FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 30, N° 4, December 2017, pp. 611 - 625
DOI: 10.2298/FUEE1704611V

AUTOMATIC OPTIMIZED DOCUMENT SKEW PRE-PROCESSOR FOR CHARACTER SEGMENTATION ALGORITHM

Vladan Vučković, Boban Arizanović

University of Niš, Faculty of Electronic Engineering, Niš, Serbia

Abstract. *In this paper, as a part of character segmentation algorithm, an automatic optimized document skew correction approach based on Hough transform is presented. The importance of skew correction in document image analysis lies in the fact that further processing is impossible if the document image is skewed. The proposed approach is based on fast implementation of the standard Hough transform which is followed by highly optimized low-level machine code implementation of the image rotation. In order to achieve high computational results, linear image representation is used. The proposed approach results from the aspect of time complexity and skew estimation accuracy which are analyzed and compared with the already existing skew correction approaches. The proposed approach gives better results compared with analogous approach used in related work, but it gives worse results compared with optimized version which exploits a BAG algorithm. Provided results show significant improvement of the standard Hough transform implementation*.

Key words: *Skew correction, Hough transform, character segmentation, spatial transformations, rotation, fast algorithm*

1. INTRODUCTION

As a part of pre-processing techniques used in document image processing, skew correction takes place in the very early stage since further processing is highly affected by document orientation. In general, image processing methods [1], which are suitable for extraction of the information, imply de-skewed image as input. In concrete case of character segmentation process, skew correction is the essential part of document image pre-processing and further processing would be impossible without it. Variety of approaches for document image skew estimation and correction has been presented in the past. Horizontal projection profiles, distribution of feature locations, a Hough transform, and distribution of responses from local directionally sensitive masks are four broad classes of document skew estimation [2], [3]. Methods for evaluation of document skew estimation techniques are of particular interest for researchers [4], [5]. Hough transform is the most frequent technique for skew estimation [6]-[8]. Most skew estimation and correction approaches are based on

Received December 18, 2016; received in revised form February 2, 2017
Corresponding author: Vladan Vučković
University of Niš, Faculty of Electronic Engineering, Computer Department, P.O. Box 73, 18000 Niš, Serbia
(E-mail: vladanvuckovic24@gmail.com)

the usage of the Hough transform modifications [9]-[11]. One recent work proposes a method based on Hough space derivatives in order to identify directions with sudden changes in their projection profiles [12]. Some approaches based on using the Hough transform do not use a voting scheme or use its modifications [13], [14]. Hough transform which exploits the cross-correlation property between the pixels in vertical lines is proposed in [15]. Many recent works based on the usage of the Hough transform are focused on improving the processing time [16], [17]. One approach for fast skew correction is based on the usage of the block adjacency graph (BAG) algorithm before applying the Hough transform [18]. Modification of the Hough transform voting scheme can also be efficient [19]. Approaches for improving the processing time of the Hough transform are usually based on determination of boundary boxes which represents edges of important image elements. In case of document images, if characters do not have the same height, this idea is not applicable and other approaches based on the usage of the boundary boxes of connected components are used [14]. Beside the Hough transform, other approaches for skew correction are exploited. Mathematical morphology proved to be useful for skew correction and can perform on both, binary and grayscale images [20]. Other skew correction methods are based on the usage of the correlation functions and geometric text-line models [21], [22]. Radon transform based projection profile technique is used for skew correction of handwritten words [23]. Another work is based on robust borderlines which are extracted using the run length based method [24]. Straight-line fitting based method can be also used for skew detection and correction [25]. Unlike the previously mentioned approaches which work in spatial domain, frequency domain approach based on using the Fourier transform and KNN clustering can be also exploited [26]. Hough transform is also used for other purpose, such as video processing [27], [28] and face recognition [29].

The proposed approach for automatic skew correction of the document images is intended for usage in the pre-processing stage of character segmentation system. Character segmentation system was initially designed for machine-typed documents, but can be used for machine-printed documents, as well. The focus of this approach is fast implementation of the standard Hough transform using the pointer arithmetic, which is followed by highly optimized low-level machine code implementation of the image rotation, which exploits the efficient generalized architecture for geometrical image transformations. In order to achieve fast implementation, linear image representation is used. The proposed approach is compared with classical implementation of the Hough transform and also with the approach which exploits BAG algorithm as a pre-processing stage of the Hough transform [18]. From the aspect of time complexity, the proposed approach uses the rotation algorithm which gives better results than all rotation algorithms presented in [18]. When it comes to time complexity of the whole skew correction approach, the proposed approach performs approximately 2.5 faster than the classical implementation. Compared with a version which exploits BAG algorithm, the proposed approach gives worse results, but the proposed ultra-fast architecture in combination with pointer arithmetic and highly optimized low-level machine code implementation, could be also used in combination with BAG algorithm in order to provide even better results than the already existing ones. From the aspect of skew estimation accuracy, the proposed approach gives results as good as the existing ones. In order to show the proposed approach importance for character segmentation system and how the skew correction affects the further character segmentation process, Nikola Tesla's documents from the "Nikola Tesla Museum" in Belgrade are used [30].

