EECS 490 DIGITAL IMAGE PROCESSING

December 2004

SESSION 1 Document Deskewing

Mike Adams	Algorithm for Text Document De-Skewing
Nattapon Chaimanonart	Document Skew Detection Using the Hough Transformation
Wen-Teng Chang	Skew Correction for Text Document Using Fourier Domain Analysis
Dmitriy Goldman	Image De-Skewing Using Fourier Transform
Issac Hirt	Image Skew Detection and Correction using Morphology
Ruchi Kothari	Detection and Correction of Skew in Document Images
Michael K. Lee	Optical Character Recognition Preparation Using MATLAB
Deng-Hung Liu	Skew Correction for Document Images Using Gradient and Fourier Analysis
Daniel Pendergast	Multi-technique Algorithm for De-skewing and Text Bounding for OCR Applications
Iouri Petriaev	OCR Preparation: Design of an Algorithm for Document Analysis Using the Global Processing via Hough Transform

Algorithm for Text Document De-Skewing

Mike Adams

Department of Electrical Engineering and Computer Science,
Case Western Reserve University, Cleveland, OH, Email: mda10@case.edu

ABSTRACT

This paper presents a method for the skew correction of text documents in digital image format, using primarily the Hough Transform. The Hough Transform provides a good estimate of line angles in an image and this can be extended to lines of text. Positive angles between 0 and 90 degrees are estimated correctly in both test images while negative angles are estimated correctly only after implementing a solution found empirically. This routine also fits the resulting corrected image to a given bounding box.

KEYWORDS

Text de-skewing, Hough Transform, Morphology

INTRODUCTION

The skew correction of text documents is the starting point for digital document analysis, because such analysis cannot proceed when a document is skewed even at a small angle. This algorithm therefore seeks to detect a document's skew angle based on the angle at which the text lies. Assuming that a text's angle is the same as that of the entire document, the Hough transform is used to detect the skew angle.

HOUGH TRANSFORM

The Hough Transform is a mapping of lines in Cartesian x-y space to radii and angles in ρ-θ space. Lines in x-y space are of the form,

$$y = ax + b$$

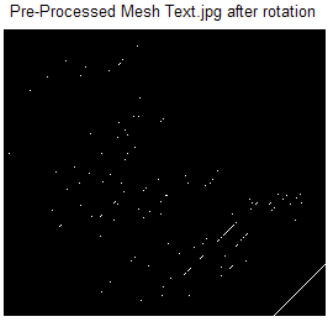
while each point in Hough line space (ρ,θ) describes a line through (x,y) with slope θ and distance ρ from the origin [1]. Lines in Hough space are sinusoids of the form,

$$x \cos \theta + y \sin \theta = \rho.$$

Hough space is separated into angle bins on one axis and radius bins on the other. Each point ρ,θ then is an accumulator counting the number of Hough space sinusoids that pass through it as the above equation is evaluated at each x,y and θ. Ultimately, the largest accumulators correspond to a line that exists in x,y space with slope θ and a distance ρ from the origin. It is easy to see that when applied to lines of text, the angle of skew may be found by looking at the ρ-θ location of the highest accumulator(s) [1][2].

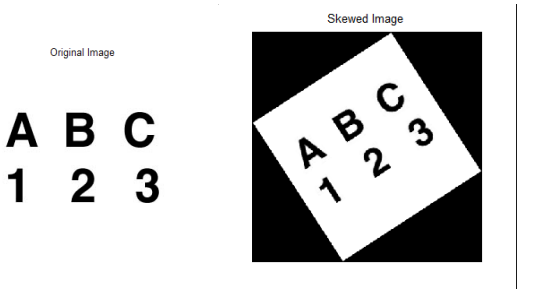
ALGORITHM [2]

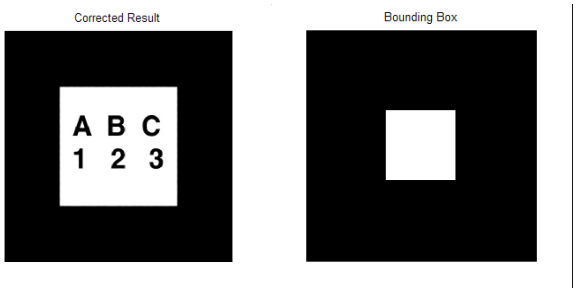
1) Find gradients in image by using a Sobel mask.
2) Threshold the Sobel result to create a binary image.
3) Perform morphological 'closing' and 'thickening' so text is thinned out and resembles many points on a line. The Hough Transform is more accurate, if it operates on clear lines in the document image[1]. A pre processed image is shown below.
4) Split image in half width-wise, in order to get better resolution in angle estimation for each half of the image.
5) Perform Hough Transform on each part of the image. Return accumulator results.
6) Perform some simple statistical analysis on the results of the Hough operation, throwing out any returned angles equal to zero or greater in magnitude than 90^o.
7) The number of times a given angle is returned by the Hough routine is tabulated, and the angle returned the most is the skew estimate.

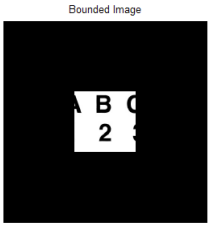


RESULTS

Experimentation was done using skew angles between but not equal to -90 and 90 degrees. For positive angles the algorithm returns perfect results, giving an exact estimate of the skew angle and the skew is corrected by rotation. An example of the algorithm's implementation is shown here for a skew of +33 degrees.

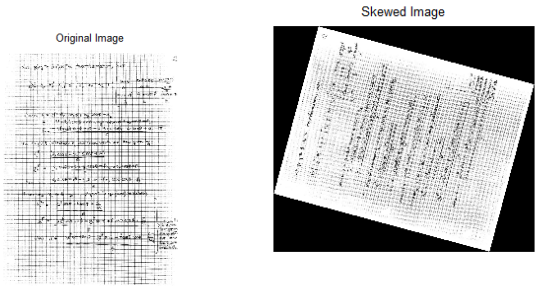


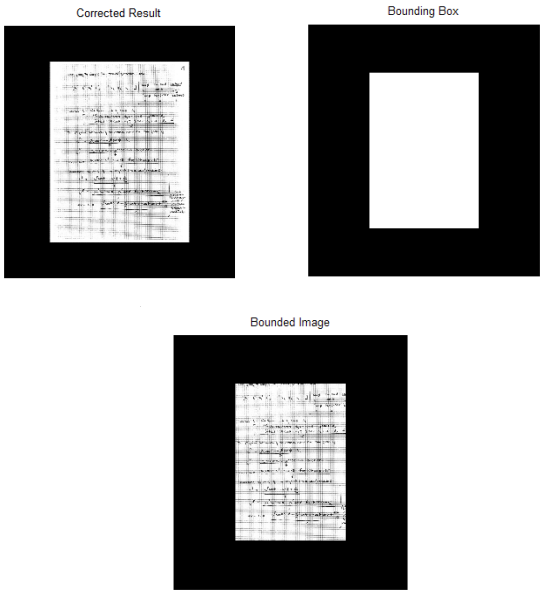




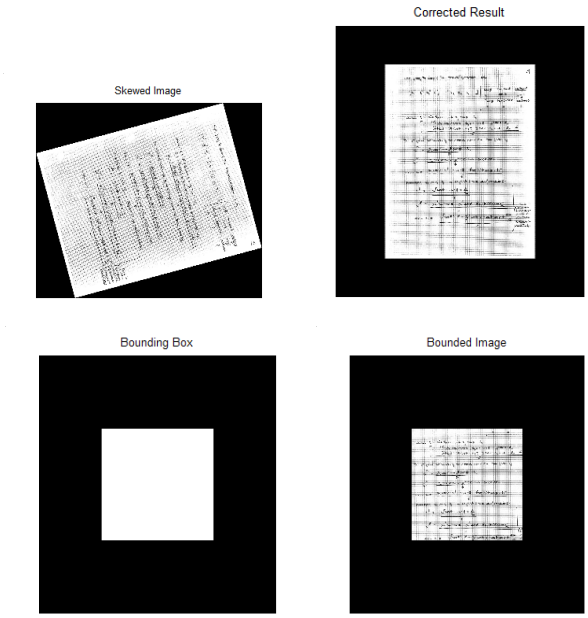
When running the algorithm with a negative skew angle, the returned estimation was consistently off by 90 degrees. For example, starting with a skewed image at -70 degrees returns an estimate of the skew angle at +20 degrees. The workaround for this is to simply correct the image using the incorrect estimate such that the image now lies with a skew of -90 and run the algorithm again. The new estimate is exactly -90 degrees and the image can be now be corrected accordingly. Running the algorithm twice with a positive angle also results in an estimate of -90 degrees so in the interest of keeping it as automatic as possible, the user should enter a guess of -1 if the skew angle is thought to be negative and a guess of +1 if thought to be positive.

Using another image, this one with a lot of text and grid lines, the results are the same. The example below is for a skew angle of 75 degrees.





The negative angle approach is shown implemented here for an angle of -75 degrees and with the help of a user guess of -1.



DISCUSSION

For both sparse text images and dense text image this algorithm works great for positive skew angles. The Hough routine is presented with preprocessed images whose line detail has been enhanced, so the routine returns some very good skew angle estimates and some that aren't so good. It is easy then to automatically throw out those that aren't within the known skew angle range between -90 and 90 degrees. Of those that remain, it is then a simple matter to count up the number of occurrences of each. The angle that occurs the most in that tabulation is the good skew angle estimate.

For negative skew angles, the story is different. It is obvious that the Hough routine is 90 degrees off on its estimates of these, because negative angles have complementary positive angles. A clear example is the dense text image above skewed at -75 degrees. Even though the whole page is skewed at that angle, look at the individual lines and notice that by following them from lower right to upper left, they make +15 degree angles with the vertical axis (positive to the left of the axis). It is these angles that the Hough routine returns as skew estimates. Knowing this allows an automated response to follow, thus ultimately correcting the skewed document.

SUMMARY

As shown before, this algorithm works very well for positive skew angles and can be made to work for negative skew angles with a little help from the user indicating the sign of the skew. With this "guess" from the user being the only interaction, this algorithm lends itself to an almost completely automated implementation for use in document analysis devices.

ACKNOWLEDGMENTS

The free version of Gonzalez's, Eddins' and Woods' DIPUM toolbox obtained from the "Digital Image Processing Using Matlab" website was used for the Hough transform routine and test images were obtained from Prof. Merat's course website.

REFERENCES

[1] Chengming Sun, Deyi Si, "Skew and Slant Correction for Document Images Using Gradient Direction". Pgs 1-2.

[2] R. C. Gonzalez, Richard E. Woods, "Digital Image Processing, "2nd Edition, Prentice Hall, Upper Saddle River, NJ, 2002.

Appendix

MDA_final.m

```
clc;clear;close all;
guess = 1; %-1 if guess of a negative skew, +1 if guess if positive
        skew
f_original = imread('Mesh_Text.jpg');
% f_original = imread('IMAGE5.jpg');
figure,imshow(f_original),title('Original Image');
image = imrotate(f_original,45,'bilinear'); %bilinear interpolation
        gives best looking rotated results
figure,imshow(image),title('Skewed Image');

[theta,table,angle(1)] = rot_correct(image);
final = imrotate(image,-theta,'bilinear'); %if skew angle is nega-
        tive this will produce a result of -90

if guess == -1
[theta,table,angle(2)] = rot_correct(final);  %returns an angle that
        is always off by 90 degrees
   if angle(2) == -90
      final = imrotate(final,90,'bilinear');
      figure, imshow(final),title('Corrected Result')
   end
end
if guess ==1
  figure, imshow(final),title('Corrected Result')
end

%Bounding Box%
%give bounding box in [rows,columns]
final = im2double(final);
box = [700 700];
startbox = zeros(size(final,1),size(final,2));

rows = (size(startbox,1) - box(1))/2 - mod((size(startbox,1) -
        box(1))/2,1);
columns = (size(startbox,2) - box(2))/2 - mod((size(startbox,2) -
        box(2))/2,1);
startbox(rows:rows + box(1)-1,columns:columns + box(2)-1) =
        ones(box(1),box(2));
figure,imshow(startbox);
bounded = startbox .* final;
figure,imshow(bounded),title('Bounded Image')

function [theta,table,angle] = rot_correct(image)

[Fx_y,Im_mag] = skew_sobel(image);


level1 = graythresh(Fx_y);
bw1 = im2bw(Fx_y,level1);
bw2 = morpho(bw1);
rowsplit = size(bw2,1)/2 - mod(size(bw2,1)/2,1);
a = 1;
b=1;
% k = zeros(1:10,1:size(bw1,2));
for z = 1:2
   bw(1:rowsplit,1:size(bw2,2)) = bw2(a:a + rowsplit -
        1,1:size(bw2,2));
   %figure,imshow(bw)
   if isempty(find(bw)) == 0
      phi(b) = hought(bw);
      b = b +1;
   end
   a = a + rowsplit;
end
[theta,table,angle] = stats(phi);


function phi = hought(bw)
[H,theta,rho] = hough(bw);

peaks = houghpeaks(H,500);

for i = 1:size(peaks,2)
   max_h(i) = max(H(peaks(i),1:180)); %peaks returns rows indices of H where p
        occur
end                          %so look in those rows and pull out max value in each r

[y,max_row] = max(max_h);  %find the biggest value in H, where y is biggest val
        %max_row is the row in H containing H's highest
        %we just need the column indice (theta)
for j = 1:180
   if H(peaks(max_row),j) == y %find column indice corresponding to theta
     phi = theta(j);
   end
end

function bwhite = morpho(blackw)
for i = 1:size(blackw,1)
   for j = 1:size(blackw,2)
      if blackw(i,j) == 1
        blackw(i,j) = 0;
      else
        blackw(i,j) = 1;
      end
   end
end

blackw = bwmorph(blackw,'close');
blackw = bwmorph(blackw,'thicken');
%figure,imshow(bw1)
for i = 1:size(blackw,1)
   for j = 1:size(blackw,2)
      if blackw(i,j) == 0
        blackw(i,j) = 1;
      else
        blackw(i,j) = 0;
      end
   end
end
figure,imshow(blackw)
bwhite = blackw;

function [fangle,table,angle] = stats(hangle)
b = 1;
for i = 1:size(hangle,2)
   if abs(hangle(i)) <= 90 & hangle(i) ~=0
      alpha(b) = hangle(i);
      b=b+1;
   end
end

table = tabulate(alpha);
max_angles = max(table(1:size(table,1),2));
for i = 1:size(table,1)
```

```
                                                    fangle = angle;
  if table(i,2) == max_angles;
       angle = table(i,1);
    end
end
```

Document Skew Detection Using the Hough Transformation

Nattapon Chaimanonart
Department of Electrical Engineering and Computer Science,
Case Western Reserve University, Cleveland, Ohio, USA
nxc35@case.edu

Abstract

In this paper, a detection of document skew by using the Hough Transformation is presented. The fundamental Hough Transform calculation and document correction procedure are described and discussed. The algorithm is implemented via MATLAB. The accurate results of transforming several images with different skew angles and different levels of complexity are shown.

Keywords

Document Skew, Hough Transformation, MATLAB

1. INTRODUCTION

Once the documents have been digitized through the scanning system, document skew correction is required before further image analysis. Failure of this compensation can cause serious performance degradation. Several methods of document skew detection have been developed, for example, Projection-based methods, Nearest-Neighbor-based methods, Cross-Correlation-based methods, and Hough-Transform-based methods [1-3].

In this paper, Hough Transformation method is focused. This transformation technique is utilized to detect an unknown skew angle of documents. After the angle is known, the image will be rotated and aligned in the horizontal manner. The primary principal of the Hough Transformation is described in Section 2. The procedures of the document skew correction are proposed in Section 3, which are implemented by using MATLAB. Furthermore, the experimental results from various documents are presented and performance of this algorithm is discussed.

2. HOUGH TRANSFORMATION

In this section, the basic idea of the Hough Transformation is fully explained. This technique transforms the pixel from the x-y coordinate systems into ρ- θ coordinate systems. For example, a point (x_i , y_i) in xy-plane can be transferred to the ρθ-plan by using equation,

$$x \cos \theta + y \sin \theta = \rho \qquad (1)$$

From this equation, x and y are substituted by x_i and y_i. The angle, θ, is varied from -90º to 90º to determine the corresponding, distance, ρ. These sets of data are plotted in ρ-θ plan, which are shown in Figure 1. The same method is applied in the case of (x_j , y_j). With these two points, a straight line can be constructed by connecting point (x_i , y_i) to point (x_j , y_j) in xy-plane. The distance from the origin to the line and θ in xy-plane will correspond to the interception of (x_i , y_i) and (x_j , y_j) plots in the ρθ-plan. By using this technique, the skew angle of the document can be determined.

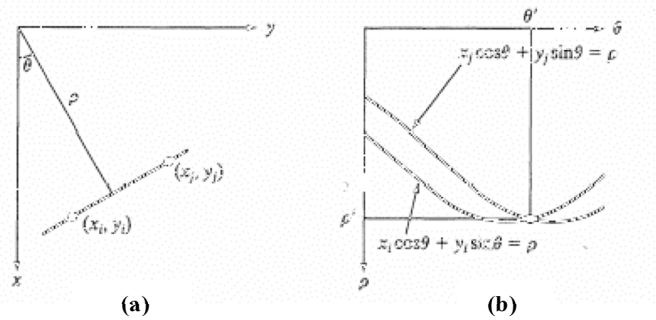


Figure 1. (a) (ρ , θ) parameterization of line in the xy-plane, (b) Sinusoidal curves in the ρθ-plane; the point of intersection, (ρ' , θ'), corresponds to the parameters of the line joining (x_i, y_i) and (x_j, y_j) [4]

For the Hough Transformation, the function written in MATLAB is used to calculate and plot the ρθ-plan. For the actual documents, they might not contain only text. Some documents might have a picture, some of dots, or grids. They are considered as noise in the image that might affect the result. However, it has been noted that an important property of the Hough transform is its insensitivity to the missing part of lines, and to image noise. This can be verified by using the triangle image with some broken lines and a lot of noise as illustrated in Figure 2.

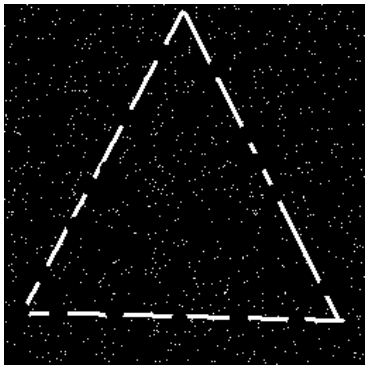


Figure 2. The broken triangle line with noise

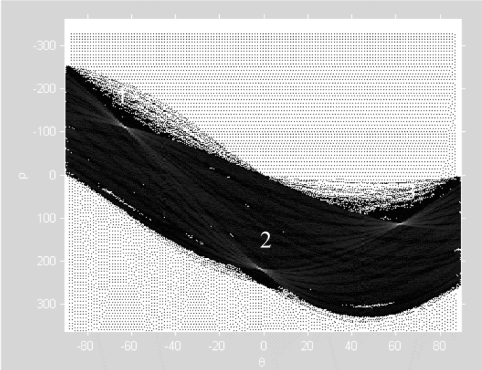


Figure 3. Hough Transform image of Figure 2 by using 'hough' function

The result by using Hough function is demonstrated in Figure 3. From the plot, the white spots in the image 1, 2, and 3 correspond to the three sides of the triangle in Figure 2. With these properties, the Hough Transform is very suitable for detecting the skew angle.

