrambled segmented segments).

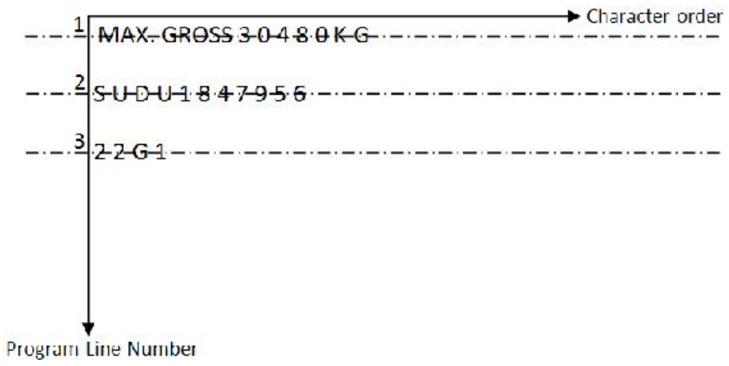

Figure 11. PLNs before program lines ordering phase

Figure 12 shows the PLNs ordered according to their Imaginary Line intersection with the Y axis.

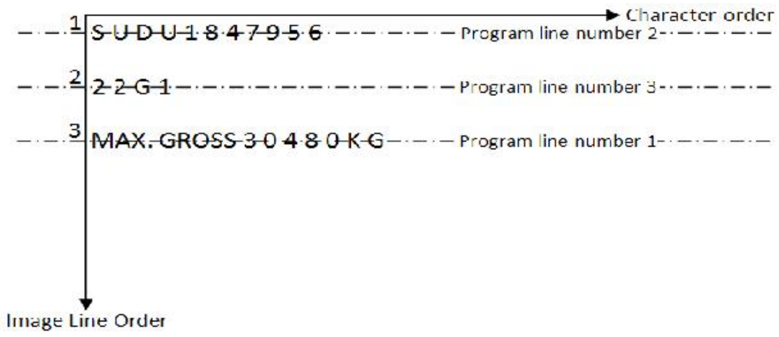

Figure 12. PLNs after the image line ordering phase

2.3 Flowchart

Figure 13 illustrates the whole flowchart of our proposed method.

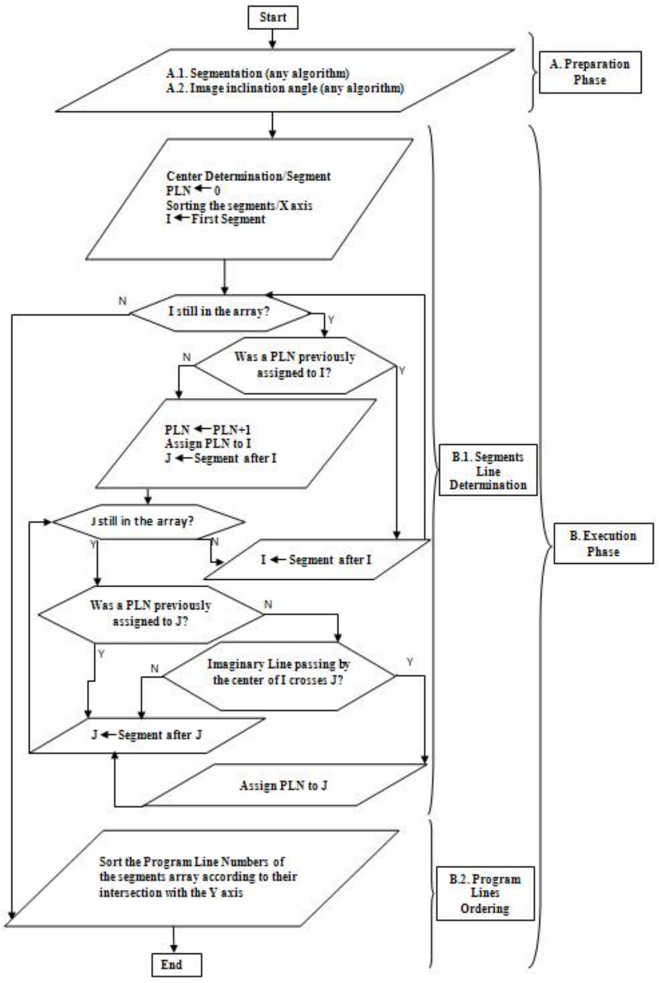
Start
A.1. Segmentation (any algorithm)
A.2. Image inclination angle (any algorithm)
A. Preparation Phase
Center Determination/Segment
PLN ← 0
Sorting the segments/X axis
I ← First Segment
I still in the array?
N
Y
Was a PLN previously assigned to I?
PLN ← PLN+1
Assign PLN to I
J ← Segment after I
B.1. Segments Line Determination
B. Execution Phase
J still in the array?
I ← Segment after I
Was a PLN previously assigned to J?
Imaginary Line passing by the center of I crosses J?
J ← Segment after J
Assign PLN to J
Sort the Program Line Numbers of the segments array according to their intersection with the Y axis
B.2. Program Lines Ordering
End