This paper is organized as follows: in section 2 the theoretical background of the Hough transform is provided. In section 3 the flowchart of the complete character segmentation system is given and ultra-fast generalized image transformation approach used for the proposed approach is presented. Section 4 offers implementation details for the main parts of the proposed approach, including the implementation details for the ultra-fast architecture which is used. Section 5 provides experimental results for the proposed approach, and shows the comparison of the proposed approach with already existing skew correction approaches. Also, this section shows how the skew correction pre-processing algorithm affects the character segmentation results. In section 6 the summary of the proposed approach performances is given.

2. THEORETICAL BACKGROUND

The central task of the proposed approach for document image skew correction is determination of the rotation angle in order to fix the document skew and make the document suitable for the character segmentation process. For this purpose, the standard Hough transform is used [6]-[8]. This part of the skew correction approach provides the detection of the rotation angle in the range from -90 to 90 degrees.

The Hough transform was primarily intended for the purpose of straight line detection, and later has been generalized for detection of arbitrary shapes [7]. Our focus will be on line detection, more precisely on detection of the angle between the line which passes through the most pixels in the image and x-axis. Taking the document images into consideration, this angle represents the rotation angle of the document. The key of the Hough transform is in representation of a line. In two-dimensional Euclidian space, lines can be represented using the slope and intercept as follows:

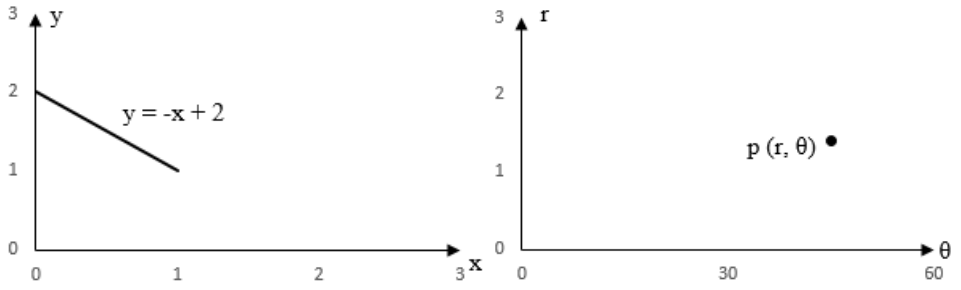
$$y = kx + n \tag{1}$$

However, a line is not defined by x and y coordinates. Complete information about a line is given by parameters k and n, and lines can be represented using pair (k, n). The problem with this representation lies in the unavailability to represent vertical lines, since the vertical lines make angle with x-axis equal to 90 degrees and tan 90° equals infinity. For that reason, the alternative way of line representation using parameters r and θ is used. In this representation r is a vector perpendicular to the given line and represents distance between the origin and the given line, and θ is the angle between the vector r and x-axis. The relation between these parameters is described using the following equations:

$$r = x\cos(\theta) + y\sin(\theta) \tag{2}$$

$$y = -\frac{\cos(\theta)}{\sin(\theta)}x + \frac{r}{\sin(\theta)} \tag{3}$$

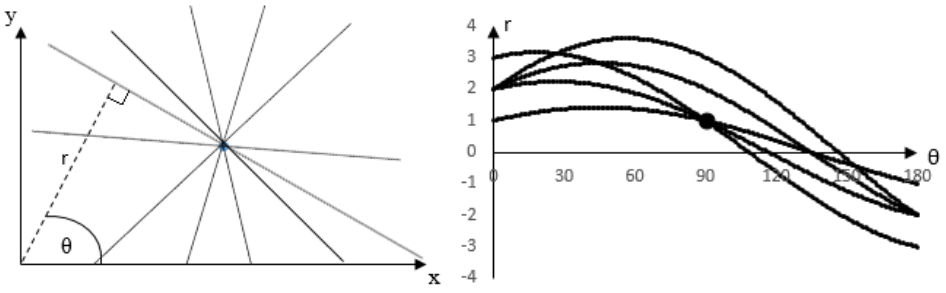
for $\theta \in [0, 180]$ and $r \in R$. Therefore, the Hough transform uses the Hough space, where each line is represented as a point using a pair (r, θ). Mapping from the Euclidian to the Hough space is shown in Fig. 1 (a) and (b).