3. PROPOSED PROCEDURE

The correction procedure of document skew is described as follows;

1. Prepare the documents for the transformation. Some of the documents need to be performed color correction. For a grey scale image, thresholding might be needed to transform the document into a binary document. In the case of color documents, they are required to use the intensity information as an input of the Hough Transformation.

2. Perform the Hough Transformation by using 'hough' function to the document [5].

3. Determine the peak pixels from the Hough Transform image. By using the 'houghpeaks' function, the high concentration pixels will be detected. The position of the high pixel concentrations will determine the skew angle. However, finding a meaningful set of distinct peaks in the Hough Transformation is very challenging because lines in the xy-plane usually are not perfectly straight lines. These create multiple peaks in ρ-θ plan. One strategy to overcome this problem is the following [5]:

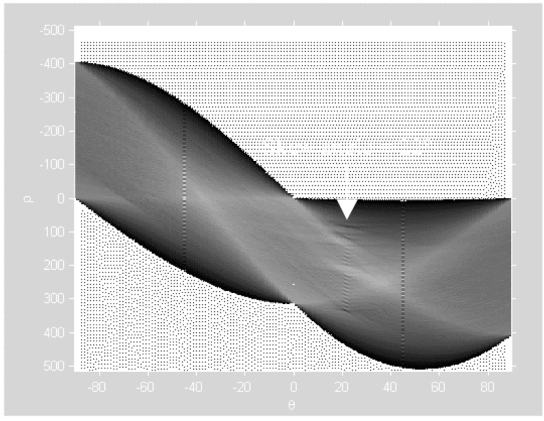
- Find the Hough Transform cell containing the highest value and record its location.
- Suppress (set to zero) Hough Transform cells in the immediate neighborhood of the maximum found in previous step.
- Repeat until the desired number of peaks has been found, or until a specified threshold has been reached.

4. Rotate the image according to the angle determined in step 3 by using 'imrotate' function

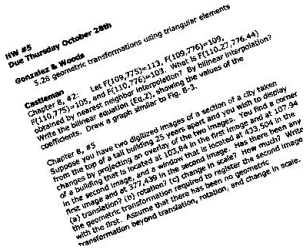
5. Crop the correction image by using 'imcrop' function

4. EXPERIMENTAL RESULTS

In this section, various types of documents are tested by using the proposed procedure. First, the plain text document in Figure 4.b is skewed by an unknown angle. By using the MATLAB function, the Hough Transformation of this document is presented as shown in Figure 4.a. From the plot, the array of the noticeable peaks are aligned in the position that θ = 22°. With this skew angle, the horizontal alignment document can be obtained as illustrated in Figure 4.c.



(a)



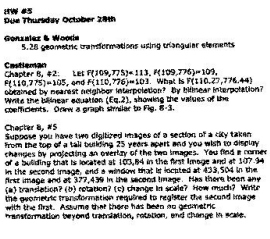
HW #5
Due Thursday October 28th

Gonzalez & Woods
5.26 geometric transformations using triangular elements

Castleman
Chapter 8, #2: Let F(109,775)=113, F(109,776)=109, F(110,775)=105, and F(110,776)=103. What is F(110.27,776.44) obtained by nearest neighbor interpolation? By bilinear interpolation? Write the bilinear equation (Eq.2), showing the values of the coefficients. Drew a graph similar to Fig. 8-3.

Chapter 8, #5
Suppose you have two digitized images of a section of a city taken from the top of a tall building 25 years apart and you wish to display changes by projecting an overlay of the two images. You find a corner of a building that is located at 103,84 in the first image and at 107.94 in the second image, and a window that is located at 433,504 in the first image and at 377,439 in the second image. Has there been any (a) translation? (b) rotation? (c) change in scale? How much? Write the geometric transformation required to register the second image with the first. Assume that there has been no geometric transformation beyond translation, rotation, and change in scale.

(b)



HW #5
Due Thursday October 28th

Gonzalez & Woods
5.26 geometric transformations using triangular elements

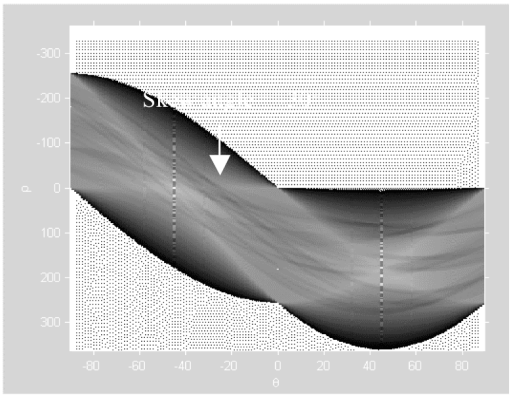
Castleman
Chapter 8, #2: Let F(109,775)=113, F(109,776)=109, F(110,775)=105, and F(110,776)=103. What is F(110.27,776.44) obtained by nearest neighbor interpolation? By bilinear interpolation? Write the bilinear equation (Eq.2), showing the values of the coefficients. Drew a graph similar to Fig. 8-3.

Chapter 8, #5
Suppose you have two digitized images of a section of a city taken from the top of a tall building 25 years apart and you wish to display changes by projecting an overlay of the two images. You find a corner of a building that is located at 103,84 in the first image and at 107.94 in the second image, and a window that is located at 433,504 in the first image and at 377,439 in the second image. Has there been any (a) translation? (b) rotation? (c) change in scale? How much? Write the geometric transformation required to register the second image with the first. Assume that there has been no geometric transformation beyond translation, rotation, and change in scale.

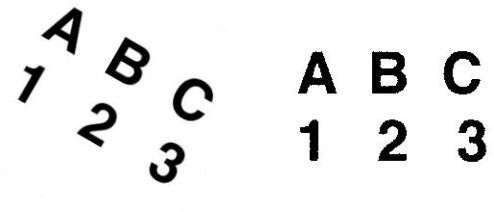
(c)

Figure 4. (a) Hough Transform Image of Figure 4.b (b) The original document (c) The corrected document

Considering a few text document in Figure 5.b, again, the result peaks in the Hough Transformation can be obviously seen at θ=-30º. Moreover, for a document with an image as shown in Figure 6.b, the peaks are presented at θ=-32º and the corrected document is shown in Figure 6.c. From these examples, the peaks in the Hough Transform can be explicitly seen. However, there are some cases that the peaks are hardly determined by observing the Hough Transformation with human eyes, for instance, the document with a grid in Figure 7.b. The solution is to use function 'houghpeaks' to calculate the maximum pixels. As a result from this calculation, the skew angle at 10.5º can be obtained.

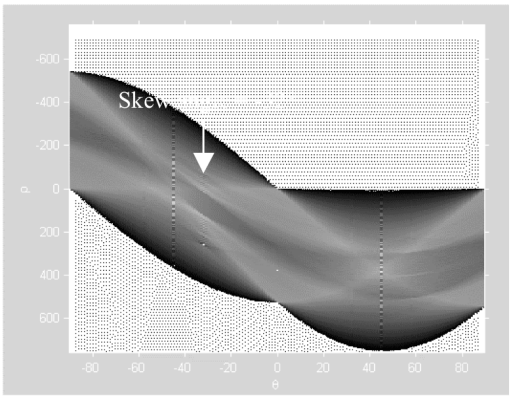


(a)

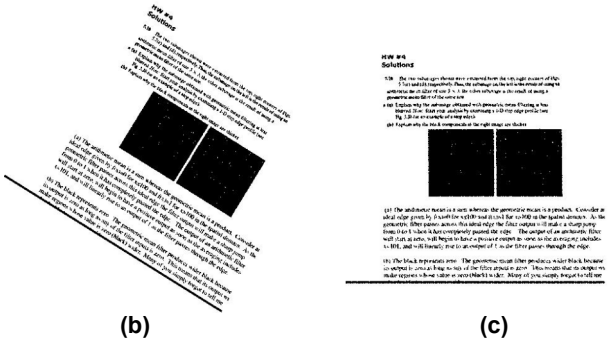


(b) (c)

Figure 5. (a) Hough Transform Image of Figure 5.b (b) The original document (c) The corrected document

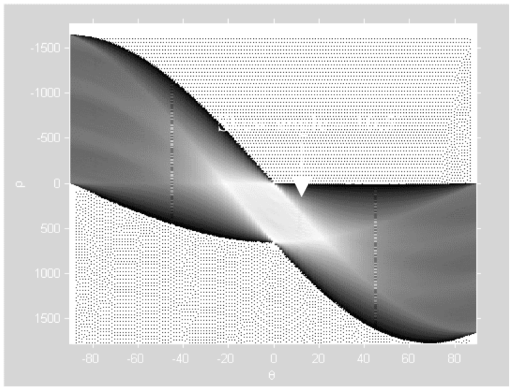


(a)

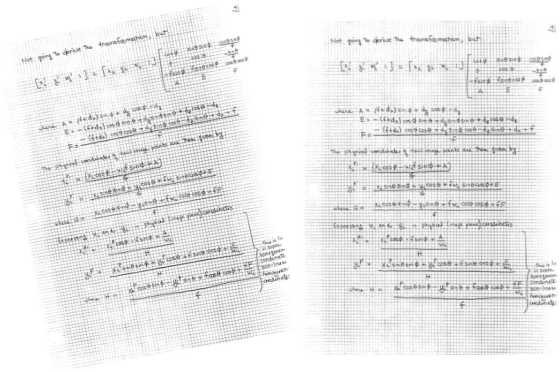


(b) (c)

Figure 6. (a) Hough Transform Image of Figure 6.b (b) The original document (c) The corrected document



(a)



(b) (c)

Figure 7. (a) Hough Transform Image of Figure 7.b (b) The original document (c) The corrected document

5. PERFORMANCE DISCUSSIONS

Although, the Hough Transformation is very suitable for this application, it has some limitations. One of the limitations, which can be found from the experimental results, is that if the documents have a grid, it may be difficult to choose a peak for skew correction. In addition, the Hough Algorithm used in this work is required a lot of computation time. The more delicate skew angle is detected, the more processing time is needed. Therefore, this might not be a proper choice for an application that computation time is restricted.

6. CONCLUSION

In this paper, the Hough Transformation technique is presented to detect the document skew. The fundamental Hough Transformation is briefly explained. The procedure of the skew correction is proposed and implemented with MATLAB. The different types of document, for example, the plain text, the document with some pictures, and the document with a grid, are used to test the program. The results are shown that the Hough Transformation method is very suitable for this application. It is easy to implement.

However, the drawback of this method is that computation time is rather slow.

ACKNOWLEDGMENTS

The Author would like to thank Professor Frank Merat for his support as an instructor, who encourages the students to work on this interesting topic.

REFERENCES

[1] R. Safabakhsh, and S. Khadivi, "Document Skew Detection Using Minimum-Area Bounding Rectangle," *IEEE International Conference Information Technology: Coding and Computing*, pp. 253–258, March 2000.

[2] C. Sun, and D. Si, "Skew and Slant Correction for Document Images Using Gradient Direction," *IEEE International Conference Document Analysis and Recognition*, Vol. 1, pp. 142-146, August 1997.

[3] S. Chen, R. M. Haralick, and I. T. Phillips, "Automatic Text skew Estimation in Document Images," *IEEE International Conference Document Analysis and Recognition*, Vol. 2, pp. 1153-1156, August 1997.

[4] R. C. Gonzalez, and R. E. Woods, *Digital Image Processing*, 2nd Edition, New Jersey, Prentice Hall, 2002.

[5] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing using MATLAB*, 1st Edition, New Jersey, Prentice Hall, 2004.

Skew Correction for Text document Using Fourier Domain Analysis

Chang, Wen-Teng
Department of Electrical Engineering and Computer Science,
Case Western Reserve University, Cleveland, OH, Email: wxc31@cwru.edu

Abstract
Skew estimation and correction is an important step in any document analysis and recognition system. This project presents an algorithm for skew recognition and implements it by Fourier domain analysis. The error estimation and process time are also presented in this project. We suppose the skew angles are within $\pm 45^{o}$ and then analyze the deviation between the corrected one by rotating a known angle.

KEYWORDS
Skew correction, Fourier transform, Frequency domain

INTRODUCTION
Skew estimation and correction is an important step in any document analysis and recognition system. It is quite common when we are digitizing a copy of an original document of text which is skewed by unknown angle. The skewed angle sometimes will severely degrade the performance of the systems.

A wide variety of methods of skew detection algorithms have been proposed in the literature. Basically they can be summarized in five categories [1] (1) project profile, (2) Hough transform technique [2] (3) Fourier method (4) nearest-neighbor clustering and by (5) correction.

Humans can determinate the document orientation without recognizing the text or contents of the document. This idea, however, suggests orientation can be implemented from the global description of the document without looking into the local details of the documents. An algorithm by Fourier transform can exactly extract the predominant direction which is derived from the edge of photo, text or diagram in the document. This project proposes a method based on Fourier domain analysis

OVERVIEW
This project uses 2 images downloaded from EECS490 website, one is simply text and the other is hand written script with vertical and horizontal mesh that are shown as figure 1 and figure 2, individually.

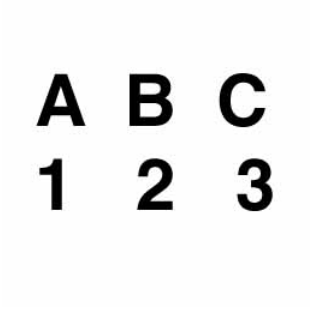


Figure 1 Simple characters

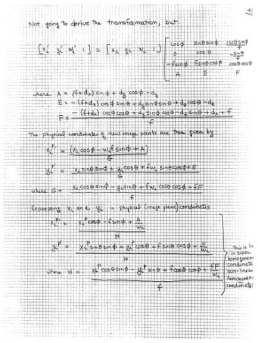


Figure 2 Mesh Text

By rotating an image by a known angle (‘imrotate’), we use Fourier transform to analyze its rotation angle. Here have several methods to find out the angle. One is to find its maximum histogram among the slopes; the other is to find its mean or median value. The rotation angle is converted from slope rate and then got the upright image. Finally, the un-skewed document will need padding its board because the board is surrounded with all black. Figure 3 is the block diagram of this correction process.

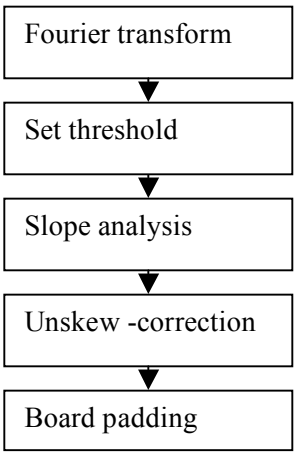


Figure 3 Block diagram of a skewed document correction

ALGORITHM

This section will discuss the process flow of orientating a skew document. All the algorithms used for skew correction are based on estimating the predominant orientation of the text in the spatial domain. When we read an image or a document, the digitized readings combine an M by N matrix with unsigned 1 to 256 gray level values.

1 Fourier Transform

When we transform the images shown upright and skewed one from spatial domain image of figure 2 to frequency domain, the images show upright and skew Fourier spectrum as figure 4 (a)~ 4 (d).

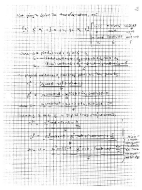


Figure 4 (a)

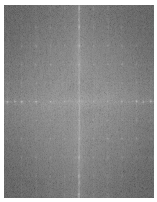


Figure 4(b)

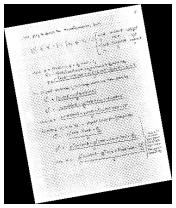


Figure 4(c)

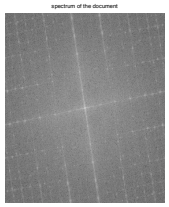


Figure 4(d)

Figure 4 (a) is an upright document whose frequency domain is upright as figure 4(b); figure 4(c) rotates 10 degree and its frequency domain rotates also.

The discrete Fourier transform of 2D image f(x,y) of size MxN is given by[2]

$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-2j\pi\left[\frac{ux}{M}+\frac{vy}{N}\right]} \quad (1)$$

2 Set threshold

In order to utilize the frequency domain to obtain skew angle, we convert it to binary image and set a threshold that is optimized to eliminate bright point that will help shorten calculation time in finding slopes. Median and maximum value are used in this algorithm that can assure the threshold a reasonable value which simplifies to find real rotation angle.

Figure 5 Binary image of the skew document in frequency domain

3 Slope analysis

There are several ways to find out optimal slope of figure 5. One is get a histogram of the slopes that are distributed on figure 6, the slope with the most distributed bright point is desired slope rate.

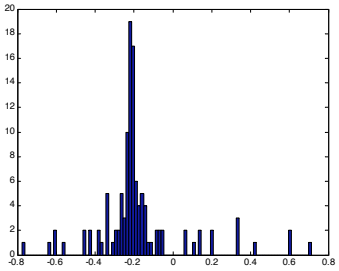


Figure 6 Histogram of slope distribution

Alternatively, mathematic methods can be used to get desired value. Function such as “mean” and “median” can get similar answers. The advantage of using histogram is more accuracy can be acquired by dividing more bins on horizontal axis.

4 Un-skew correction

We can easily convert desired slope rate to angle and rotate this angle in opposite direction. There might be some deviation after correction. The latter part will discuss this question. However, this algorithm will limit skew angle within $\pm 45^{o}$ because we regard document should orientate toward the other direction if angle is over 45^{o}.

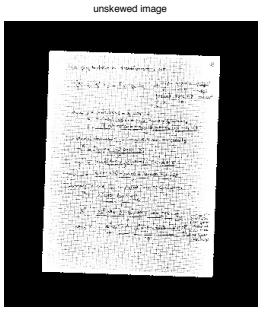


Figure 7 Unskew picture by correcting angle on frequency domain

5 Board padding

To fix the picture on a given box, we need to pad the black board of figure 7. The padding result is shown on figure 8. Note the padding is not completed because the corrected image is not exactly upright.

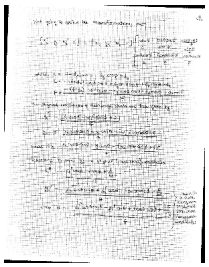


Figure 8 padding result of figure 7

RESULT DISCUSSION

In this section, we will evaluate the gap between corrected angle and true angle from this algorithm. We also calculate the required process time. The process time includes the time of Fourier transform, image presentation of skew and un-skew, histogram and board padding [see appendix]. This analysis uses figure 1, 512x512 pixels gray level image. Table 1 is the summary of the result.

From the result, we can find that the gap between corrected angle and true rotation angle roughly increases along with rotation angle, so the corrected angle may be not satisfying if the skew angle is too large. One way to resolve this problem is to do another Fourier transform. However, it will take approximate double process time according to the result of table 1. Meanwhile, the process time is highly correlated with rotation angle: the higher skew degree requires more time to process the correction and the required time is roughly symmetric to central zero degree.