Figure 13. Flowchart of the whole proposed method

3 Experiments

Our proposed method was initially designed for skewed back side container images and it succeeded to return very good results regarding its performance and its ability to deal with rotated images.

The result of our experiments returned a response time around 0.11 milliseconds for executing our approach on an image containing the container number and the container type while using a laptop having a 2.00 GHz processor and 2 GB of RAM, which makes our proposed method very convenient and highly desirable for real-time applications.

Our proposed approach is very robust because regarding its efficiency and complexity, because supposing that we have N lines and M characters in a container image, the worst CPU time usage is O(NxM) and the worst memory usage is O(NxM) while the network usage and the disk usage for one image is O(1).

It is essential to note that the existence of spaces before or in the middle of the container number or container type doesn't have any negative effect on the result.

We will illustrate in the figures 14, 15 and 16 some samples from our experimental results. To make the figures more readable, each figure will contain two parts where the part above the middle thick line represents the original image and contains two lines (the first one represents the Container Number and the second one represents the Container Type) and the part below the middle thick line represents the extracted segments without applying any image rotation.

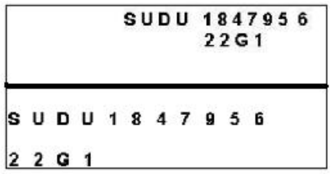

Figure 14. Image rotation = 0º

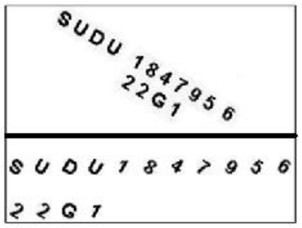

Figure 15. Image rotation = -30º

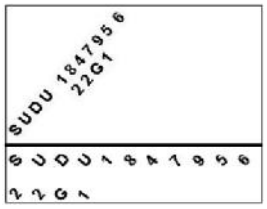

Figure 16. Image rotation = 50º

Worthwhile mentioning that the inclination of the extracted segments doesn't represent any issue because any OCR application can easily recognize them after being trained.

Finally, we would like to highlight that our proposed method can tolerate and deal with distorted segments, due to bad image quality or subject to image filtering, because we designed our approach in a way that a segment is affected by a PLN if the Imaginary Line crosses it (i.e. disregarding if it is higher or lower from the center of the segment).

4. Application Field

Although this paper was intended to read the container number of a skewed image (from the front figure 17, from the back figure 18, left, right or top side of the container), it can also be used to read the container owner name figure 19 and the vehicle plate number figure 20.