(a) (b)

Fig. 1 Mapping of line in Euclidian space to point in Hough space:
(a) line in Euclidian space and (b) corresponding point in Hough space

The basic idea of the Hough transform is representing for each black pixel in the binary image all lines that can pass through that pixel. As it is shown before, in the Hough space each line is represented as a point, therefore it is clear that all lines passing through the given pixel can be represented as a sine curve. An illustration of this process is shown in Fig. 2 where spectrum of lines passing through the single point in the Euclidian space is shown in (a), and line spectrums for different points in the Hough space are shown in (b). All points from different curves represent lines which pass through at least one black pixel in the binary image. If different points from different curves have the same values for r and θ, it means that those points actually represent the same line, and the number of pixels that lies on that line is equal to the number of points with these coordinates. As the output of the algorithm, the θ coordinate of the line with most black pixels is taken. This value θ represents an angle of rotation and is used for the image rotation.



(a) (b)

Fig. 2 Representation of line spectrum: (a) for one point in the
Euclidian space and (b) for different points in the Hough space

The important part of the Hough transform is a voting matrix which is used in the process of counting the lines with same parameters (r, θ). Another term used for the voting matrix is accumulator. In a general case, the number of dimensions of the voting matrix depends on the shape we want to detect and is equal to the number of unknown parameters. In case of lines, the number of unknown parameters is 2, therefore a matrix is used. Dimensions of a voting

matrix may vary and they define the precision of calculations. In our case, both dimensions r and θ have a precision of 1 pixel for r and 1 degree for θ. An illustration of the voting matrix is shown in Fig. 3.

	Θ_1	Θ_2	. . .	θ_{MAX}
R_1	17	10	14	28	60
R_2	22	31	54	83	74
.	13	98	57	44	61
.	15	19	63	51	18
.	12	21	35	24	32
R_{MAX}	4	9	47	33	55

Fig. 3 Representation of the voting matrix

The pseudo-code for the Hough transform is as follows:

```
Input:
    Image f
Output:
    Angle a;
  1:    RMax = Sqrt(ImageWidth*ImageWidth + ImageHeight*ImageHeight) + 1
  2:    Theta = 180
  3:    Fill voting matrix V[RMax][Theta + 1] with zeros
  4:    For each black pixel f (x, y) do
  5:    for K = 0 to Theta do
  6:    Angle = K * Pi / 180
  7:    R = x*sin(Angle) + y*cos(Angle)
  8:    if R ≥ 0 and R < RMax then
  9:    V[R][K] = V[R][K] + 1
10:    end if
11:    end for
12:    a = FIND-MAX-POS-Y(V)
13:    a = a - 90
14:    return a
```

3. Proposed Approach

The proposed optimized automatic skew correction approach represents the extended version of the character segmentation algorithm for machine-typed documents. The algorithm is extended with optimized skew correction pre-processing approach in order to fix the potential document skew. Skew correction approach used here exploits the fast implementation of the standard Hough transform used for determining the angle of document

skew, and highly optimized machine code implementation of the image rotation. Efficient image rotation is achieved using the ultra-fast generalized architecture for geometrical image transformations. In order to achieve the fast implementation of the proposed approach, linear image representation is used. The flowchart of the complete character segmentation system is shown in Fig. 4.

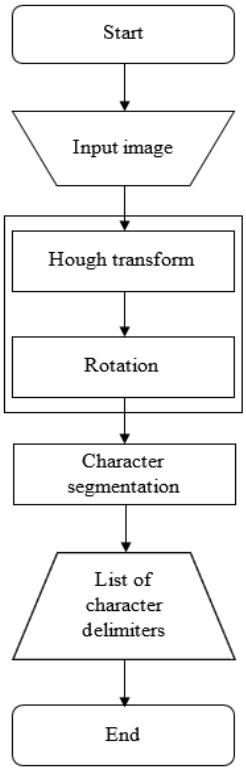


Fig. 4 Flowchart of the complete character segmentation system

Rotation transformation is achieved using the ultra-fast architecture for geometrical image transformations. This architecture is generalized and can be also used for other spatial transformations. The architecture scheme is shown in Fig. 5.

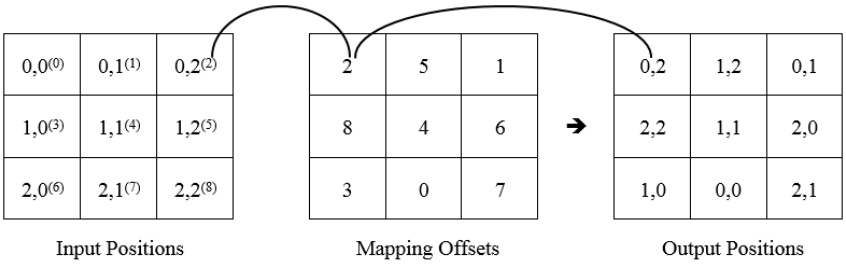


Fig. 5 Ultra-fast architecture for image transformation

The architecture used for image rotation is based on using the mapping offsets which represent the transformation matrix for chosen transformation. The transformation matrix is a matrix of offsets where each offset represents the offset relative to the first element of the input matrix. Using the mapping offsets each input position is mapped to the specific output position. In practice, this architecture provides that each pixel in the input image can be mapped to the pre-computed position in the output image. Calculation of the mapping offsets for the chosen transformation with specific parameters is performed at the start and whenever the transformation needs to be applied, already calculated mapping offsets are used. In this case, mapping offsets are calculated using the standard rotation transformation pair:

$$S_X = x\cos(\theta) + y\sin(\theta) \tag{4}$$

$$S_y = -x\sin(\theta) + y\cos(\theta) \tag{5}$$

where angle θ is previously estimated document skew angle. Since the mapping offsets are valid for a given angle of rotation, it would be necessary to calculate the mapping offsets for different angles to make it possible to combine different mapping offsets for achieving a rotation for the desired angle. This approach proved to be very efficient and does not depend on the type of image transformation.