True angle (^{o})	Detected angle (^{o})	Required time (sec)
-30	-27.474	3.609
-20	-18.434	3.515
-10	-8.972	2.844
-5	-4.398	3.219
-3	-2.726	3.891
-1	-1.273	2.766
1	1.735	2.719
3	3.366	2.843
5	5.527	2.813
10	9.782	2.906
20	18.434	3.266
30	26.474	3.906

Table 1 Test result of (1) corrected angle and (2) process time at different rotation angle

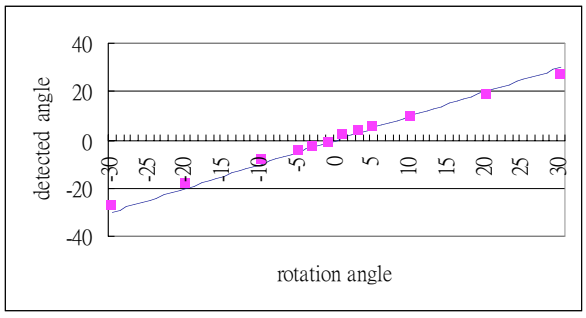


Figure 9 Rotation angle versus detected angle ranged from -30^{o} to 30^{o}. The square points denote the corrected angle; the line indicates true angle.

SUMMARY

The required time by using frequency domain to detect skew angle is highly correlated to skew angle. Besides, the higher skew degree yields roughly larger degree deviation, however, even it cannot obtain a satisfying corrected angle at first correction, the second correction process can reach as good accuracy at least less one degree. For example, if the rotation angle is 30^{o} in this case, we only have to repeat the same procedure so that the accuracy can be further improved to 0.3^{o} deviation at the second time. Nevertheless, comparing to other literature, it may take relatively long

time [4], which may be a disadvantage on the application such as on OCR, which requires fast recognition.

In the result, we find the corrected angle increases roughly along with rotation angle, so there still is limitation for high angle rotation angle. However, Fourier transform can recognize the skew angle without knowing the detailed context; the predominant direction will be extracted form the edge of the text. This is a major feature and advantage of using this algorithm

REFERENCES

[1] "*Skew and Slant Correction for Document Images Using Gradient Direction*" Changming Sun and Deyi Si, in Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on , Volume: 1 , 18-20 Aug. 1997.

[2] *"Analysis of textual images using the Hough transform"*, Machine and Application,pp 141-153, 1989, S,Srihari, and V. Govindaraju

[3] "*Digital Image Processing, 2nd Edition"*. Prentice-Hall, 2001,Rafael C. Gonzalez and Richard E. Woods *electromechanical Systems*, vol. 11, pp. 592-597, 2002.

[4] "*Document Skew Detection Using Minimum-Area Bounding Rectange"* Reza Safabakhsh and Shahram Khadivi, in Information Technology: Coding and Computing, 2000. Proceedings. International Conference on , 27-29 March 2000. Pages:253 - 258

Image De-skewing Using Fourier Transform

Dmitriy Goldman

Department of Electrical Engineering and Computer Science,
Case Western Reserve University, Cleveland, OH, Email: dxg68@cwru.edu

ABSTRACT

When the document is scanned it can be placed on the scanner at arbitrary angle so that it would appear on computer monitor at the same angle. The goal of this project is to detect the angle and correct for it so that the document would appear on computer monitor straight regardless of scanned angle.

KEYWORDS

Fourier, skew, image.

INTRODUCTION

There are number of different methods for image de-skewing available in literature. This method I developed myself and it is based on Fourier transform. The approach is to transform input image from spatial domain to frequency domain and look at direction of frequency distribution. Usually text represents letters arranged in horizontal rows and those rows are stacked one under another. Because of that most of the energy in frequency domain should be along the rows of letters and perpendicular to them. The following experiment confirms this assumption.

IMPLIMENTATION

Fig. 1 shows the input text scanned at some arbitrary angle. In this case the angle is known for test purposes and is equal to 30 degrees. I used the geometric transformation to rotate an input image by 30 degrees. So that if my de-skewing algorithm works correctly I know that the resulting de-skewing angle has to be very close to 30 degrees. Because of rounding errors the resulting angle can't be exactly 30 degrees but if it is close, for example the error is less than 0.1 degrees, then from human point of view it will be completely de-skewed.

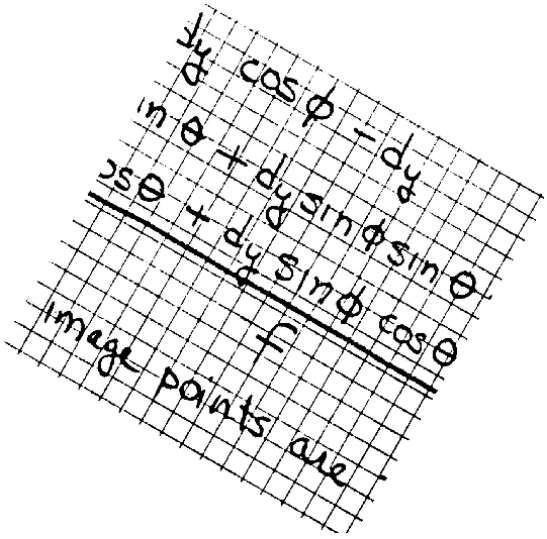


Fig.1: Input Image Scanned at arbitrary angle

Fig. 2 shows the spectrum of the input image and as it was expected most of the energy is distributed along the axis parallel and perpendicular to text lines and greed.

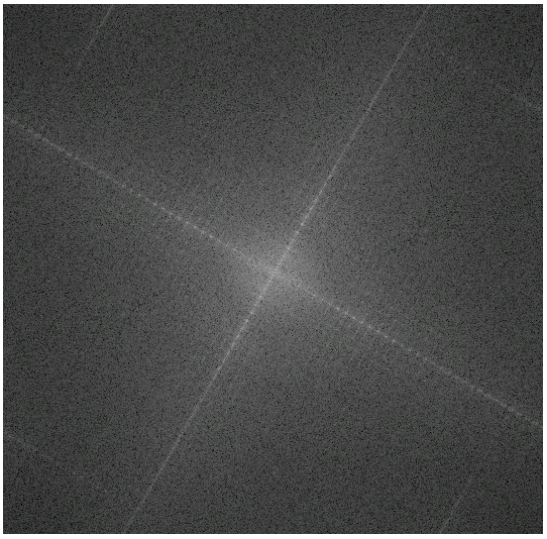


Fig.2: Spectrum of the Input Image

Next as shown on **Fig. 3** the spectrum of the input image was segmented in four quadrants corresponding to quadrants of Cartesian coordinate system.

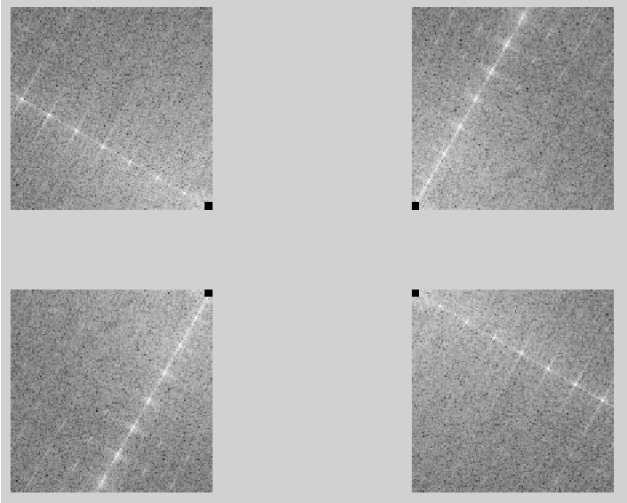


Fig.3: Spectrum of the Input Image Segmented in four Quadrants.

Clearly the brightest points of the spectrum make up the lines that make the same angles with **x** and **y** axis as the original document placed on the scanner. The spectrum image was segmented this way because each quadrant processed by itself should result in right de-skewing angle. And to increase accuracy and minimize possibility for error an average of for resulting errors can be found. Also as it shown in **Fig. 3** there is little black square in the corner of each quadrant. It was done to mask the pixel in the middle of the spectrum image and some pixels around it because the following algorithm is based on finding the brightest points in each quadrant and the middle point is the brightest since it is an average value but this point doesn't help in finding the angle so it needs to be ignored.

The de-skewing algorithm works as follows. *Independently for each quadrant* I find 20 brightest points and their coordinates. Next I fit a straight line through those points and determine the angle between the line and **x** axis. When all four quadrants are evaluated in the same way and four angles (one for each quadrant) are found the **de-skewing** angle is determined as an average of four angles found above.

RESULTS AND DISCUSSION

Fig. 4 shows the de-skewed text after applying the rotational transformation to the input image at determined above angle.

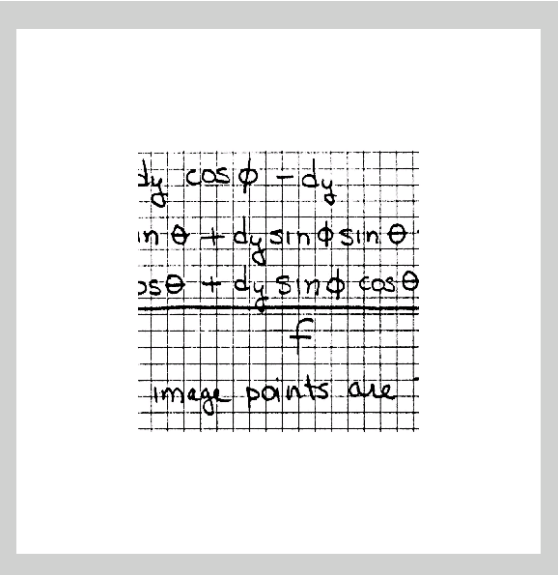


Fig. 4: De-skewed Input Image

The MATLAB code shown in **Appendix 1** after being run resulted in de-skewing angle of 30.0182 degrees. This is a very good result since error is so small that it is absolutely invisible. The image in **Fig. 4** was processed using rotational transformation by the found above angle. And it appears perfectly straight.

Fig. 5 shows the de-skewed image after fitting it into some text boundaries 1000 x 1000 pixels in this example but they can be user selected.

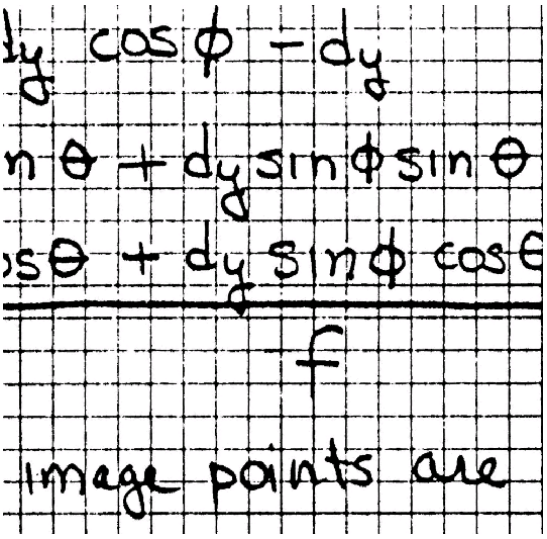


Fig. 5: De-skewed Input Image inside the Text Box

SUMMARY

The described above algorithm definitely does its job. It works with great precision and is easy to implement. The only computationally intensive part is a Fourier transform of the input image. The inverse Fourier transform is not required. Rest of the algorithm is significantly less computationally intensive. Overall this algorithm can successfully be used for text de-skewing.

REFERENCES

[1] Rafael C. Gonzalez, Richard E. Woods, "Digital Image Processing," Second Edition. 2002.

[2] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, "Digital Image Processing Using MATLAB". 2004.

APPENDIX 1

```
clc
clear
f = imread('angle.jpg');
f = im2double(f);
figure(1), imshow(f, [])
title('Fig.1:  Input Image')

F = fft2(f);
Fc = fftshift(F);
S = log(1 + abs(Fc));
figure(2), imshow(S, [])
title('Fig.2:  Spectrum of the Input Image')

[m, n] = size(Fc);
M = floor(m/2)+1; N = floor(n/2)+1;
S1 = S(M-150:M, N:N+150);
[M, N] = size(S1);
S1(M-5:M, 1:5) = 0;  S5 = S1;
%figure(3), imshow(S1, [])
%title('Fig.3:  First Quadrant of the Spectrum of the Input Image')
for k = 1:20
   maximum = max(max(S1));
   [I1(k), J1(k)] = find(S1 == maximum);
   S1(I1(k),J1(k)) = 0;
   I1(k) = M - I1(k);
end
J1 = J1'; I1 = I1';
[row, col] = size(J1);
one = ones(row);
H1 = [J1, one(1:row, 1)];
p1 = H1 \ I1;
angle(1) = atan(p1(1));
angle_degree(1) = angle(1)*180/pi;

[m, n] = size(Fc);
M = floor(m/2)+1; N = floor(n/2)+1;
S2 = S(M-150:M, N-150:N);
[M, N] = size(S2);
S2(M-5:M, N-5:N) = 0;  S6 = S2;
%figure(4), imshow(S2, [])
%title('Fig.4:  Second Quadrant of the Spectrum of the Input Image')
for k = 1:20
   maximum = max(max(S2));
   [I2(k), J2(k)] = find(S2 == maximum);
   S2(I2(k),J2(k)) = 0;
   I2(k) = M - I2(k);
   J2(k) = N - J2(k);
end
J2 = J2'; I2 = I2';
[row, col] = size(J2);
one = ones(row);
H2 = [J2, one(1:row, 1)];
p2 = H2 \ I2;
angle(2) = atan(p2(1));
angle_degree(2) = 90 - angle(2)*180/pi;

[m, n] = size(Fc);
M = floor(m/2)+1; N = floor(n/2)+1;
S3 = S(M:M+150, N-150:N);
[M, N] = size(S3);
S3(1:5, N-5:N) = 0;  S7 = S3;
%figure(5), imshow(S3, [])
%title('Fig.5:  Third Quadrant of the Spectrum of the Input Image')
for k = 1:20
   maximum = max(max(S3));
   [I3(k), J3(k)] = find(S3 == maximum);
   S3(I3(k),J3(k)) = 0;
   I3(k) = M - I3(k);
   J3(k) = N - J3(k);
end
J3 = J3'; I3 = I3';
[row, col] = size(J3);
one = ones(row);
H3 = [J3, one(1:row, 1)];
p3 = H3 \ I3;
angle(3) = atan(p3(1));
```

```
angle_degree(3) = -angle(3)*180/pi;



[m, n] = size(Fc);
M = floor(m/2)+1; N = floor(n/2)+1;
S4 = S(M:M+150, N:N+150);
[M, N] = size(S4);
S4(1:5, 1:5) = 0;  S8 = S4;
%figure(6), imshow(S4, [])
%title('Fig.6:  Fourth Quadrant of the Spectrum of the Input Image')
for k = 1:20
   maximum = max(max(S4));
   [I4(k), J4(k)] = find(S4 == maximum);
   S4(I4(k),J4(k)) = 0;
   I4(k) = M - I4(k);
   J4(k) = N - J4(k);
end
J4 = J4'; I4 = I4';
[row, col] = size(J4);
one = ones(row);
H4 = [J4, one(1:row, 1)];
p4 = H4 \ I4;
angle(4) = atan(p4(1));
angle_degree(4) = 90 - angle(4)*180/pi;

angle_degree = sum(angle_degree)/4
angle = angle_degree * pi / 180;

figure(3),
subplot(2,2,1), imshow(S5, [])
%title('Fig.3:  First Quadrant')
subplot(2,2,2), imshow(S6, [])
%title('Fig.4:  Second Quadrant')
subplot(2,2,3), imshow(S7, [])
%title('Fig.5:  Third Quadrant')
subplot(2,2,4), imshow(S8, [])
%title('Fig.6:  Fourth Quadrant')

if angle_degree >= 45
   th = -pi/2 + angle;
else
   th = angle;
end
T = [cos(th) sin(th) 0;...
   -sin(th) cos(th) 0;...
    0        0     1];
tform = maketform('affine', T);
g = imtransform(f, tform, 'FillValue', 1.0);
figure(4), imshow(g, [])
title('Fig.7:  De-skewed Image')
[M,N] = size(g);
out = g(floor((M-1000)/2):M-floor((M-1000)/2),...
   floor((N-1000)/2):N-floor((N-1000)/2));
figure(5), imshow(out, [])
title('Fig.8:  De-skewed Image Fitted into Bounding Box')
```

Image Skew Detection and Correction using Morphology

Isaac Hirt, Frank Merat
Department of Electrical Engineering and Computer Science,
Case Western Reserve University, Cleveland, OH, Email: ijh1@cwru.edu

Abstract
This paper presents an algorithm, which is used to detect the skew in scanned documents. This algorithm uses morphology to morph the text into rectangles, then find a line through the base of the rectangles which closely resemble a line of text. The line detection process is run for the entire image, next the average slope is used to detect the estimated skew of original text document. The original image is then skewed by the detected skew in the opposite direction to attempt to remove the skew from the image. The skew detection portion of the algorithm takes approximately 1.8 seconds on a 2.4 Gigahertz Intel Celeron laptop with 256 Megabytes of RAM.

KEYWORDS
Morph, skew, connect, bounding box, traverse

INTRODUCTION
As computing power increases, it is desired to have more data stored electronically. Many data processing algorithms require the data to have a minimal skew for processing [1]. To convert documents stored on paper to an electronic format they are scanned, when modern scanners equipped with automatic feeders are utilized, the skew is limited to a range of +/- 3 degrees [2].

There are five main methods for detecting the skew of a document:

1: Projection – Creating a projection profile for the document which will show more variation when the profile is in the direction of the skew, and thus the skew angle is found.

2: Hough-Transform – The Hough Transform is used on the document image, the transform is then analyzed to find the skew of the document.

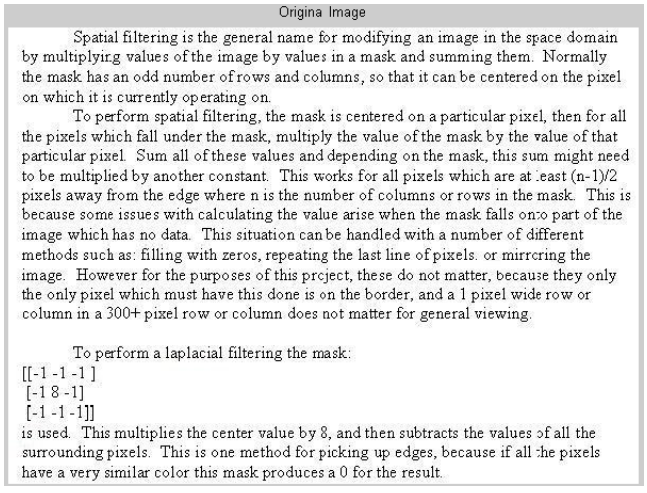
3: Nearest-Neighbor Method – The nearest neighbor to each connected segment in the document is found for all segments, the angle between these neighbors is calculated and a histogram of these angles is analyzed to determine the skew of the document.

4: Cross-Correlation Methods – Calculates the documents skew by finding the vertical shift required to maximize the cross-correlation between horizontal elements.