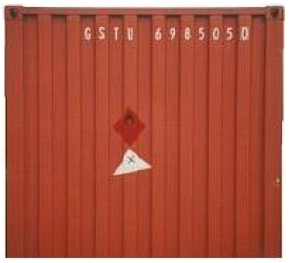

Figure. 17. Container front side

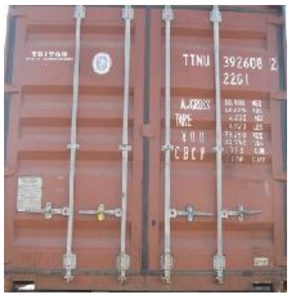

Figure. 18. Container back side

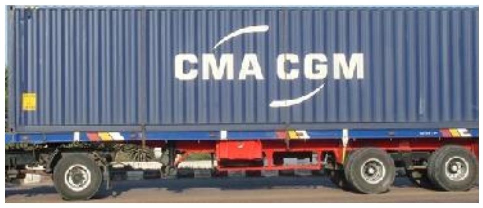

Figure. 19. Image showing the owner of the container

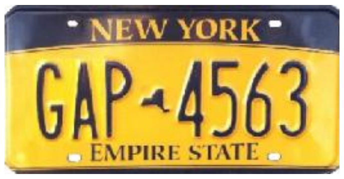

Figure. 20. Vehicule plate.

5. Comparative Study

In this study, we will present the benefit from being able to extract segments from a skewed image without rotating it; to this end, below we will list the execution time for image de-skewing as mentioned in their corresponding paper:

a. The average processing time for skew correction is between 17-110 milliseconds for 0.45-3.0 mega pixel images [22].
b. The time cost varies from 200 to 1000 milliseconds [23].
c. Around 14.7 milliseconds are needed to rotate 256x256 image [24].
d. The average processing time is 77.998 milliseconds if implemented on a CAM-based architecture and 128.367 milliseconds when implemented on RAM-based Architecture [25].

6. Future Works

Because our approach was intended to extract the container number from a skewed container image, so it was designed to read any font and letter size -but having the same height- and thus further enhancements should be done to deal with capital and small letters belonging to the same line; the second issue that needs further research is to calculate, from the segmented characters, the image rotation angle without relying on any other existing algorithm because they are not 100% accurate while detecting the rotation angle of an image which may lead in some cases to incorrect results.

Figure 21 illustrates an example of an inaccurate angle rotation value which led to an incorrect result because the container number was split into two lines. The dash dot lines in the figure 21 represent the Imaginary Lines where the upper part (i.e. the part above the middle thick line) of the figure represents the original image and the lower part represents the result.

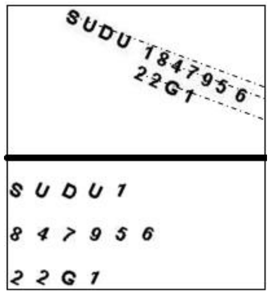

Figure. 21. Incorrect angle rotation.

Finally, we will list an extract of the deviation error as mentioned in their corresponding paper:

a. The average deviation error is estimated within two degrees from the true angle [26]
b. The Rotation from -5^o to 5^o presumed an error deviation of 0.2^o [12]
c. The average deviation error of the skew corrected text lines is from -3^o to 3 ^{o}[22]
d. The skew angle error varies from 0^o to 0.6^o [23]

4. Conclusion

Our approach consists of accelerating the segments' extraction from skewed images by avoiding image rotation; as a result, the overall OCR process will be faster and more accurate.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Duan TD, Du TLH, Phuoc TV, Hoang NV. Building an Automatic Vehicle License-Plate Recognition System. International Conference in Computer Science, RIVF; 2005.

[2] Lee JCM. Automatic character recognition for moving and stationary vehicles and containers in real-life images. International Joint Conference on Neural Networks, IJCNN. 1999;4.

[3] Mollah AF, Majumder N, Basu S, Nasipuri M. Design of an Optical Character Recognition System for Camera-based Handheld Devices", IJCSI International Journal of Computer Science Issues. 2011; 8(1).

[4] Mobahi H, Rao S, Yang A, Sastry S, Ma Y. Segmentation of Natural Images by Texture and Boundary Compression. International Journal of Computer Vision (IJCV). 2011;95(1):86-98.