The bad side of the presented ultra-fast architecture for geometrical image transformations is the memory usage. In order to achieve the fastest possible computational performances, the mapping offsets and other support lookup tables are loaded at the start and kept in memory all the time.

4. FAST IMPLEMENTATION

The focus of this paper is the fast implementation of the automatic optimized skew correction approach. In order to achieve the fast implementation, image is represented as a one dimensional array. This linear image representation is shown in Fig. 6. This image representation provides the direct memory access to the pixel intensity values using the pointer arithmetic. Furthermore, implementation of the image rotation is achieved using the highly optimized low-level machine code.

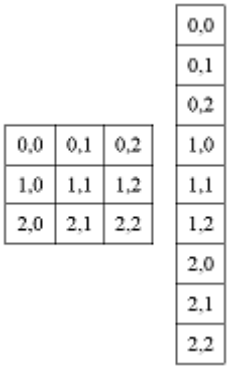


Fig. 6 Linear image representation

Implementation of both parts of the skew correction approach exploits the lookup support tables for values of trigonometric functions. This is the common way to avoid multiple calculations with the same parameters inside big loops. Pascal implementation of the ultra-fast image transformation architecture adapted for image rotation is shown in the following listing:

```
for J := 0 to N do {Number of transformation arrays}
  begin
    SetLength(R[J], Count);   {Set each transformation array to be equal to
                               the number of pixels}

    S := DPtr^;                 {Get value for Sin from lookup table}
    Inc(DPtr);                  {Increment pointer}
    C := DPtr^;                 {Get value for Cos from lookup table}
    Inc(DPtr);                  {Increment pointer}

    RPtr := @R[J, 0];             {Pointer to current transformation array}

    X := Trunc(HTemp * C + WTemp * S); {Determine the X coordinate of the
                                        image center after rotation}
    Y := Trunc(-HTemp * S + WTemp * C); {Determine the Y coordinate of the
                                         image center after rotation}
    OffsetX := WTemp - Y; {Determine the offset from image center for X
                           coordinate}
    OffsetY := HTemp - X; {Determine the offset from image center for Y
                           coordinate}

    ImageStartPtr := @Image[0]; {Source image pointer}
    Ptr1 := @PosMap[0];             {Pointer to support lookup table}
    Ptr2 := @PosMap[1];             {Pointer to support lookup table}

    for I := 0 to Count - 1 do
    begin
      X := Trunc(Ptr1^ * C + Ptr2^ * S) + OffsetY; {Determine the X coordinate
                                                    after rotation}
      Y := Trunc(-Ptr1^ * S + Ptr2^ * C) + OffsetX; {Determine the Y
                                                     coordinate after rotation}

      if (X >= 0) and (X < Height) and (Y >= 0) and (Y < Width) then
                                                    {If coordinates are valid}
        RPtr^ := X * Width + Y{Store offset to transformation array}
      else
        RPtr^ := -1;                      {Store -1}

      Inc(RPtr);                           {Increment Pointer}
      Inc(Ptr1, 2);                        {Increment Pointer}
      Inc(Ptr2, 2);                        {Increment Pointer}
    end;
  end;
```

The following subsections provide implementation details for the main parts of the proposed approach.

4.1. Hough transform implementation

In order to achieve the fast implementation of the Hough transform, the pointer arithmetic is used. For both, linear and matrix representation of the image, classical implementation is