5: Various other Methods – Such as the Fourier Transform, Morphological operators and local region complexity. [3]

This paper examines an algorithm based on method number five, using morphological operators to determine the skew. There are six steps to this algorithm along with one preprocessing. The preprocessing step for testing the algorithm is to induce skew into a test image. The first step of the algorithm is to perform the morphological opening on the document; the second step follows then performs a closing operation on the document. The third step attempts to find the base lines of the rectangular results from the morphological operations. The fourth step is to use these lines to determine the skew of the image. The fifth step is to create a bounding box for the valid regions of text. The final step is performing the skew to align the document for processing.

ALGORITHM - PREPROSSING
The preprocessing portion of the algorithm involves either randomly determining an angle to skew which to skew the document image or using a skew angle provided by the user. To perform the skew calculations a function provided by MathWorks, the creator of Matlab was used, which is called rotate_image [4]. This function performs rotates a given image through a specified number of degrees, and is also able to specify where specific points are located after the rotation. This function will rotate the text document shown in Figure 1 by 5 degrees; the rotation is shown in Figure 2. The primary thing to notice is that this algorithm needs to add padding around the image to perform the rotation due to the width and heights being enlarged in rotation.


Origina Image

Spatial filtering is the general name for modifying an image in the space domain by multiplying values of the image by values in a mask and summing them. Normally the mask has an odd number of rows and columns, so that it can be centered on the pixel on which it is currently operating on.

To perform spatial filtering, the mask is centered on a particular pixel, then for all the pixels which fall under the mask, multiply the value of the mask by the value of that particular pixel. Sum all of these values and depending on the mask, this sum might need to be multiplied by another constant. This works for all pixels which are at least (n-1)/2 pixels away from the edge where n is the number of columns or rows in the mask. This is because some issues with calculating the value arise when the mask falls onto part of the image which has no data. This situation can be handled with a number of different methods such as: filling with zeros, repeating the last line of pixels. or mirroring the image. However for the purposes of this project, these do not matter, because they only the only pixel which must have this done is on the border, and a 1 pixel wide row or column in a 300+ pixel row or column does not matter for general viewing.

To perform a laplacial filtering the mask:
[[-1 -1 -1]
[-1 8 -1]
[-1 -1 -1]]
is used. This multiplies the center value by 8, and then subtracts the values of all the surrounding pixels. This is one method for picking up edges, because if all the pixels have a very similar color this mask produces a 0 for the result.

Figure 1: Original image to be used as a processing example

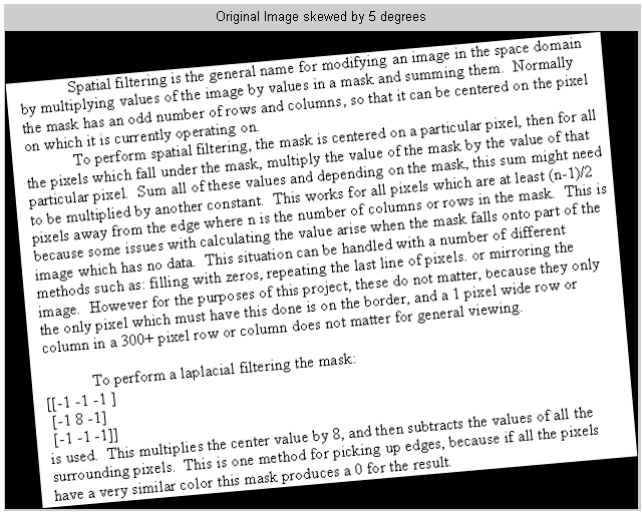


Figure 2: Original image skewed by 5 degrees.

ALGORITHM – STEP ONE

The first step of the algorithm is to perform the morphological opening operation upon the skewed image. This closes attempts to close the gaps internal to characters, between characters and between different words. This process is only applied once to the image due to a large structuring element being used, a 6 by 8 array of ones, for processing the image. The results of performing the opening operation on Figure 2 are shown in Figure 3.

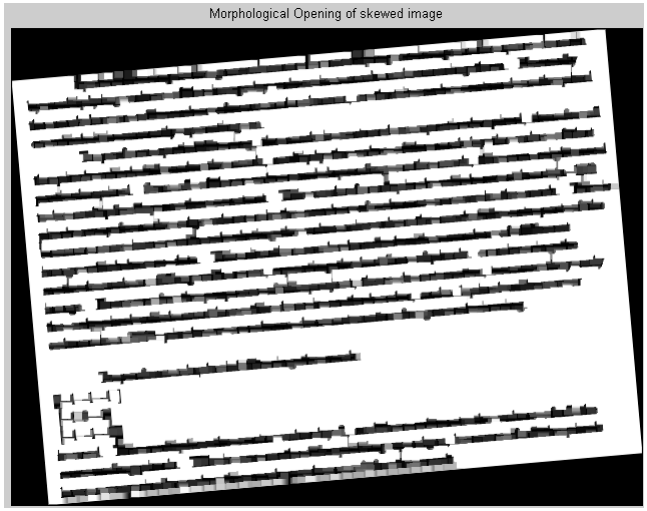


Figure 3: Performing the opening operation upon Figure 2.

ALGORITHM – STEP TWO

Next the morphological closing operation is performed upon the image; this is shown in figure 4. The goal of the closing the image is to remove the ascenders and descanters of the characters for easier base line detection in step 3.

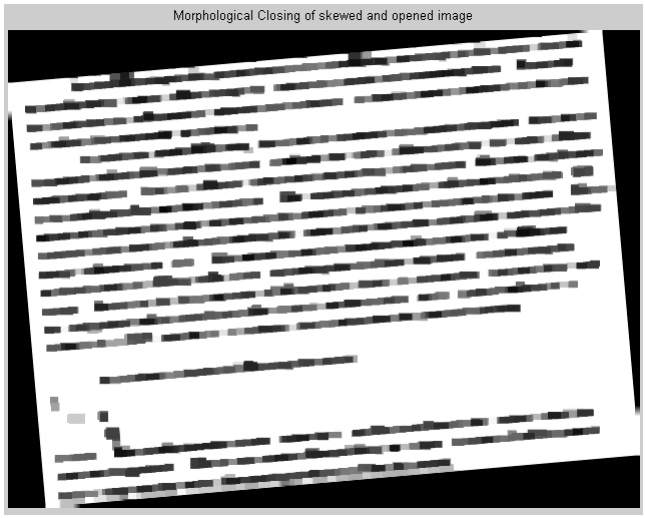


Figure 4: Performing the closing operation upon Figure 3.

The final part to step 2 is threshold the result so that the resultant image is a binary image.

ALGORITHM – STEP THREE

After the image has been opened, closed and thresholded, it should look similar to Figure 5.

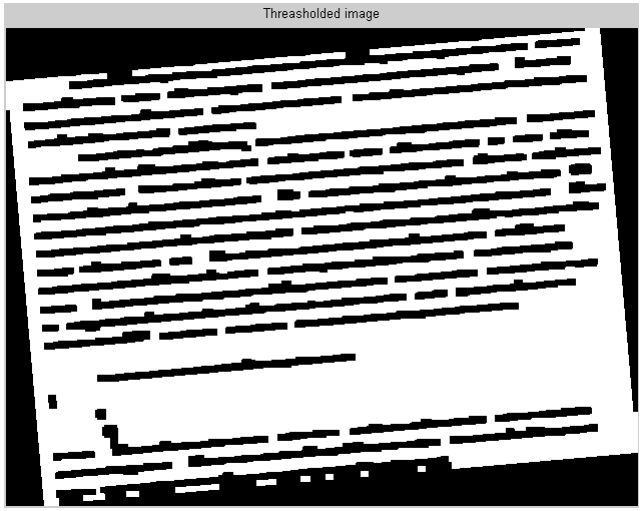


Figure 5: Thresholded image after morphological processing

The algorithm now attempts to detect the bottom of each rectangular section, and connect rectangles which should end up on the same level after de-skewing.

To accomplish this, the image is traversed by along the top of the image, one quarter of the distance to the upper right corner. Then the scanning down the image for the first pixel which is white, once this white pixel is found, it means that the start of the document has been found, so the algorithm starts to look for a black pixel in the column. Once a black pixel is found, the algorithm descends to the

lowest pixel on the column before finding a white pixel; next it follows the bottom of the rectangle until it reaches a gap between rectangles. If there is a large enough gap between the current location and the rightmost border, the algorithm attempts to span the gap of white pixels. There are a maximum number of white pixels which the algorithm is able to process. The number of white pixels must be low enough to insure that if another rectangle is found, it has originated on the line which is being processed.

Once an the algorithm has reached the end of a connected set of black rectangles, a line is fitted from the starting location to the ending location, which ends up being the bottom of both sides of the rectangle. This line than has its slope calculated for determination of the skew further in the algorithm.

Next the line connection portion of the algorithm continues to descent from its starting column finding more horizontal lines across the image. The results of this processing are shown in Figure 6.

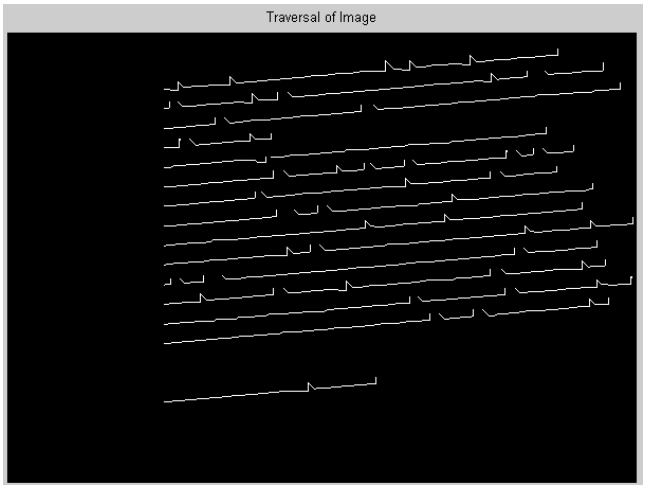


Figure 6: Horizontal line detection

One thing to note in Figure 6 is the saw tooth location on most lines. These locations are where the algorithm attempted to find more black pixels before crossing a gap between rectangles. Also each line segment has a vertical bar at the right side; this represents the line detection's attempt to continue finding more black pixels at the bottom right corner of the line.

ALGORITHM – STEP FOUR

The skew of the image can now be approximated using the slopes of the detected lines. First a mean and standard deviation of the collection of slopes is calculated. Next the standard deviation is used to eliminate slopes which are too far away from the standard deviation; this is an attempt to remove bad data. Next an average of the remaining good slopes is calculated. Slope is the result of tan(φ), therefore the skew can be found by using the arctangent of the slope.

ALGORITHM – STEP FIVE

Now the bounding boxes need to be calculated. To calculate the bounding boxes, the morphologically processed image is used. Since that already has black boxes which roughly surround each word, this will be used. However the image needs to be eroded using a new structuring element, which in this cast is a 6 by 6 array of ones, to enlarge the black rectangles to include the ascenders and descanters of the characters, the results of this are shown in Figure 7.

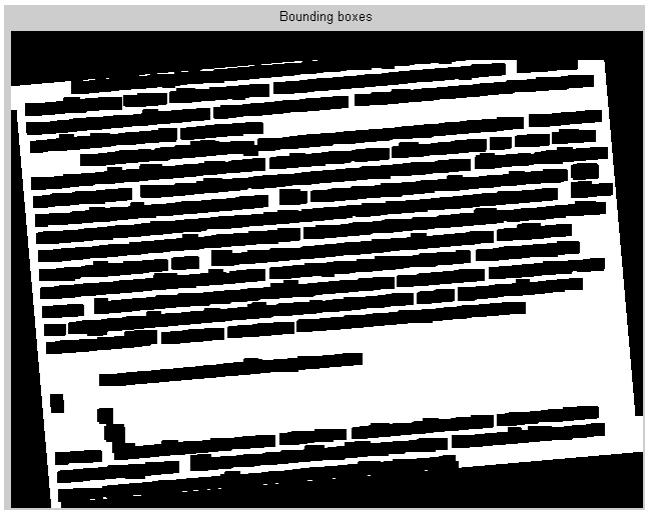


Figure 7: Bounding box mask

Now the logical 'or' operation is performed with the mask, and the original skewed image. This will keep only the areas where the mask is black, while attempting to keep all the text regions their original value, the results of this are shown in figure 8.

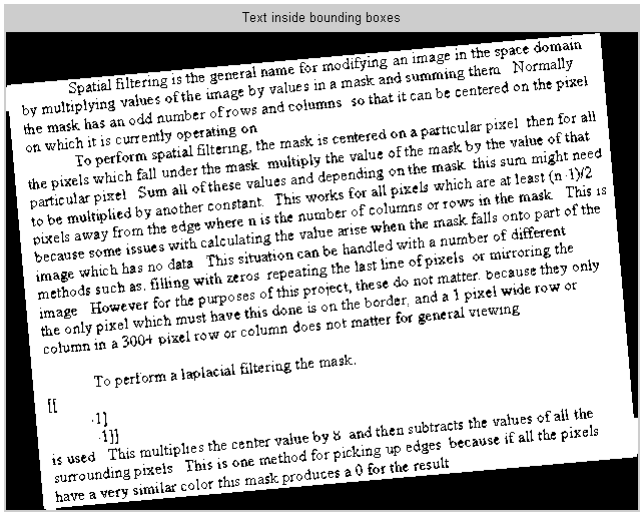


Figure 8: Skewed image after bounding regions are used

Now only the areas of the text which were large enough to survive the opening and closing of the image will be deskewed.

ALGORITHM – STEP SIX

The last step of the algorithm is to de-skew the image. This is done using the same function which skewed the image. The angle the image is de-skewed by is the negative of the value calculated in step 4, and the image which is de-skewed is from step 5, the results of the de-skewing are shown in Figure 9.

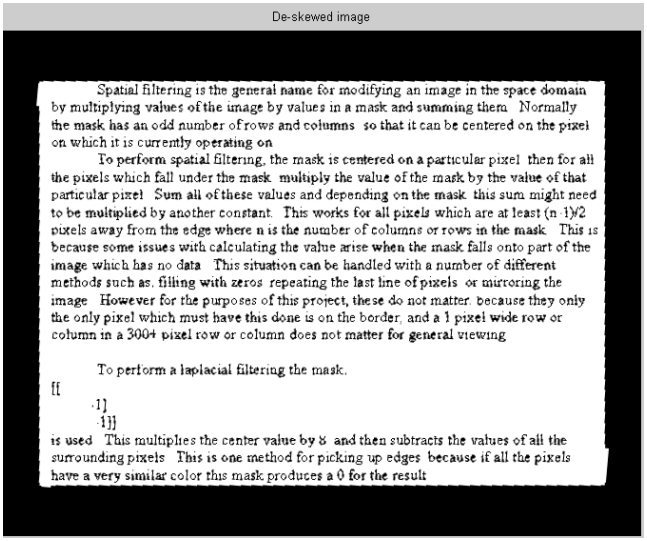


Figure 9: De-skewed Image

Now the final part of the algorithm is to remove the black border which is added from the image rotation function, this is shown in Figure 10.

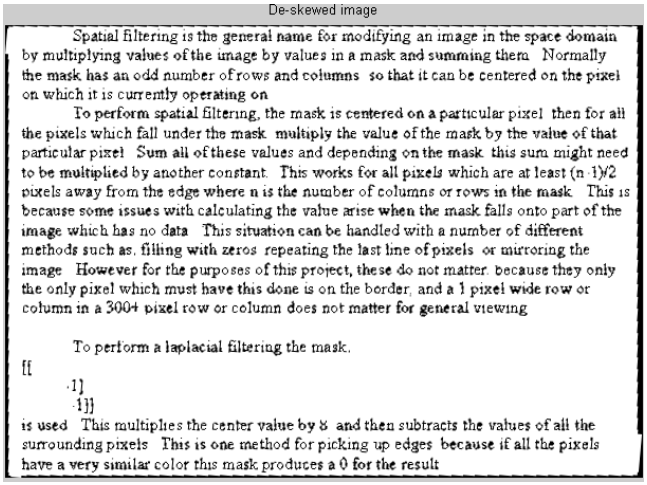


Figure 10: Final results of the de-skewing algorithm

It should be noted that there is some warping of the image due in part to the image being rotated two times, this ends up blurring the image due to data getting averaged in the rotations. This is also the cause of the upper left and lower right corners having the extra bulge of white.

RESULTS

This algorithm was tested on a set 4 of images taken from documents submitted for EECS 490 at Case Western Reserve University a variety projects. Each of these 5 images was subjected to 10 different random skews, φ, and the de-skewing algorithm processed these images. The results are shown in Tables 1-4.

Table 1: First Sample:

		Sample 1	
Trial	Act φ	Calc. φ	Δ φ
1	-3.2373	-3.1227	0.1146
2	-0.9429	-0.9516	-0.0087
3	4.3547	4.3507	-0.004
4	4.169	4.2724	0.1034
5	-0.8973	-0.8977	-0.0004
6	3.9365	3.9336	-0.0029
7	-4.4211	-4.2647	0.1564
8	-1.4713	-1.3943	0.077
9	3.1317	3.1068	-0.0249
10	-4.9014	-4.7405	0.1609
		Mean φ	0.06532
		Max φ	0.1609

Table 2: Second Sample:

		Sample 1	
Trial	Act φ	Calc. φ	Δ φ
1	-3.6111	-3.3798	0.2313
2	-2.9723	-2.8094	0.1629
3	-3.0128	-2.8734	0.1394
4	1.0379	1.0246	-0.0133
5	-2.2781	-2.1744	0.1037
6	-3.0119	-2.8734	0.1385
7	-4.8473	-4.4898	0.3575
8	2.4679	2.508	0.0401
9	-0.549	-0.4791	0.0699
10	4.3181	4.341	0.0229
		Mean φ	0.12795
		Max φ	0.3575

Table 3: Third Sample

		Sample 1	
Trial	Act φ	Calc. φ	Δ φ
1	3.1797	3.1201	-0.0596

2	1.6023	1.4944	-0.1079
3	-1.5803	-1.1489	0.4314
4	-2.1027	-1.9402	0.1625
5	-1.5881	-1.1496	0.4385
6	0.3408	0.2261	-0.1147
7	2.2711	2.3805	0.1094
8	-1.9071	-1.7873	0.1198
9	3.385	3.3675	-0.0175
10	0.6807	1.3182	0.6375
		Mean φ	0.21988
		Max φ	0.6375

Table 4: Fourth Sample

		Sample 1	
Trial	Act φ	Calc. φ	Δ φ
1	-1.2959	-1.2932	0.0027
2	2.0274	2.1311	0.1037
3	0.4657	0.6307	0.165
4	-0.5512	-0.6408	-0.0896
5	1.9457	1.8181	-0.1276
6	1.2131	1.246	0.0329
7	2.9482	2.8955	-0.0527
8	4.5684	4.5261	-0.0423
9	0.2259	0.4549	0.229
10	3.8014	3.7725	-0.0289
		Mean φ	0.08744
		Max φ	0.229

This data yields an average error of 0.12 degrees with a maximum error of 0.637. These results are better than the similar algorithm by Su Chen and Robert Harakick, however their algorithm was created 9 years ago. Another point of note, is that although the range of +/- 5 degrees was tested, it is unlikely that any skew of more than 3 degrees would be encountered.