[5] Batenburg KJ, Sijbers J. Adaptive thresholding of tomograms by projection distance minimization", Pattern Recognition. 2009;42(10);2297-2305.

[6] Arbelaez P, Maire M, Fowlkes C, Malik J. Contour Detection and Hierarchical Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2011;33.

[7] Ren Z, Shakhnarovich G. Image Segmentation by Cascaded Region Agglomeration. CVPR; 2013.

[8] Achanta R,et Al. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE TPAMI;* 2012.

[9] Arbelaez P, Maire M, Fowlkes C, Malik J. Contour detection and hierarchical image segmentation. IEEE TPAMI. 2011;33(5).

[10] Carreira J, Caseiro R, Batista J, Sminchisescu C. Semantic segmentation with second-order pooling. In *ECCV;* 2012.

[11] Karthikeyan V, Sindhu R, Anusha K, Vijith DS. Vehicule License Plate Character Segmentation – A STUDY. International Journal of Computer and Electronics Research. 2013;2.

[12] Papandreou A, Gatos B. A Novel Skew Detection Technique Based on Vertical Projections. International Conference on Document Analysis and Recognition; 2011.

[13] Fan X, Fan G, Liang D. Joint Segmentation and Recognition of License Plate Characters. IEEE International Conference on Image Processing, ICIP; 2007.

[14] Singh C, Bhatia N, Kaur A. Hough transform based fast skew detection and accurate skew correction methods. Pattern Recognition. 2008;41:3528-3546.

[15] Jagannathan J, Sherajdheen A, Deepak RMV, Krishnan N. License plate Character Segmentation using horizontal and vertical projection with dynamic thresholding. International Conference on Emerging Trends in Computing, Communication and anotechnology (ICE-CCN); 2013.

[16] Liolios N, Fakotakis N, Kokkinakis G. Improved document skew detection based on text line connected component clustering. International Conference on Image Processing. 2001;1:1098-1101.

[17] Sarfraz M, Mahmoud SA, Rasheed Z. On Skew Estimation and Correction of Text. Computer Graphics, Imaging and Visualisation, CGIV; 2007.

[18] Lu Y, Tan CL. A nearest-neighbor chain based approach to skew estimation in document images. Pattern Recognition Letters. 2003;24:2315–2323.

[19] Akiyama T, Hagita N. Automatic entry system for printed documents," Publisher Elsevier Science, Pattern Recognition. 1990;23(11):1141– 1154.

[20] Chen M, Ding X. A robust skew detection algorithm for grayscale document image. Proceedings of the Fifth International Conference on Document Analysis and Recognition, ICDAR; 1999.

[21] Chaudhuri A, Chaudhuri S. Robust detection of skew in document images. IEEE Transactions on Image Processing. 1997;6:344–349.

[22] Mollah AF, Basu S, Das N, Sarkar R. A Fast Skew Correction Technique for Camera Captured Business Card Images. IEEE Annual India Conference (INDICON); 2009.

[23] Zhongda Y, Junyu D, Zhiqiang W, Jianxiang S. A Fast Image Rotation Algorithm for Optical Character Recognition of Chinese Documents. International Conference on Communications, Circuits and Systems Proceedings; 2006

[24] Xiao-Gang JIANG, Jian-Yang ZHOU, Jiang-Hong SHI, Hui-Huang CHEN. FPGA Implementation of Image Rotation Using Modified Compensated CORDIC. 6th International Conference on ASIC Proceedings, ASICON; 2005.

[25] Chen YP, Yeh TD, Ni FC, Chang CH. A CAM-Based Pattern Accumulated Vector Method for Real-Time Character Recognition of License Plates. IJCSNS International Journal of Computer Science and Network Security. 2009;9(1).

[26] Hase H, Tanabe K, Tran THH, Tokai S. Multi-Font Rotated Character Recognition Using Periodicity. The Eighth IAPR International Workshop on Document Analysis Systems, DAS; 2008.

© 2014 Al-Khawand et al.; This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
www.sciencedomain.org/review-history.php?iid=410&id=6&aid=3444