not suitable due to slow indexed access to the array elements. Using the linear image representation combined with pointer arithmetic, the direct memory access is achieved. The following code represents the Pascal implementation of the standard Hough transform based on using the pointer arithmetic, which is capable to determine skew angles in the range from -90 to 90 degrees. It should be mentioned that using the different parameters in the following implementation, it is possible to estimate angles in different ranges which is exploited for obtaining the experimental results.

```
DMax := Trunc(Sqrt(Width * Width + Height * Height)) + 1; {Maximal allowed
                                                          line length}
  Teta := 90;       {Angle which defines the range of estimation angle}
  Ptr1 := @ImageBinary[0];  {Pointer to binary image which is being processed}
  Ptr2 := @PosMap[0];       {Pointer to support lookup table}
  Ptr3 := @VotingMatrix[0]; {Pointer to voting matrix}
  for I := 0 to Count - 1 do  {Main loop}
  begin
    if Ptr1^ = 1 then         {Is it black pixel?}
    begin
      J := Ptr2^;             {Get X value from linear offset lookup table}
      Inc(Ptr2);              {Increment pointer}
      L := Ptr2^;             {Get Y value from linear offset lookup table}
      Inc(Ptr2);              {Increment pointer}
      Ptr5 := @SinCos[0];     {Pointer to lookup table of trigonometric values
                               - Sin}
      Ptr6 := @SinCos[1];     {Pointer to lookup table of trigonometric values
                               - Cos}
      for K := 0 to Teta do   {Loop through all angles}
      begin
        D := Trunc(J * Ptr5^ + L * Ptr6^);{Determine the line length for
                                           current parameters}
        if (D >= 0) and (D < DMax) then  {Is it in range?}
        begin
          Ptr4 := Ptr3;            {Get the starting pointer of voting matrix}
          Inc(Ptr4, D * Teta + K); {Increment the pointer bydetermined offset}
          Ptr4^ := Ptr4^ + 1;      {Increment the voting matrix value}
        end;
        Inc(Ptr5);                 {Increment pointer}
        Inc(Ptr6);                 {Increment pointer}
      end;
    end
    else
      Inc(Ptr2, 2);                {Increment pointer}
    Inc(Ptr1);                     {Increment pointer}
  end;
Result := MaxInd(VotingMatrix) - 90;{Final estimated angle}
```

4.2. Rotation implementation

Image rotation is performed using the previously described ultra-fast architecture for geometrical image transformations. This approach proved to be very efficient and performs almost 50 times faster than standard approach for image rotation. In order to achieve the highest computational performances, the highly optimized low-level machine code implementation of the image rotation is used. The following listing shows the machine routine for image rotation:

```
asm
  pushad{Push all registers to stack}
  mov ecx,Count{Number of pixels to process}
  mov esi,RPtr{Pointer to R transformation array}
  mov ebx,ImageSrcPtr{Source image pointer}
  mov edi,ImageDstPtr{Destination image pointer}
@main:                    {Main loop}
  LODSD                   {Load current offset from R transformation array}
  mov edx,eax             {Save current offset}
  or eax,eax              {Is it -1?}
  js @init                {If true, jump to label init}
  shl edx,2               {Offset * 4}
  mov eax,[edx+ebx]{Calculate final offset and load value from source to EAX}
  STOSD                   {Store loaded value from EAX to destination}
  dec ecx                 {Decrement counter}
  jnz @main               {If not zero, loop again through ECX}
  jmp @ex                 {Else,jump to ex label}
@init:                    {Label init}
  mov eax,WHITE_COLOR{Store white color definition to EAX}
  STOSD                   {Store value from EAX to destination}
  dec ecx                 {Decrement counter}
  jnz @main               {If not zero, loop again through ECX}
@ex:                      {Label ex}
  popad                   {Pop up all registers from stack}
end;
```

5. EXPERIMENTS

Testing of the proposed optimized skew correction approach is performed on PC machine with an AMD Quad Core Processor running at 3.1 GHz and 4 GB RAM installed. The proposed approach performances from the aspect of time complexity are analyzed and compared with related work. Beside the time complexity, skew estimation accuracy results are also provided. For the purpose of demonstrating the proposed approach performances and importance of the skew correction for character segmentation process, Nikola Tesla's documents from the "Nikola Tesla Museum" in Belgrade are used. The proposed approach performances are compared with results provided in [18].

Table 1 shows comparison of rotation processing time for the proposed approach and algorithms analyzed in [18].

Table 1 Comparison of processing time for image rotation

Image dimensions (px)	C. Singh et al. [18] processing time (ms)								Proposed approach processing time (ms)
	Float rotation		Integer rotation		Fast implementation		Bresenham's line like algorithm		
1249x1249	15	172	15	78	15	78	16	31	6
4148x4068	218	1875	156	844	157	841	235	313	64

Provided results show the efficiency of the rotation algorithm proposed as a part of skew correction approach. C. Singh et al. provided results for different algorithms, including results for forward and inverse rotation for each analyzed rotation algorithm. An important fact is that the proposed rotation approach is not dependent on complexity of calculations, thus both, forward and inverse rotation, will be performed with the same processing time. Taking this fact into consideration, results show that the proposed approach gives better results than any of the rotation algorithms analyzed in [18].

Beside the rotation algorithm processing time, other important aspects of the skew correction approaches are the Hough transform processing time and skew estimation accuracy. C. Singh et al. used BAG algorithm in the process of skew estimation and they compared the overall processing time with standard approach which does not use BAG algorithm. Table 2 shows the comparison of the proposed approach results with results provided in [18]. Results given in Table 2 show that the proposed approach, from the aspect of time complexity, gives better results than classical implementation of the Hough transform and worse results than implementation which exploits BAG algorithm. Considering the fact that the proposed approach performs approximately 2.5 times faster than classical implementation, this is a significant improvement. Also, taking into consideration that the proposed approach exploits ultra-fast architecture for image transformation, BAG algorithm could be used in combination with the proposed approach to provide even faster results than the already presented ones. Results provided in Table 1 and Table 2 are obtained using the document images with the same characteristics as images used in related work [18].