The processing time for the average set of data was 1.5 seconds on a laptop with a 2.4 Gigahertz Intel Celeron, with an average rotation time of 1.2 seconds on the same laptop.

SUMMARY

The algorithm developed for this paper does a good job detected the actual skew of the image, usually within 0.2 degrees of the actual skew. However the bounding boxes which are calculated are not as accurate as possible because the closing operation attempts to remove the ascenders and descanters on the letters for calculation of the base line, where the bounding box function is attempting to add these back to the image. Also the opening operation removes the small detail such as most of the values of the array shown in figure 1, so this does not receive a bounding box and is not in the final image.

FUTURE WORK

The algorithm does an accurate job of detecting the skew of an image. However the bounding box calculations could be improved by using more specialized morphological operations on the image specifically to determine the valid regions of text.

ACKNOWLEDGMENTS

This work was done for Dr. Frank Merat's EECS 490 'computer Vision' class.

REFERENCES

[1] Su Chen and Robert Haralick. "Automatic Text Skew Estimation in Document Images". 1995., Proceedings of the Third International Conference on , Volume: 2 , 14-16 Aug. 1995. Pages:1153 - 1156 vol.2

[2] Abhishek Gattani, Maitrayee Mukerki and Hareish Gur. "A Fast Multifunctional Approach for Document Image Analysis". 2003. Proceedings. Seventh International Conference on , 3-6 Aug. 2003.

[3] Reza Safabakhsh and Shahram Khadivi. "Document Skew Detection Using Minimum-Area Bounding Rectangle". Proceedings. International Conference on , 27-29 March 2000.

[4] Ohad Gal. rorate_image.m. http://www.mathworks.com/matlabcentral/fileexchange/loadCategory.do, 2003-10-21, written for Rev 12.1

APPENDIX A – SPECIAL MATLABFUNCTIONS:

Deskew: implements the algorithm discussed in the paper
Threashold_image: quickly thresholds a desired image based on an input parameter.
Rotate_image: Rotates an image through a specified number of degrees counterclockwise
Trim_image: Removes as much of the black border added to the image from rotate_image as possible.

'Detection and Correction of Skew in Document Images'

Ruchi Kothari

Department of Electrical Engineering and Computer Science,

Case Western Reserve University, Cleveland, OH, Email : rxk89@cwru.edu

Abstract

This paper presents a fast and efficient algorithm for the estimation of skew in document images and subsequent correction to de skew the text. Skew angle detection is an important step in Optical character recognition (OCR) [4]. The method employed is fast and simple and can detect skew angles in the range of -80 degrees to 80 degrees for text with text lines as dominant feature. The algorithm is based on Fourier domain operations and is easy to implement.

KEYWORDS

OCR, Skew detection/correction, spectral texture

INTRODUCTION

Document image processing is required to improve the quality of document text. Document skew detection is an important pre-processing step in document image processing. Skew is inherent in scanning operations and small skew angles are routinely encountered [5]. Skew detection is an important part of document image processing applications as uncompensated skew can deteriorate document features and can complicate further document image processing steps. A great deal of effort has been put in this area and various algorithms have been developed to eliminate document skew. These algorithms can be compared based on performance criteria. Most important performance criteria are accuracy of skew angle detection, range of skew angle detection/correction, speed of processing the image and computational complexity involved. Basically, the methods developed for skew correction can be categorized into following main types: 1). Hough transform based technique: Skew of document text lines is detected based on Hough transform. Hough transform basically transforms the (x, y) space to the (ρ, Θ) space of lines where ρ signifies distance from the origin and Θ is the slope of the line [3]. The peaks found correspond to the most prominent lines and hence skew can be determined by finding the angles of the most prominent lines in the document text. This method is reliable as it always computes the angle of the most dominant lines in the document image. These methods work well with noise, gaps and break in characters [3]. These methods are limited by inability in detection of linewidths. Also, in presence of very few text characters, sometimes line detection may not be as accurate as desired. HT methods are not particularly suited in dealing with large size images. 2). Projection profile based methods: The projection profile obtained corresponding to most variations correspond to the skew angle [5]. This process is performed for different angles and the largest magnitude variations correspond to the skew angle. 3). Cross-correlation based techniques: These determine vertical shift needed to maximize correlation between pairs of vertical document columns.

4). Nearest neighbor clustering: This bottom-up method is based on calculating histogram angles of connected components and their nearest neighbors [3].

5). Fourier techniques for skew correction: Fourier methods are very popular and more intuitive than other methods. The Fourier based method is based on finding the largest density in the Fourier space. Usually, it is assumed that the most density lines in the Fourier space will be in the vertical direction (corresponding to the horizontal lines) but this might not be the case always depending on the text.

6). Character slant angle detection based methods: Most methods are based on detection of the text-line direction. Few methods are based on detecting the slant of characters. These methods attempt to normalize the character slant to the vertical. Methods based on slant of characters are less trustworthy. Character based methods need to be more complex to achieve functionality in a myriad of situations.

SPECTRAL TEXTURE BASED METHOD [1]:

We employ a skew correction method that is based on finding spectral texture of images which in turn is based on 2-D Fourier methods [1]. Fourier transform can detect periodic and non-periodic patterns in an image easily by examining their directionality. These spectral texture patterns can be used to detect periodic and non-periodic patterns and can also quantify the differences between the non-periodic patterns. Global texture patterns which can be distinguished from each other as high-energy bursts are difficult to detect with spatial methods. For easier interpretation of the spectral patterns, the results are displayed in the polar-coordinate space. This is ideally suited for detecting skew angles. The spectrum function S obtained in the polar coordinate space wherein r and Θ are the coordinate system variables. For each direction Θ, $S_{\square}$ (r) is a 1-D function that can be analyzed to yield the behavior of the spectrum on a radial direction from the origin for a fixed value of Θ.

Similarly, for each frequency $S_r(\Theta)$ is a 1-D.

$S(r) = \sum S_{\square}(r)$ on $\Theta = 0$ to π

$S(\Theta) = \sum S_r(\Theta)$ on r=1 to R0

Analyzing $S_r(\Theta)$ yields the behavior on a circle centered on the origin for a fixed value of r.

Ro is the circle-radius centered at the origin.

For each pair of coordinates (r, Θ), a pair of values namely [S(r), S(Θ)] is found. We can thus compute 2-D functions S(r) and S (Θ) that contain the spectral energy information for the whole image by calculating S(r) and S (Θ) for all values of r and Θ.

Here, we have exploited the following features of the Fourier Spectrum [1]:

1). Dominant peaks in the spectrum correspond to the prominent direction of texture in the spectrum

2). Peaks in the frequency plane correspond to fundamental spatial period of patterns

As the Fourier spectrum is symmetric, only half of the frequency plane needs to be considered. Thus, we can easily associate each pattern in the spectrum with one prominent peak in the Fourier spectrum rather than two.

The program based on the above spectral approach is easy to implement and the results obtained are straightforward.

RESULTS AND DISCUSSION

We obtain very good results from the implementation of our program based on the above described spectral texture technique.

Skew angle detection and correction was reliably performed for correcting skew angles in the range of -80 degrees to 80 degrees. This range is sufficient to be applicable in skew correction of document texts routinely encountered.

The program was tested for different images and different angles. Results obtained were satisfactory for text that contained more than two text lines in the document. When the text document contains sparse text or very few characters, the dominant lines in the Fourier space do not correspond to the angle of skew of the text lines. These can correspond to the prominent lines in the characters (usually vertical slant) and for such sparse text, slant detection based correction methodologies should be employed.

But in practical situations, the document text encountered has more than two lines of text, so this is not a serious limitation.

Below, we show the results of performing skew detection/correction for text-documents using our program.

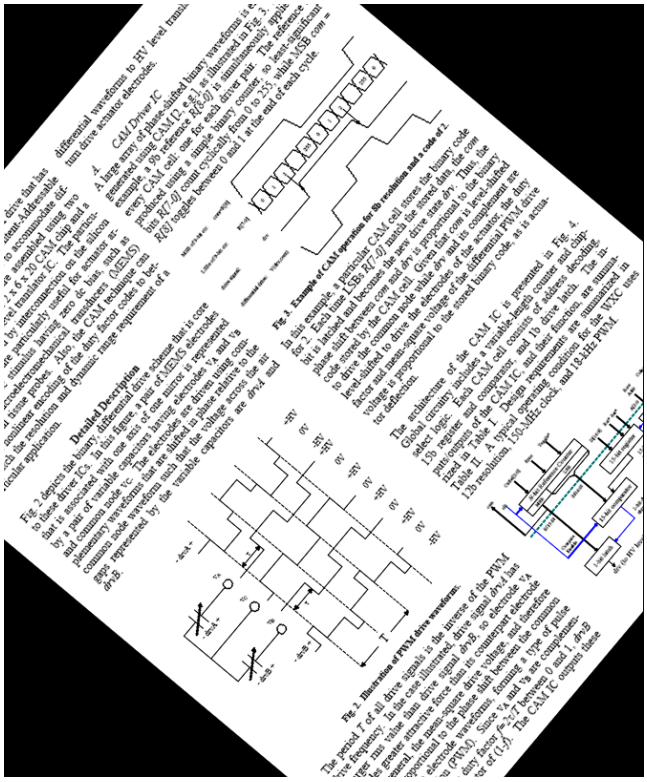


Figure 1. Skewed image with few figures. Skew angle is (positive (counter-clockwise) 50 degree)

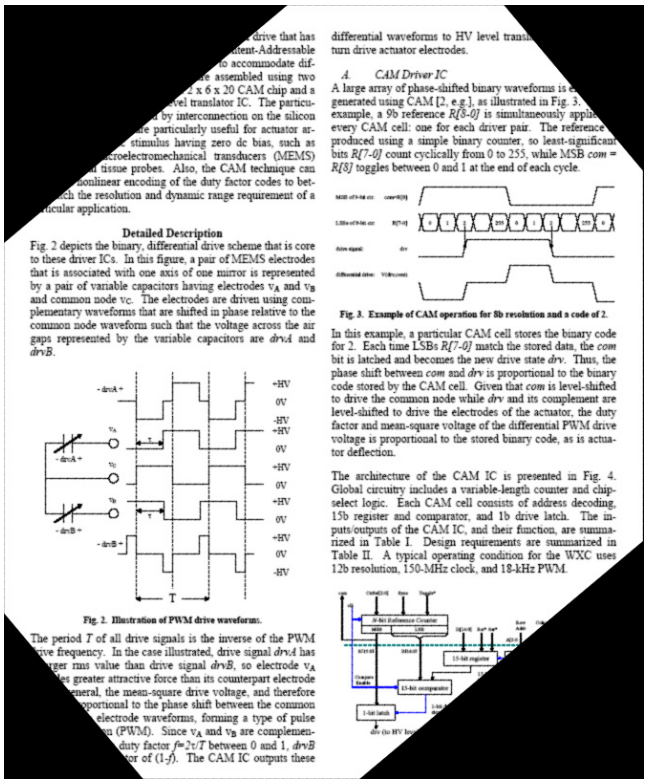


Figure 2. Skew corrected-image corresponding to Figure 1

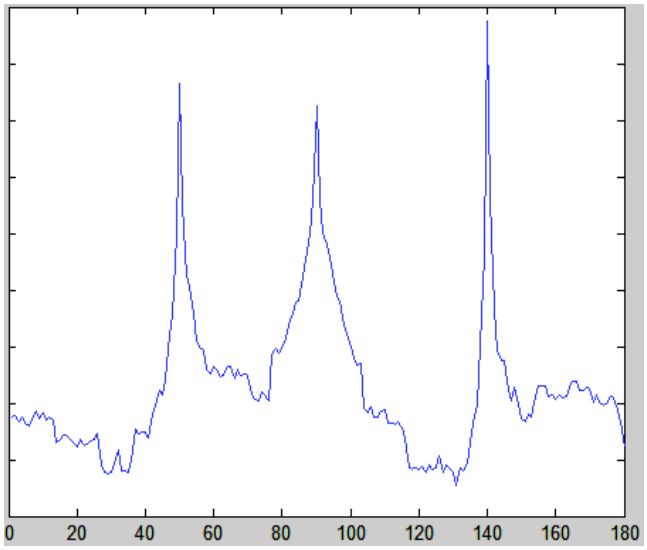


Figure 3. Plot of S (Θ) for skewed image of Figure 1. Highest peak corresponds to the skew. For un skewed images, highest peak usually occurs at 90 degrees. Here, peak occurs at 140 degrees corresponding to a 50 degree skew

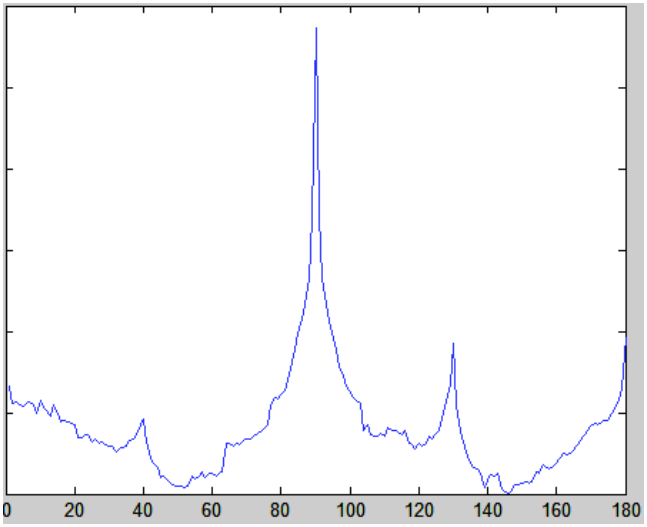


Figure 4. Plot of S (Θ) for skew corrected image of Figure 2. Highest peak corresponds to the skew. It is now at 90 degrees which signifies no-skew

We employ the skew correction algorithm to a variety of test-images to test its functionality. Our algorithm has the capability to fix skews in the range of -35 degrees to 35 degrees for text with few images (worst case performance).

For images with high content of text lines, we observe an improved skew correction range. Skew correction capabilities over the range of -80 degrees to 80 degrees (best case performance) have been reported for simple text images with no figures.

The skew correction capabilities are largely dependent on the content of the document image. Images rich in hori

zontal lines of text pose no problem and are faithfully corrected. Images with diagrams and figures behave differently if the image is dominated in content by the diagram /figure. The skew correction capabilities deteriorate for text document images with larger number of figures.

For e.g., skew in document text image of Figure 1, could be corrected for the range -50 degrees to 50 degrees. It contains 3 figures.

Our algorithm can also correct for negative angles as illustrated by the following figures.

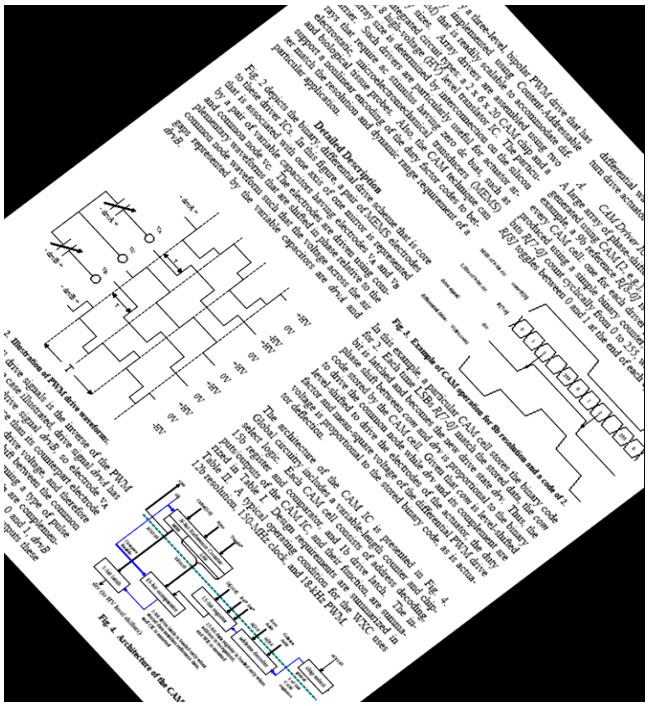


Figure 5. Skewed image (negative (clockwise) -50 degrees skew)

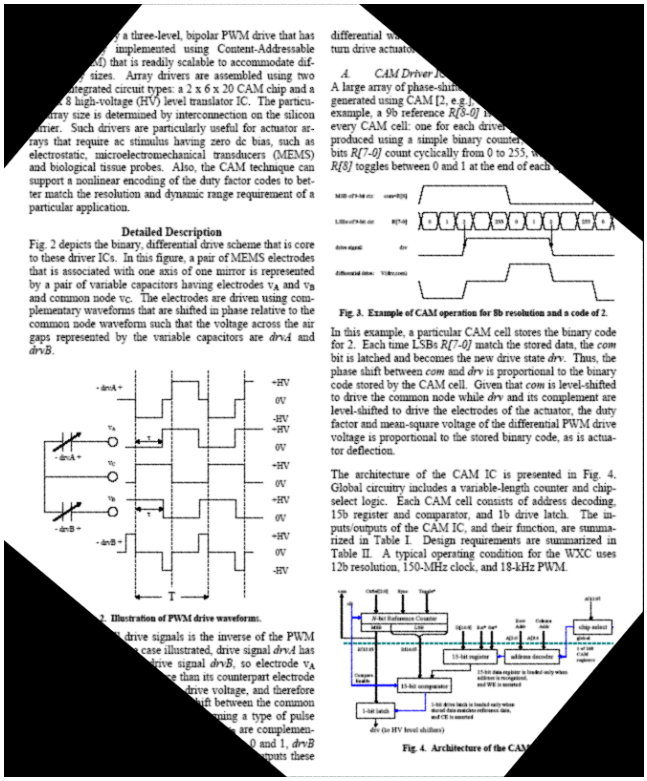


Figure 6. Skew corrected image corresponding to Figure 5

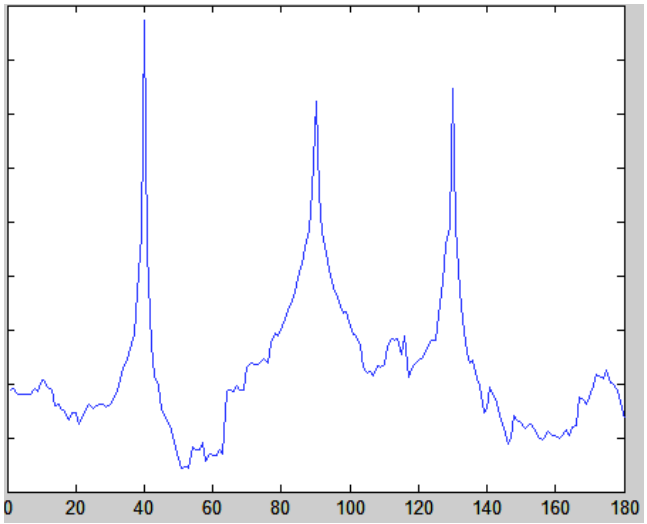


Figure 7. Plot of S (Θ) for skewed image of Figure 5. Highest peak corresponds to the skew. For un skewed images, highest peak usually occurs at 90 degrees

As expected, we see that the peaks for the positive-angle skewed image and negative-angle skewed image are 90 degrees apart.

Following figures illustrates the capability of our algorithm to detect skew angles as large as 80 degrees (both positive and negative).

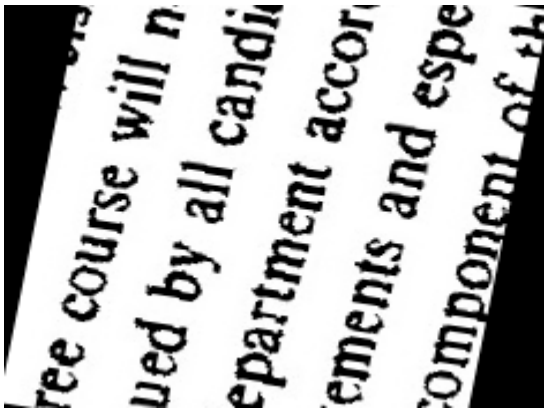


Figure 8. Skewed image (positive (counter-clockwise) 80 degrees skew) with no figures

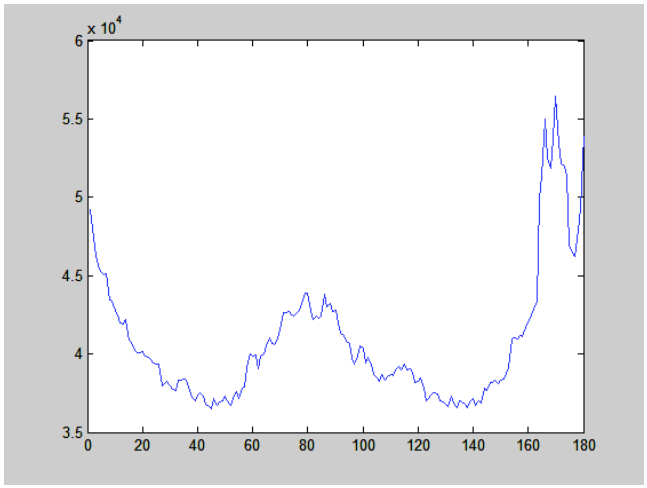


Figure 9. Plot of S (Θ) for skewed image of Figure 8. Highest peak corresponds to the skew. For un skewed images, highest peak usually occurs at 90 degrees

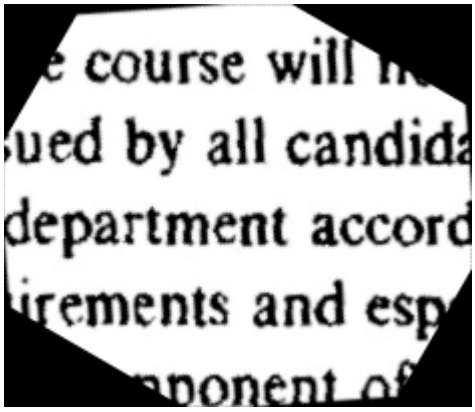


Figure 10. Skew corrected-image corresponding to Figure 8

Our observations are illustrated by Figures [7-9]. As Figure 7 has an image that is rich in text-lines and has no other prominent peaks corresponding to diagrams/figures etc. embedded in the text, skew angles detected were in the range of -80 degrees to 80 degrees.

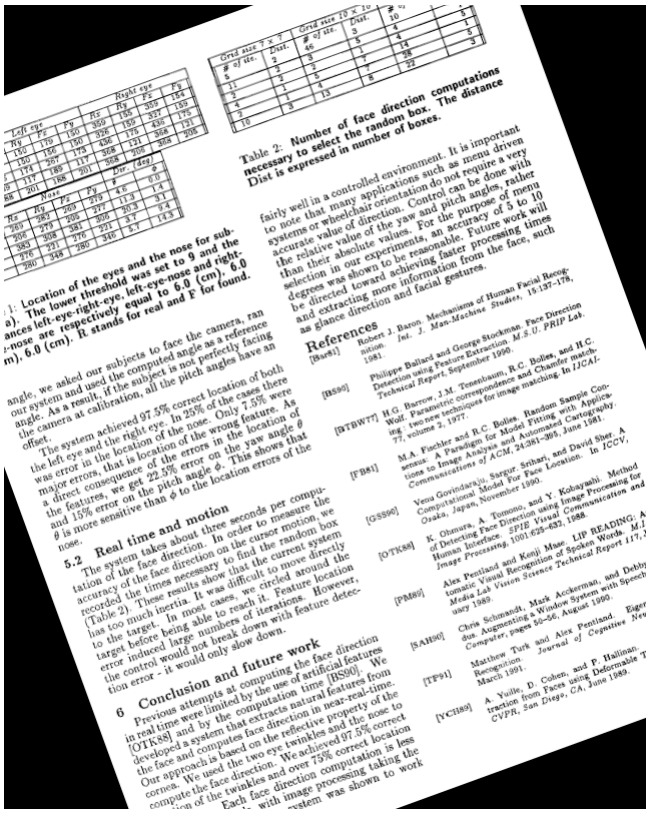


Figure 11. Skewed Image with tables (35 degree skew)

Figure 12. Skew corrected image corresponding to Figure 11

Our program can fit a given bounding-box (polygon) of certain dimensions to the text thereby removing text outside the text box. The dimensions of the bounding box need to be provided in the program depending on the requirement.

The bounding-box fitting is performed using matlab function 'roipoly'.

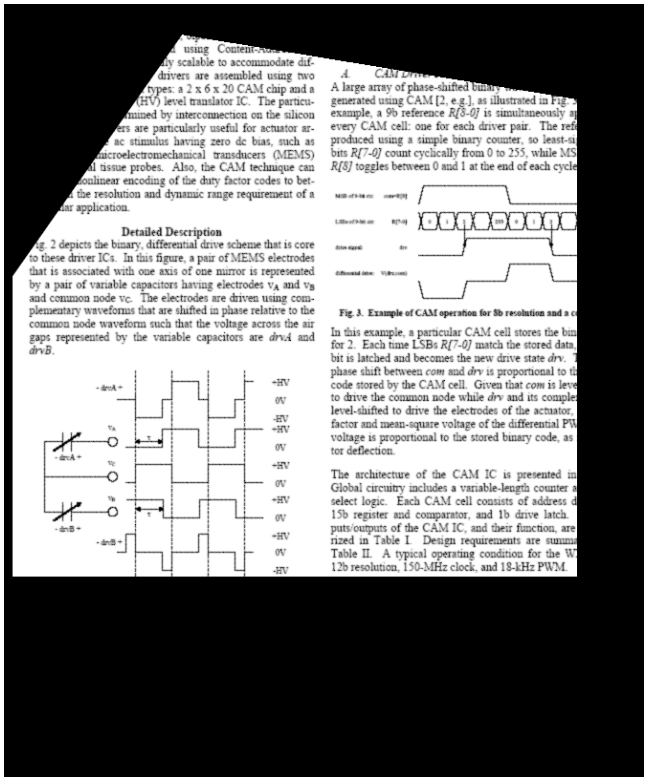


Figure 13. A polygon of arbitrary size fitted to the document text in Figure 2 to illustrate bounding-box fitting capability.

SUMMARY

An algorithm for skew detection/correction of document images has been implemented to correct for skew angles in a wide range of -80 degrees to 80 degrees for simple text. Program developed is based on spectral texture technique. The program works reliably for images with text lines as the dominant content.

REFERENCES

[1]. DIGITAL IMAGE PROCESSING USING MATLAB® ,RAFAEL C. GONZALEZ, RICHARD E. WOODS AND STEVEN L. ED DINS, PRENTICE-HALL, 2004.

[2]. Su Chen, Robert M. Haralick, and Ihshin Phillips; 'Automatic Text Skew Estimation in Documents', 1995., Proceedings of the Third International Conference on , Volume: 2 , 14-16 Aug. 1995. Pages: 1153 - 1156 vol.2 (pdf, 376 kB)

[3]. Changming Sun and Deyi Si; 'Skew and Slant Correction for Document Images Using Gradient Direction', 1997. Proceedings of the Fourth International Conference on , Volume: 1 , 18-20 Aug. 1997. Pages: 142 - 146 vol.1 (pdf, 668 kB)

[4]. Huiye Ma and Zhenwei Yu; 'An Enhanced Skew Angle Estimation Technique for Binary Document Images', 1999. ICDAR '99. Proceedings of the Fifth International Conference, 20-22 Sept. 1999. Pages: 165 - 168. (pdf, 40 kB)

[5]. Reza Safabakhsh and Shahram Khadivi, 'Document Skew Detection Using Minimum Area Bounding Rectangle', 2000. Proceedings. International Conference, 27-29 March 2000. Pages: 253 - 258. (pdf, 288 kB)

[6]. Abhishek Gattani, Maitrayee Mukerji and Hareish Gur; 'A Fast Multifunctional Approach for Document Image Analysis', 2003. Proceedings. Seventh International Conference, 3-6 Aug. 2003. Pages: 1178 - 1182. (pdf, 260 kB)

Optical Character Recognition Preparation Using Matlab

Michael K. Lee
Department of Electrical Engineering and Computer Science,
Case Western Reserve University, Cleveland, OH, Email: mkl7@cwru.edu

ABSTRACT
This paper presents an algorithm that detects the skew angle and aligns the text to be horizontal. Also, it erases the surrounding margins so that an Optical Character Recognition software would perform better. Despite the simplicity of the algorithm, it produced an acceptable quality results.

KEYWORDS
Skew Angles, Text Aligning

INTRODUCTION
When digitizing a hard copy of a document with a scanner, it is likely to place the text off center and skewed by an unknown angle. For the character recognition software to perform better, it would be a preferable idea to align the text to the center, rotate it back by the skewed angle, and crop off any unnecessary margins from the image.

DETERMINING THE SKEW ANGLE
The approach I made was to apply a Fourier Transformation on the scanned image and see in which direction the frequency components are aligned. Since the text would be written in the horizontal direction, and a lot of English characters consist vertical strokes, the Fourier Transformed image would tell us how much angle the text has been rotated.

I tried to find the average of gray levels along a line across the center of the image with various angles. For an image with width a and height b, the equation for the line would be

$$(y - b/2) = \tan\theta\ (x - a/2)$$

The line with the greatest average would probably be the skew angle of the text, but I came to minor problems. Since I was using functions over x, I was only picking one pixel per column. If the angle θ is too steep, I could only have few pixels to sample. Therefore, this method could have errors due to lack of sampling size at high angles. For example, Figure 1 has 8 pixels to sample where at higher angles (Figure 2) has only 4 pixels along the line.

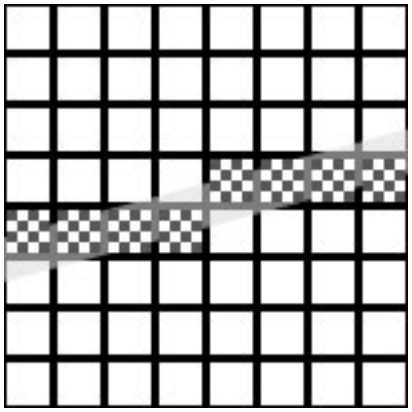


Figure 1. Averaging Gray-levels at Low Angles

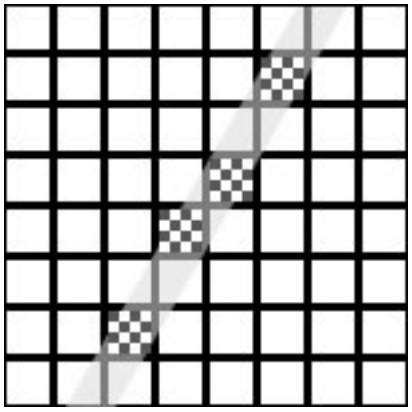


Figure 2. Averaging Gray-levels at High Angles

In order to overcome this problem, I used the inverse equation, saying:

$$(x - a/2) = (y - b/2) / \tan\theta$$

Now this equation is over y, I could do the same thing and get enough pixels to get a more accurate skew angle detecting algorithm.

ROTATING THE IMAGE BACK

Once I found out what the skew angle is, I could rotate it back using simple geometric transformations. This takes two phases – the first part was spatial transformation. Rotating an image with coordinates (x, y) along the axis by the angle θ will be

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

where (x', y') becomes the transformed image's coordinates. However, if I wanted to rotate the image along the center, I had to shift the image beforehand. A Translation transformation would simply adding/subtracting the new offset of the coordinates as

$$x' = x + x_0$$
$$y' = y + y_0$$

Therefore, the actual spatial transformation takes three steps – first to align the center at (0, 0), then to rotate it by the angle θ, and finally shift the image back to its original orientation. If we write down the coordinates (x,y) as a column matrix, the translation transformation to (x_0, y_0) could be expressed by multiplying a 3x3 matrix as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Also, rotating an image along the axis by the angle θ will be the matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Therefore, the whole spatial transformation could be done in a series of matrix multiplications. If the center of the image is (a/2, b/2) and the skew angle is θ,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a/2 \\ 0 & 1 & b/2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) & 0 \\ \sin(-\theta) & \cos(-\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -a/2 \\ 0 & 1 & -b/2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Now we know which coordinates (x, y) correspond to the rotated coordinates (x', y'). However, (mostly due to the trigonometry functions) the transformation can yield noninteger coordinates (x, y) – because the original scanned image is digital, the pixel values are defined only at integer coordinates. Therefore, we need to decide the gray-level values at those locations based on the pixel values of the neighboring integer coordinate locations. This process is called gray-level interpolation, which is the second phase of the geometric transformation.

There are several gray-level interpolations to use, but I decided to use bilinear interpolation approach – since I know the gray-levels of the neighboring integer coordinates, the gray-level value at a non-integer coordinate (x, y), which I'll call v(x, y), can be interpolated from the gray-level values by using the relationship

$$v(x, y) = a\,x + b\,y + c\,x\,y + d$$

where the four coefficients (a, b, c, and d) are determined from the four equations that can be written using the four known neighbors of (x, y).

In Matlab, I made a function named getcoor() for the spatial transformation, and another function named applyt() that does the rotation and the gray-level interpolation.

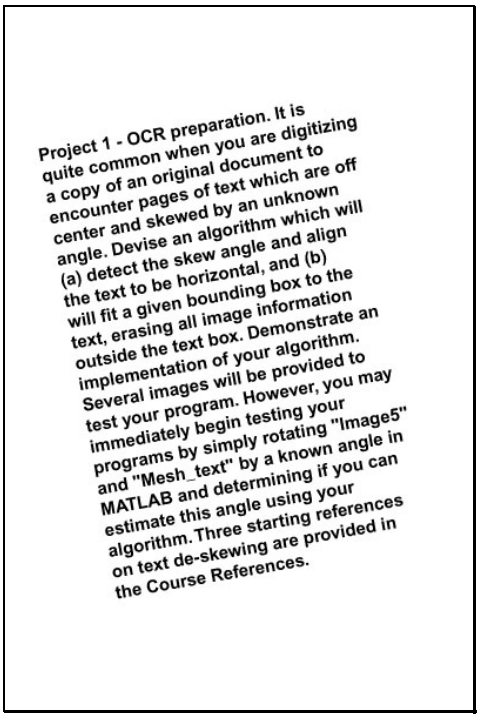


Figure 3. A Skewed Text Image

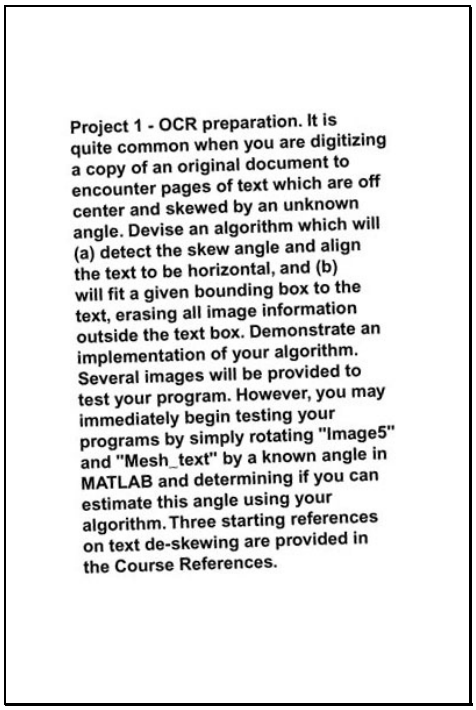


Figure 4. After Rotating Figure 3.

REMOVING THE BORDERS

The only part left is to crop out the unnecessary borders from the image. Since the text would be aligned, I assume the text would be in a box in the middle of the image. I first made a duplicate of the aligned text and applied an average filter, so I could reduce the effect of noises for the border detection and a reasonable margin around the text. Since the plain background would be white, or at least a bright color, I found the rows and columns which pixels are all above some threshold gray level. These rows will be backgrounds without any text information, and I only needed to remove the consecutive non-information rows beginning from the top and bottom, and the same thing goes to the columns. Now I know which rows and columns are backgrounds, I could crop the border from the original align image. The result image now only has the minimal margins along the text.

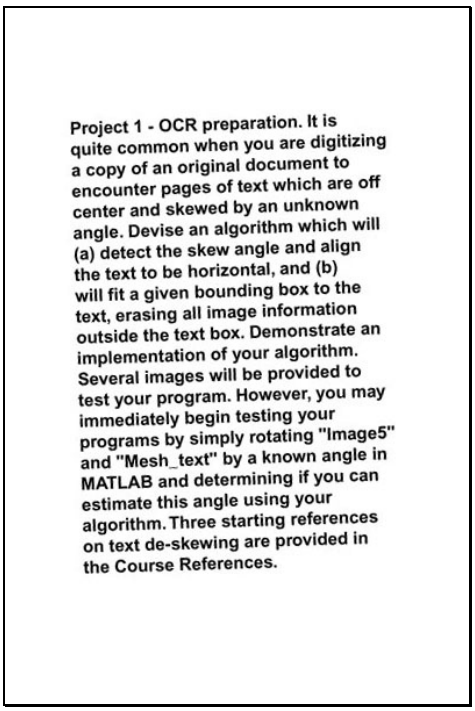


Figure 5. Before Removing the Borders

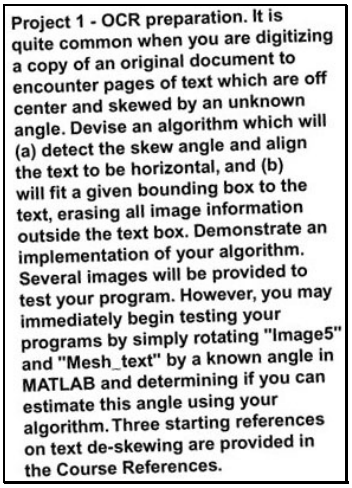


Figure 6. After Removing the Borders from Figure 5.

RESULTS AND DISCUSSION

After implementing all functions, I made a wrapper function that calls all other functions so every step would be done in one statement. I made a few skewed text image using Adobe Photoshop and ran several test runs, and the results had acceptable quality.