When it comes to the importance of the skew correction for character segmentation system, character segmentation system which is in the background of the proposed skew correction approach proved to be very sensitive to document skew. Document images skewed for random angles are used for obtaining results. It is shown that document images skewed even for a small angle higher than 2° highly decrease the successful character segmentation percentage.

Table 2 Comparison of overall processing time and skew estimation accuracy

Image dimensions (px)	% of black pixels	Skew angle	C. Singh et al. processing time (ms)		Difference in skew (BAG)	Difference in skew (no BAG)	Proposed approach processing time (ms)	Difference in skew
			BAG	No BAG				
4168x4088	7.16	0	593	3875	0.00	0.000	1548	0.00
4308x4231	6.6	2	610	3969	0.06	-0.855	1589	-0.35
4508x4436	6.1	5	704	3968	0.079	0.079	1585	-0.15
4815x4750	5.3	10	797	4025	0.08	0.08	1612	-0.10
4981x4921	4.9	13	860	4079	-0.48	-0.65	1631	-0.25
5272x5222	4.42	19	953	4188	-0.19	0.195	1654	0.12
5654x5624	3.8	30	1046	4234	0.02	-0.26	1687	0.20
5838x5838	3.57	45	1234	4484	0.00	0.00	1711	0.00
4088x4168	7.1	90	522	3781	0.00	0.00	1536	0.00
5267x5315	4.3	110	985	4134	-0.43	-0.43	1644	-0.47
5739x5759	3.6	150	1374	4655	-0.32	-0.19	1720	-0.20
1904x2588	13.88	0	280	2034	0.20	0.00	478	0.00
1993x2653	14.07	2	250	2281	-0.73	-1.11	515	-0.88
2122x2744	11.69	5	235	2094	-0.43	-1.19	487	-0.65
2324x2879	9.47	10	281	2124	0.49	-1.16	492	-0.45
2437x2950	9.47	13	327	2125	-0.70	-0.48	489	-0.32
2643x3067	8.4	19	328	2140	-1.14	-0.57	493	-0.60
2943x3193	7.24	30	375	2172	-0.95	-0.95	500	-0.95
3176x3176	6.75	45	407	2187	-1.14	-0.64	508	-0.42
2588x1904	13.88	90	171	2078	0.00	0.00	481	0.00
3083x2674	8.25	110	281	2141	-0.65	-0.43	496	-0.71
2943x3193	7.24	150	375	2172	0.96	0.96	504	0.90

This characteristic of the character segmentation algorithm is due to its nature. The graph that shows the dependency of the successful character segmentation percentage from the skew angle is shown in Fig. 7.

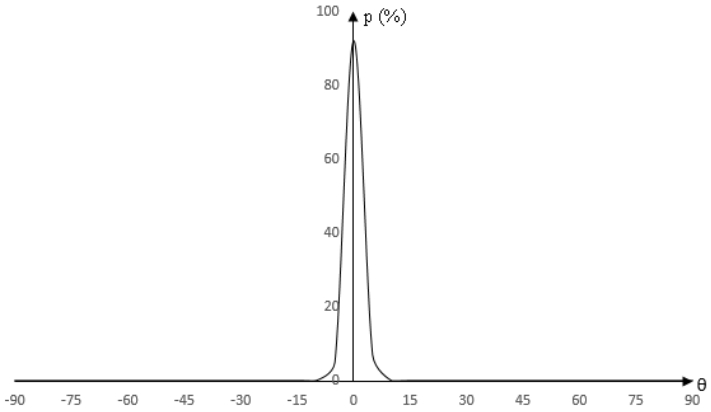
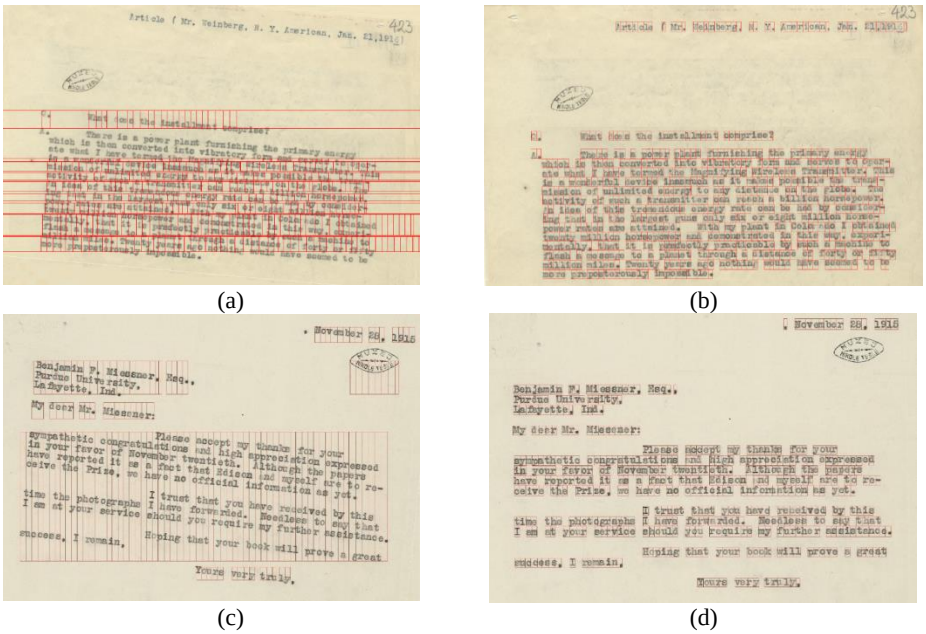