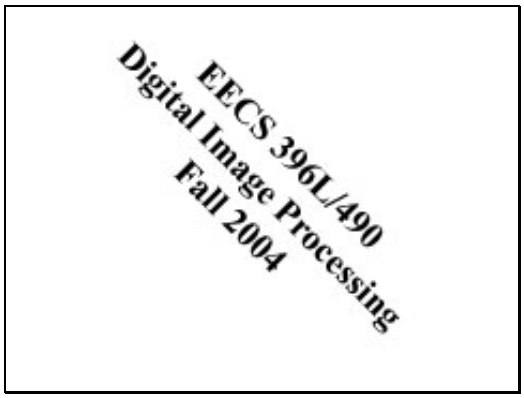


Figure 7. A Skewed Text Image

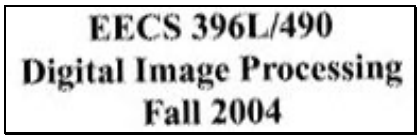


Figure 8. After Performing the Preparation Funcion on Figure 7.

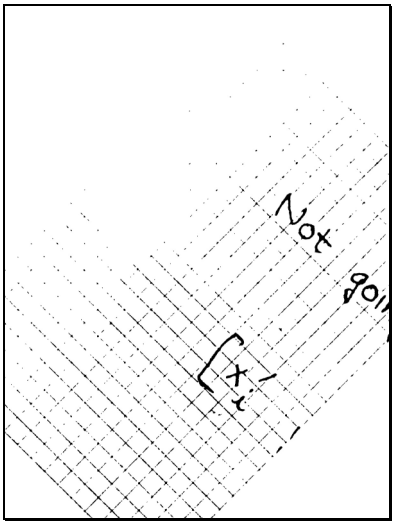


Figure 9. Another Skewed Text Image

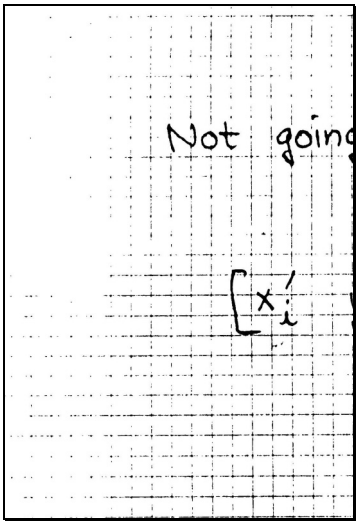


Figure 10. After Performing the Preparation Funcion on Figure 9.

SUMMARY

The Fourier Transformed image had enough information to find the skew angle of the text. The problem I had was the sample size issue, which resolved without any big issues. After determining the skew angle, I performed a series of geometric transformation to rotate it back, and cropped out the unnecessary borders. Despite of the simplicity of the algorithm, it was able to perform a fine job yielding an acceptable result.

REFERENCES

[1] Rafael C. Gonzalez, Richard E. Woods, "Digital Image Processing" Second Edition, Prentice Hall, 2002.

[2] Kenneth R. Castleman, "Digital Image Processing", Prentice Hall, 1995

Skew Correction for Document images Using Gradient and Fourier analysis

Deng-Hung Liu

Department of Electrical Engineering and Computer Science,

EECS 490 Digital Image Processing, Midterm Project
Case Western Reserve University, Cleveland, OH, Email: dxl74@cwru.edu

Abstract

Detection of document skew is an important step in any Optical Character Recognition (OCR) and document image analysis system. This report describes two algorithms to estimate the text skew angle in a document image. Then the skew image is corrected by a rotation at such an angle which is estimated by the algorithm. The skew angle is obtained by looking for a peak in histogram of the gradient orientation or Fourier analysis of the input grey-level image. The method is not limited in the range of detectable skew angles and the achievable accuracy. Simulation results which is demonstrated by Matlab show the high performance of the algorithms in detecting document skew for a variety of documents with different levels of complexity.

Keywords

Skew correction, gradient, Fourier analysis

Introduction

The conversion of paper-based documents to electronic image format is important in systems for automated document delivery, document preservation and other applications. Document conversion includes scanning, displaying, quality assurance, image processing, text recognition, and creating image and text databases. Document skew is a distortion that often occurs during document scanning or copying. During the document scanning process, the whole document or a portion of it can be fed through the loose-leaf page scanner. Some pages may not be fed straight into the scanner, however, causing skewing of the bitmapped-images of these pages; these pages may eventually be identified and rescanned by a quality control operator. In an attempt to partially automate the quality assurance process as well as to improve the text recognition process, a document skew angle detection algorithm has been developed.

Many methods have been developed for the correction of skewed document images.

1. Baird's algorithm: First, Baird applies a connected component analysis to the document. The midpoint of the bottom of each connected component is then projected onto an imaginary accumulator line perpendicular to different projection angles. For each projection direction, the sum of squares of the accumulated values of each bin is calculated, although Baird notes that any positive super-linear function provides the same result.
2. Le et al. algorithm: This algorithm applies the connected component analysis to the original image. The nearest-neighbor chain are extracted from the adjacent nearest neighbor pairs, in which the slopes of the nearest-neighbor chain with a largest possible number of components are computed to give the skew angle of document image.

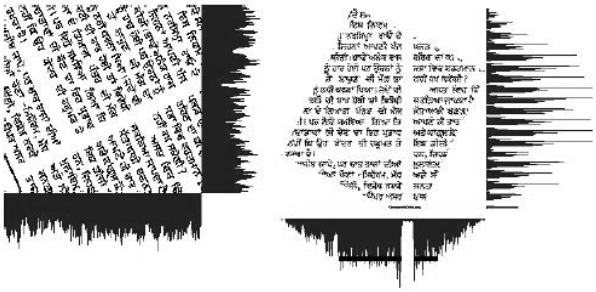


Figure 1. skew correction by project profile method

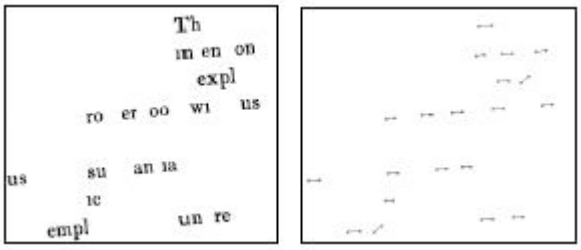


Figure 2.Nearest neighbor based method

3. Hinds et al. algorithm: Hinds et al. applies the vertical run length analysis for the image. A gray scale is created from the black run lengths that are perpendicular to the text lines by placing the length of the run in the run's bottom most pixel. The Hough transform is then applied to each of these burst image. A line function passes through an edge point p is determined by an angle θ and distance d: $x\cos\theta + y\sin\theta = d$, as shown in Fig.3. Hough transform specifies all possible angles θ to the points, and calculates the corresponding distance d. By using a accumulator A

for all d and θ, the peaks in A are the most probable lines in the image.

4. Fourier transform method: By using 2D Fast Fourier transform. The image is filtered in the Fourier domain to retain predominantly text and edge information. A direction histogram is extracted from the Fourier spectrum of the image. The histogram yields the predominant direction of text in the document. We can get estimated angle form the predominant direction of text.

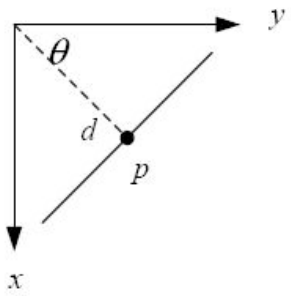


Figure 3.Hough transform method

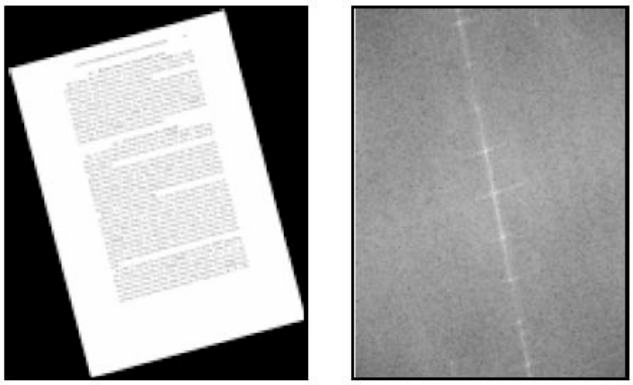


Figure 4.Fourier transform method

5. O'Gorman's algorithm: O'Gorman's discusses his skew detection method as part of a structural page layout analysis system, which detects the inline and between-line spacing, and locates text lines and blocks as well as calculating the skew angle.

In this report we implement two simple and fast algorithms for determining the skew angle of an image only using the orientation histogram. Their development are based on the observation that the gradient orientation should be mainly in the direction perpendicular to the text line and predominant orientation of the text in the spatial domain. After the skew angle has been determined, the image can then be rotated for correction.

Overview

In this section, we outline the main points of the skew detection algorithm. The documents are always scanned as grey scale images at 300 dpi resolution. Though this resolution is required for the recognizers, skew estimation can be done at much lower resolution. The input image is downsampled to 96 dpi to reduce computational complexity. The resultant image transformed in to gradient by using Sobel filtering operation. A directional histogram is extracted from the orientation of this gradient vector and polar form of Fourier tansform. The histogram yields the predominant direction of text in the document. The estimated angle is used to correct the skew in the original document.

Gradient analysis

In this section, the application of gradient analysis on documents images is explained in detail. First-order derivatives of a digital image are based on various approximations of the 2-D gradient. The gradient of an image $f(x, y)$ at location (x, y) is defined as the vector

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

It is well

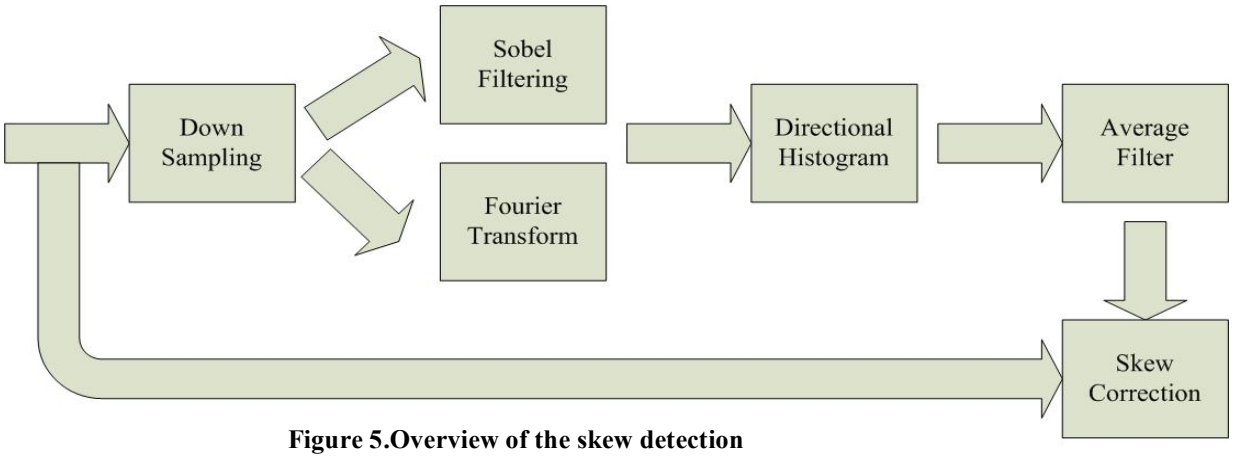


Figure 5.Overview of the skew detection

known from vector analysis that the gradient vector points in the direction of maximum rate of change of f at coordinates (x, y).

An important quantity in edge detection is the magnitude of this vector, denoted ∇f, where

$$\nabla f = mag(\nabla f) = [G_x^2 + G_y^2]^{1/2}$$

This quantity gives the maximum rate of increase of $f(x, y)$ per unit distance in the direction of ∇f. It is common practice to refer to ∇f also as the gradient. The direction of the gradient vector also is an important quantity. Let $\theta(x, y)$ represent the direction angle of the vector ∇f at (x, y). Then , from the vector analysis ,

$$\theta(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

where the angle is measure with respect to the x-axis. The direction of an edge at (x, y) is perpendicular to the direction of the gradient vector at that point.

Computation of the gradient of an image is based on obtaining the partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ at every pixel location. These derivatives can be implemented for an entire image by using an $N \times N$ (7×7)Sobel filtering operation.

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 3 & 2 & 1 & 0 & -1 & -2 & -3 \\ 4 & 3 & 2 & 0 & -2 & -3 & -4 \\ 5 & 4 & 3 & 0 & -3 & -4 & -5 \\ 6 & 5 & 4 & 0 & -4 & -5 & -6 \\ 5 & 4 & 3 & 0 & -3 & -4 & -5 \\ 4 & 3 & 2 & 0 & -2 & -3 & -4 \\ 3 & 2 & 1 & 0 & -1 & -2 & -3 \end{bmatrix}$$

$$\frac{\partial f}{\partial y} = \left(\frac{\partial f}{\partial x}\right)^T$$

The domain of θ is $[-\pi, \pi]$. For detection the orientation of a skewed document, half of the range of θ will be enough. I eliminated the negative value of θ. If we have some knowledge about the range of the skew angle, we may reduce the domain of The negative value of θ even more.

This algorithm is based on an observation about the gradient orientation distribution of the image. For a skewed document, there will be more points in the image whose gradient orientations are perpendicular to the document text lines. It is expected that the statistical information of the gradient orientation of an image can be used for skew angle detection. From the obtained histogram $h(\theta)$, the angle of the skewed document is the difference between $h(\theta)$ and $\pi/2$. For a non-skewed image, θ will be $\pi/2$,hence the skew angle is zero. The difference between $\pi/2$ and θ will be the skew angle.

The predominant direction may be obtain by

$$\text{Skew angle } = \phi = \max(h(\theta)) - 90$$

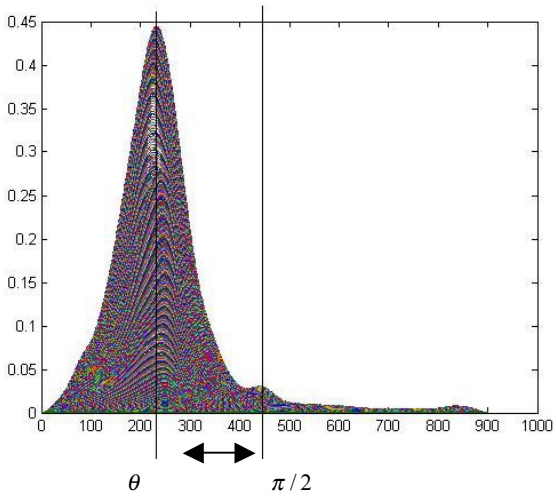


Figure 6. The skew angle (25) between $\pi/2$ and θ

Fourier Analysis

The Fourier domain description of an image yields orientation information more accurately. The discrete Fourier transform of a 2D image $f(x, y)$ of size $M \times N$ is given by

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$$

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)}$$

We shall represent the Fourier coefficients in the polar form instead as $F(r, \theta)$ where

$$r = \sqrt{u^2 + v^2}$$

$$\theta = \tan^{-1}\left(\frac{v}{u}\right)$$

We can get a directional histogram like the gradient analysis. The estimation of the skew angle involves determining the predominant direction of symbols in the document image. This direction is marked by high energy density in the Fourier spectrum. It is expected that this information can be determined by creating a directional histogram of the energy distribution in the spectrum.

Skew Estimation and correction process

After Sobel filtering or Fourier transform, we got the result from the orientation histogram. In most of the gradient orientation histograms, peaks often appear at 45 degree and its multiples. These distinct peaks have been removed by applying a median average filter over the histogram.

After we got the skew angle, we have to do the rotation process to correct the image skew.

$$\begin{bmatrix} X_{out} \\ Y_{out} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{skew} \\ Y_{skew} \\ 1 \end{bmatrix}$$

Then my proposed process for achieving the skew angle correction is :

I. Convert RGB image to grey scale (if needed).
II. Down-sampling image to 96 dpi in order to reduce computational complexity.
III. Perform gradient operation or Fourier transform on the image (on both x and y directions).
IV. Obtain the orientation histogram.
V. Smoothing this histogram with a median filter and let the peak we want become predominant.
VI. Search for maximum in this histogram to get an skew angle.
VII. Rotate image for skew correction.

Experiment results and conclution

Both gradient and Fourier analysis are using orientation histogram to search the maximum counts, this is why I want to do these two analysis in this report. I also applied some filter in the Fourier process in order to get accurate value.

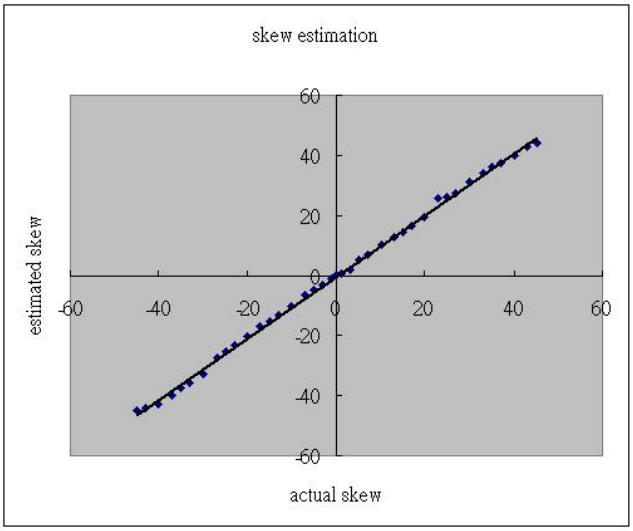


Figure 7. Plot indication the actual and detected angles

A number of documents with different levels of complexity from papers, journals, and books were selected to carry out experiments. These documents include diagrams, images formulas, and single and multicolumn English. The documents were rotated at different angles between [-45 45] degree and were digitized at 96 dpi .As the Figure 7. you can see above. The error is less than 5%.

Figure 8.Original and skew corrected document image containing only text information

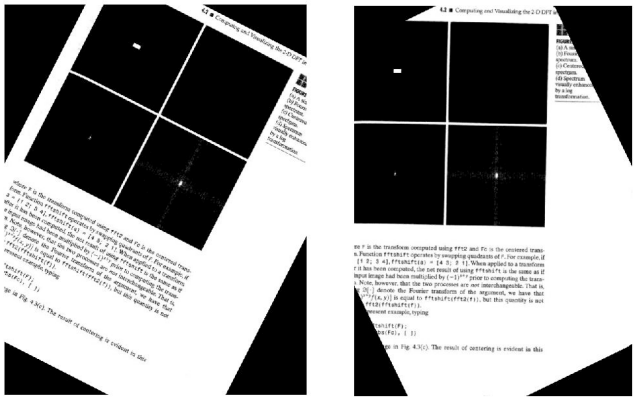


Figure 9.Original and skew corrected document image containing photos.

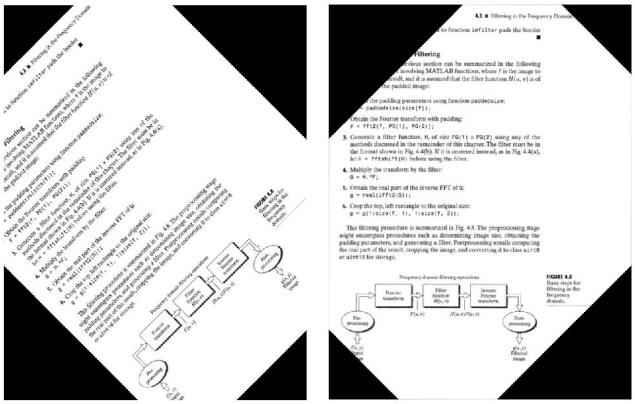


Figure 10.Original and skew corrected document image containing diagrams.