Fig. 7 Character segmentation results as a function of document skew angle

This graph shows that even a small document skew represents a big problem for character segmentation algorithm. Visual results of the extended character segmentation algorithm are provided using the Nikola Tesla's documents from the "Nikola Tesla Museum" in Belgrade. These results are shown in Fig. 8 a), b), c), and d).



(a) (b)

(c) (d)

Fig. 8 Extended character segmentation results for skewed and de-skewed original Nikola Tesla's documents: a) first skewed document, b) first de-skewed document, c) second skewed document, d) second de-skewed document

The character segmentation results shown in Fig. 8 confirm the previous conclusion based on the graph results. Based on the scanned document images from the "Nikola Tesla Museum", it is clear that further character segmentation is impossible without skew correction performed in the very early stage of the character segmentation process.

6. CONCLUSIONS

In this paper, the optimized Hough transform based approach for skew correction is presented as an essential pre-processing part of character segmentation system. Character segmentation system is initially designed for machine-typed documents, but can be used for machine-printed documents as well. In section 2 the theoretical background of the Hough transform is provided. In section 3 the flowchart of the complete character segmentation algorithm is shown and description of the proposed skew correction approach is provided. The proposed approach uses the ultra-fast generalized image transformation architecture for achieving high computational performances. In order to achieve fast implementation, linear image representation is used. Ultra-fast image transformation architecture is used for implementation of the image rotation algorithm, which is implemented using the highly optimized low-level machine code. The standard Hough transform is implemented using the pointer arithmetic. For both implementations, support lookup tables are used. In section 4, the experimental results for the proposed approach from the aspect of time complexity and estimation accuracy are given and are compared with existing approaches. Also, the results which show how the skew correction affects the character segmentation process, are given. Based on the results, the proposed approach performs approximately 2.5 faster than the classical implementation used in [18]. Also, the proposed approach gives worse results than skew correction approach which exploits a BAG algorithm. Although the proposed approach does not give the best results, it could be used in combination with a BAG algorithm to provide even better results than existing ones. The estimation accuracy results show that the proposed approach gives results as good as results provided in the related work. On the other side, it is clear that character segmentation of skewed documents is impossible and gives bad results without a skew correction. Since character segmentation system is initially designed for needs of the "Nikola Tesla Museum", namely for conversion of Nikola Tesla's scanned documents to electronic form, the original Nikola Tesla's documents from the "Nikola Tesla Museum" are used for testing of the complete character segmentation algorithm performances. Also, the official evaluation of the complete character segmentation system performances will be performed at the "Nikola Tesla Museum". Our future work will be focused on the automatization of the character segmentation algorithm manual parts, improving its performances, the optimization of the complete algorithm including the proposed skew correction approach, and integration of the character segmentation system into the complete real-time OCR system.

Acknowledgments: *This paper is supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia (Project III44006-10), Mathematical Institute of Serbian Academy of Science and Arts (SANU) and Museum of Nikola Tesla (providing original typewritten documents of Nikola Tesla).*

REFERENCES

[1] S. S. Cvetković, S. V. Nikolić and S. Ilić, "Effective combining of color and texture descriptors for indoor-outdoor image classification", *Facta Universitatis: Electronics and Energetics*, vol. 27, no. 3, pp. 399-410, 2014.

[2] J. J. Hull, "Document image skew detection: Survey and annotated bibliography", *Series in Machine Perception and Artificial Intelligence*, vol. 29, pp. 40-66, 1998.

[3] H. S. Baird, "The skew angle of printed documents", Document image analysis, pp. 204-208, 1995.

[4] A. Papandreou et al., "ICDAR2013 Document Image Skew Estimation Contest (DISEC'13)", In Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR), 2013.

[5] A. D. Bagdanov and J. Kanai, "Evaluation of document image skew estimation techniques", In SPIE Proceedings 2660: Document Recognition III, 1996, pp. 343-354.

[6] P. Mukhopadhyay and B. B. Chaudhuri, "A survey of Hough Transform", *Pattern Recognition*, vol. 48, no. 3, pp. 993-1010, 2015.

[7] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures", In proceedings of the Communications of the ACM, vol. 15, no. 1, pp. 11-15, 1972.

[8] S. N. Srihari and V. Govindaraju, "Analysis of textual images using the Hough transform", *Machine Vision and Applications*, vol. 2, no. 3, pp. 141-153, 1989.

[9] O. G. Okun, "Geometrical approach to skew detection for documents containing the Latin/Cyrillic characters", In Proceedings of the SPIE, vol. 3811: Vision Geometry VIII, 1999, pp. 357-365.

[10] A. Boukharouba, "A new algorithm for skew correction and baseline detection based on the randomized Hough Transform", *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 1, pp. 29-38, 2016.

[11] D. Kumar and D. Singh, "Modified approach of Hough transform for skew detection and correction in documented images", *International Journal of Research in Computer Science*, vol. 2, no. 3, pp. 37-40, 2012.

[12] F. Stahlberg and S. Vogel, "Document Skew Detection Based on Hough Space Derivatives", In Proceedings of the 13th International Conference on Document Analysis and Recognition, 2015.

[13] V. Shapiro, "Accuracy of the straight line Hough Transform: The non-voting approach", *Computer Vision and Image Understanding*, vol. 103, no. 1, pp. 1-21, 2006.

[14] S. Guo et al., "An improved Hough transform voting scheme utilizing surround suppression", *Pattern Recognition Letters*, vol. 30, no. 13, pp. 1241-1252, 2009.

[15] B. Gatos, N. Papamarkos and C. Chamzas, "Skew detection and text line position determination in digitized documents", *Pattern Recognition*, vol. 30, no. 9, pp. 1505-1519, 1997.

[16] U. Pal and B. B. Chaudhuri, "An improved document skew angle estimation technique", *Pattern Recognition Letters*, vol. 17, no. 8, pp. 899-904, 1996.

[17] A. Amin et al., "Fast algorithm for skew detection", In Proceedings of the SPIE 2661: Real-Time Imaging, 1996, pp. 65-77.

[18] C. Singh, N. Bhatia and A. Kaur, "Hough transform based fast skew detection and accurate skew correction methods", *Pattern Recognition*, vol. 41, no. 12, pp. 3528-3546, 2008.

[19] L. A. F. Fernandes and M. M. Oliveira, "Real-time line detection through an improved Hough transform voting scheme", Pattern Recognition, vol. 41, no. 1, pp. 299-314, 2008.

[20] L. A. Najman, "Using mathematical morphology for document skew estimation", In Proceedings of the SPIE 5296: Document Recognition and Retrieval XI, 2003, pp. 182-192.

[21] G. Bessho, K. Ejiriand J. F. Cullen, "Fast and accurate skew detection algorithm for a text document or a document with straight lines", In Proceedings of the SPIE 2181: Document Recognition, 1994, pp. 133-141.

[22] J. van Beusekomand T. M. Breuel, "Resolution independent skew and orientation detection for document images", In Proceedings of the SPIE 7247: Document Recognition and Retrieval XVI, 2009, pp. 72470K-72470K-8.

[23] R. Kapoor, D. Bagai and T. S. Kamal, "A new algorithm for skew detection and correction", *Pattern Recognition Letters*, vol. 25, no. 11, pp. 1215-1229, 2004.

[24] H. Liu et al., "Skew detection for complex document images using robust borderlines in both text and non-text regions", *Pattern Recognition Letters*, vol. 29, no. 13, pp. 1893-1900, 2008.

[25] Y. Cao, S. Wang and H. Li, "Skew detection and correction in document images based on straight-line fitting", *Pattern Recognition Letters*, vol. 24, no. 12, pp. 1871-1879, 2003.

[26] J. Fabrizio, "A precise skew estimation algorithm for document imagesusing KNN clustering and Fourier transform," In Proceedings of the IEEE International Conference on Image Processing (ICIP), 2014.

[27] A. Chan-Hon-Tong, C. Achard and L. Lucat, "Simultaneous segmentation and classification of human actions in video streams using deeply optimized Hough transform", *Pattern Recognition*, vol. 47, no. 12, pp. 3807-3818, 2014.
[28] C. Tu et al., "Vehicle Position Monitoring Using Hough Transform", In Proceedings of the International Conference on Electronic Engineering and Computer Science, vol. 4, pp. 316-322.
[29] R. Varun et al., "Face Recognition Using Hough Transform Based Feature Extraction", *Procedia Computer Science*, vol. 46, pp. 1491-1500, 2015.
[30] V. Vučković and S. Spasić, "3-D stereoscopic modeling of the Tesla's Long Island", *Facta Universitatis: Electronics and Energetics*, vol. 29, no. 1, pp. 113-126, 2016.