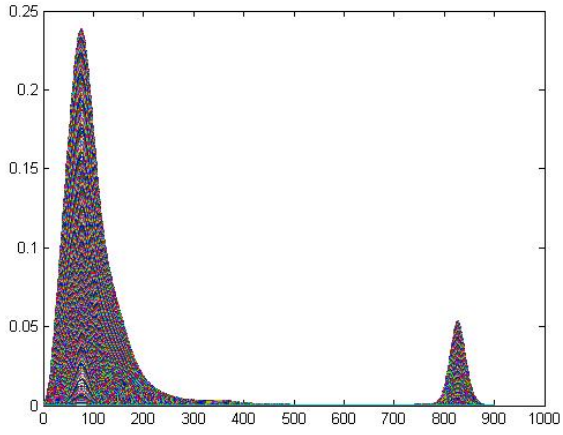


Figure 11. The histogram of Figure 10.

Referencece

[1] C. Sun and D. Si, Skew and Slant Correction for Document Images Using Gradient Direction, In: *Proceedings of the 4th International Conference on Document Analysis and Recognition*, Ulm, Germany, 1997, pp. 142-146.

[2] Huiye Ma, Zhenwei Yu, "An enhanced skew angle estimation technique for binary document images".

[3] S. Chen, R.M. Haralick, and I.T. Phillips, Automatic Text Skew Estimation in Document Images, In: *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, Montreal, Canada, 1995, pp. 1153-1156.

[4] Abhisek Gattani, Maitrayee Mukerji, Hareish Gur, "A fast multifunctional approach for document image analysis ", 2003 IEEE.

[5] Adnan Amin, Stephen Fischer, Tony Parkinson, Ricky Shiu, "Fast algorithm for skew detection".

[6] Hua Shen, Xiaoou Tang, "Generic Sign Board Detection in Images".

[7] A. Bagdanov and J. Kanai, Projection Profile Based Skew Estimation Algorithm for JBIG Compressed Images, In: *Proceedings of the 4th International Conference on Document Analysis and Recognition,* Ulm, Germany, 1997, pp. 401-405.

[8] L. O'Gorman, The Document Spectrum for Page Layout Analysis, *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 15, No. 11, 1993, pp. 1162-1173.

[9] B. Yu and A. Jain, A Robust and Fast Skew Detection Algorithm for Generic Documents, *Pattern Recognition,* Vol. 29, No. 10, 1996, pp. 1599-1629.

[10] S.C. Hinds, J.L. Fisher, and D.P. D'Amato, "A Document Skew Detection Method Using Run-length Encoding and the Hough Transform," *Proceedings, 10th International Conference on Pattern Recognition*, pp. 464-468, 1990.

[11] Rafael C. Gonzalez and Richard E. Woods, "Digital Image Processing"

Multi-technique Algorithm for De-skewing and Text Bounding for OCR Applications

Daniel Pendergast

Department of Electrical Engineering and Computer Science,
Case Western Reserve University, Cleveland, OH, Email: djp8@case.edu

Abstract

This paper presents an algorithm that combines various techniques known in the art of OCR pre-processing. Specifically, the concern is in de-skewing of document images and text/information bounding & cropping. The algorithm, a Fourier technique combined with statistical analysis of gradient information, performs well on a variety of test images.

KEYWORDS

OCR, de-skewing, Fourier, 'image gradient'

INTRODUCTION

OCR applications are currently in great demand in a variety of industries. Although focus in OCR research has shifted somewhat as of late to applications for handheld devices such as PDAs, demand is still high for processing of scanned document images. One common example today would be scanned images of business cards - from which contact information can be read and stored in a variety of database-type programs. OCR algorithms are typically very complex and sensitive to a number of factors that are potentially correctable in the pre-processing stage. One factor is document skew. Skew refers to the angle relative to the horizontal that the document is scanned in at. Most OCR algorithms assume that the input text will be aligned along a horizontal line.If it is not, the OCR algorithm will likely perform poorly. Most papers on the subject today, such as [1] categorize attempts to de-skew image documents into families of techniques. The basic categories include nearest-neighbor techniques, projection profiles, Fourier methods, Hough transforms, and correlation techniques. Other attempts such as [2], fitting a rectangle of least area, or [3] fall outside the scope of the general categories. The combination technique presented in this paper uses a simple Fourier technique for de-skewing, but incorporates some elements of [3].

Another area of interest in OCR pre-processing involves "text selection" or cropping the image to only include relevant information. This may be as simple as cropping out all surrounding white space from an image or as complex as an attempt to isolate text from charts or pictures within the document. The technique presented in this paper performs white space cropping and attempts a simple bounding box that isolates text within the image.

TECHNICAL APPROACH

The combination technique is a three-stage process. First, a document skew angle is estimated and the image is rotated to correct the skew. Next, the de-skewed image is cropped to include only the information within the image (the surrounding white space is removed). Finally, a rectangular bounding box meant to capture only the text in an image is fitted.

To estimate the skew angle, the algorithm first checks for the trivial case of a black background on the skewed document. It is conceivable that when scanning a document in, there may be a discernable background surrounding the document. If this is the case, skew angle detection becomes a trivial matter of measuring the angle of the borders of the document. However, unless this fortunate situation arises, the technique employs the use of the Fourier method. Before moving to the frequency domain, thresholding is carried out on the original image to produce a binary image. Next, run-length smoothing similar to that presented in [2] is performed. Any horizontal run of four white pixels or less that is surrounded by black pixels is set to black. This helps to make text lines more dominant. A transform of the image is then performed using a two-dimensional FFT. The transform image is shifted so that the dc component is centered. Then, a radial sweep of the transform is performed from -44.6 degres to 44.6 degrees in .2 degree increments. Two lines of pixel values (one at the test angle and one at 90 degrees above the test angle) are summed and stored in a test array. As there will be some rounding error in calculating the pixels that truly lie on the test line, each position in the test array contains a +/- .2

degree spread. The maximum value held in this array should correspond with the skew angle of the document image. The image is then rotated about the negative of this angle using bilinear interpolation.

The next stage involves cropping of the image to remove white space and extraneous outliers. A removal of white space only would involve a simple scan of the image to find the most extreme border pixels and then cropping of anything outside of these borders. To remove “outliers” however, a more complex routine is necessitated. The de-skewed image is filtered with a 3x3 sobel filter to find the significant horizontal edges. Each contiguous vertical run of white pixels in the filtered image is considered a “singular edge”. All the singular edges found in the filtered image are stored in an array that tracks their position as well as their length. From this data, an average length of horizontal edges in the image can be calculated. It is assumed that the scanned image contains predominantly, if not exclusively, text. It follows then that the majority of the horizontal edges in the image will have a length close to that of the height of the text. Edges that lie outside of one standard deviation of the mean value of edge lengths can then be ignored. The simple scan for border pixels described above can now be performed ignoring these outliers.

The final stage is a simple attempt to bound the text only in a document image. To do this, a count of the relevant edges contained in each column and each row is conducted for rows and columns within the white space bounding box created in the previous stage. From this data, the average count for each row and column can be found. Then, similar to the previous stage, the border rows and their 5 inner neighboring rows are evaluated to determine if they fall within +1.75/-.75 standard deviations from the average. If not, the border is moved in and then next set of rows is evaluated. An identical procedure is performed with the columns. The concept here, again based on the assumption that text is the principle component in the document image, is that non-text information will lie outside of this range and therefore be eliminated.

RESULTS

The de-skewing stage of the technique worked reasonably well, with an occasional error of one or two degrees. In the trivial case, there was no error as shown in Figure 1 below. Figure 2. shows the result of an implementation of the non-trivial case. Error usually only appeared, if at all, in document images containing pictures. The error can be attributed to frequency contributions made by picture components. Frequency range for the radial sweep was adjusted (highpass) in an attempt to minimize such error.

The white space/outlier cropping worked quite well with no test images failing this stage. This is show in Figure 1, 2 and 3. The selection of +/- 1 standard deviation for determination of “relevant edges” was experimentally found.

An example of a result of the third stage is shown in Figure 3. In this case, the technique was successful in eliminating the picture from the document image. The selection of +1.75/-.75 standard deviations was experimentally found.

The third stage has limitations known at the time of design. The algorithm will not completely detect or remove a picture that is surrounded by text on any side, as the algorithm “stops looking” once it is satisfied the border in question contains the appropriate statistics. In addition, the algorithm will eliminate lines of text at the bottom of the bounding box that run short of the average length of lines in the paragraph above it. It is suggested that only stage 1 and 2 be trusted, and stage 3 be considered to an prototypical estimate for further development and refinement.

Fig 1

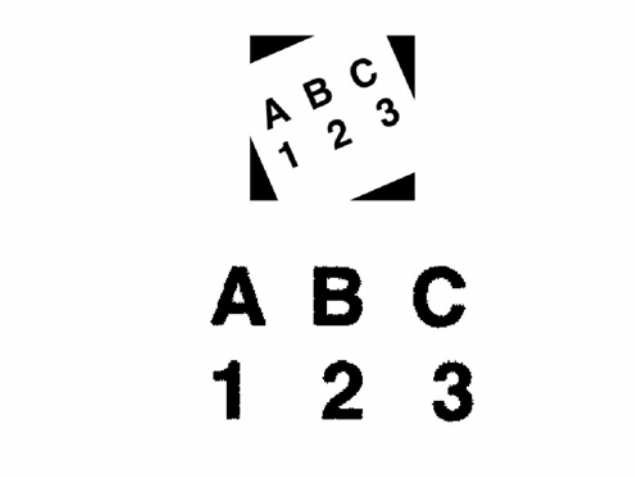


Fig 2

Fig 3

Fig 3b

CONCLUSIONS

As discussed in the results, stage 1 and 2 work within acceptable bounds, with stage 2 performing almost without error. Stage 3 could be developed further by performing a similar statistical examination on set regions throughout the document image as opposed to simply on the borders. This would also allow for a non-rectangular bounding box to be constructed, potentially eliminating all image components that are not text information. The original aim of this project focused on stage 1, the de-skewing of the image. There are a multitude of de-skewing techniques known in the art presently though, and less work has been done on text identification. Thus, emphasis was shifted towards an exploration of the text isolating algorithm.

ACKNOWLEDGMENTS

The author credits graduate education at Case Western Reserve University with inspiring examination of this subject.

REFERENCES

[1] C.Sun and D. Si, "Skew and Slant Correction for Document Images using Gradient Direction", *Proc. of Forth International Conference on Document Analysis and Recognition*, pp. 142-146, August 1997.

[2] R. Safabakhsh and S. Khadivi, "Document Skew Detection Using Minimum-Area Bounding Rectangle", *Proc. of The International Conference on*

Information Technology:Coding and Computing, pp. 253-258, March 2000

[3] H. Ma and Z. Yu, "An enhanced Skew Angle Estimation Technique for Binary Document Images", *Proc. of Fifth International Conference on Document Analysis and Recognition*, September 1999

OCR Preparation
Design of an Algorithm for Document Alignment Using the Global Processing via Hough Transform

Iouri Petriaev

Department of Electrical Engineering and Computer Science,

Case Western Reserve University, Cleveland, OH, Email: iap4@cwru.edu

Abstract

This paper presents the design details of an algorithm which will (a) detect the skew angle and align the text to be horizontal, and (b) will fit a given bounding box to the text, erasing all image information outside the text box. The solution discussed in this paper explains application of Hough Transform and regional descriptors to given problem. Commonly used for obtaining properties of the given regions, regional descriptors in this solution are combined with the Hough Transform, which allows creation of automatic pictorial pattern recognition and scene analysis algorithm.

KEYWORDS

Hough Transform, Chain codes, Regional descriptors, Digital Image, IPT, Freeman codes

INTRODUCTION

Digital image interpretation and object recognition have been challenging but desirable for many applications, especially robotics. One of the main elements of image interpretation is segmentation. Segmentation subdivides an image into its constituent regions or objects. Segmentation accuracy determines the eventual success or failure of computerized analysis procedures. That is why especial care should be taken to improve probability of rugged segmentation. In many situations, such as robotics, some degree of control over the environment is implemented.

Most of the segmentation algorithms are based on discontinuity and similarity. Those based on discontinuity partition an image based on abrupt changes in intensity, such as edges in a digital image. Algorithms based on similarity partition an image into regions that are similar according to a set of predefined criteria. Examples of this category of algorithms include thresholding, region growing, and region splitting and merging.

In this solution, we concentrate on image discontinuities. One of the main characteristic of an object in real life as well as in digital image is a boundary (edges) with in which the object is confined, and which determine its shape, area, radius and many other properties. Numerous edge detection algorithms such as Sobel, Prewitt, Roberts gradients seldom characterize an edge completely because of noise, breaks in the edge from non-uniform illumination and spurious intensity discontinuities. One of the approaches of solving the edge discontinuities concentrates on the analysis of the characteristics of pixels in a neighborhood or predefined size (3x3, 5x5 and such) around every pixel of an image which is fit to be an edge. Global Processing via the Hough Transform is the approach that is used in this solution.

In Hough Transform algorithm, edge points are linked through determining if they lie on a curve of specified shape. This approach, unlike the one mentioned before, concentrates on global relationships between pixels. Hough [1962] considered a fact that infinitely many lines pass through point (x_i, y_i), and they all satisfy the equation $y_i = ax_i + b$ for varying values of a and b. Rewriting this equation as $b = - ax_i + y_i$ and considering the ab-plane yields the equation of a single line for a fixed pair (x_i, y_i). The next point (x_j, y_j) also has a line within the same parameter space which intersects the line associated with (x_i, y_i) at (a', b'), where a' is the slope and b' is the intercept of the line containing both (x_i, y_i) and (x_j, y_j) in the xy-plane. All points located on this line contain lines in ab-plane that intersect at (a', b').

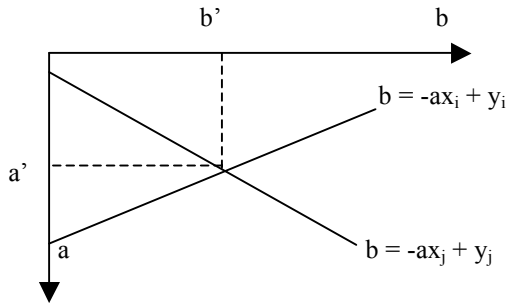


Figure 1. Parameter space (ab-plane).

Hough transform subdivides the parameter space into accumulator cells where (a_{max}, a_{min}) and (b_{max}, b_{min}) are the expected ranges of slope and intercept vlues. For instance, the cell at (i, j) coordinates and with accumulator value $A(i, j)$ corresponds to the square associated with parameter space coordinates (a_i, b_j).

Implementation of transform starts with all accumulator cells initialized to zero. Letting the parameter a equal each of the allowed subdivision values on a-axis for every point (x_k, y_k), we solve for the corresponding b using the equation $b = -ax_k + y_k$. Later, resulting b's are rounded off to the nearest allowed value in the b-axis. If chosen a_p results in

solution b_q, we let $A(p,q) = A(p,q) + 1$. The entire procedure ends with a value of Q in $A(i,j)$ corresponding to Q points in the xy-plane located on the $y = ax_i + b_i$.

The described above solution has a flaw. As a line approaches vertical direction, the slope of the line (a) approaches infinity. A suggested work around this difficulty is a normal representation of a line: $x\cos\theta + y\sin\theta = \rho$

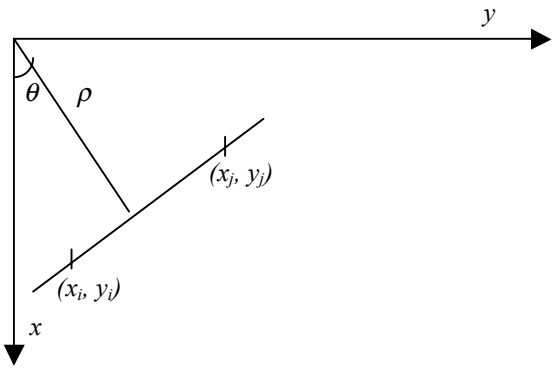


Figure 2. (ρ', θ') parameterization of lines in the xy-plane

The following properties of this solution need to be mentioned. With ρ being equal to the positive x-intercept, a horizontal line has $\theta = 0^0$. A vertical line has $\theta = 90^0$, with ρ being equal to the positive y-intercept, or $\theta = -90^0$, with ρ being equal to the negative y-intercept. The intersection point (ρ', θ') corresponds to the line that passes through both (x_i, y_i) and (x_j, y_j). The accumulator cells are generated from subdividing the $\rho\theta$-parameter space.

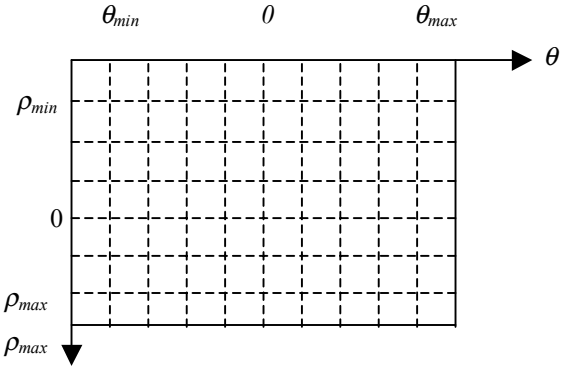


Figure 3. Division of the $\rho\theta$-parameter space into accumulator cells

In the following Figure 4, the first image shows the digital image with five dots. The second image demonstrates its Hough transform obtained by *hough* function in MATLAB.

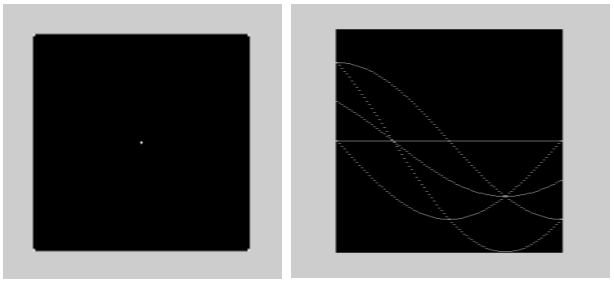


Figure 4. (a) Binary image with five dots (four of the dots are in the corners and one in the middle). (b) Hough transform.

Each of the five points on the first image is mapped onto the $\rho\theta$- parameter space as shown in the second image. Each line on the second image corresponds to a one point on the first one. The horizontal line resulting from the mapping of one of the points is a special case of a sinusoid with zero amplitude.

Hough transform is appliable to any function of the form $g(v,c) = 0$, where v is a vector of coordinates and c is a vector of coefficients. For example, the points lying on the circle

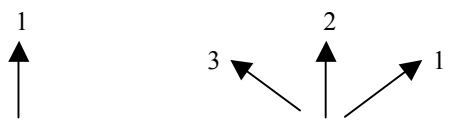
$$(x - c_1)^2 + (y - c_2)^2 = c_3^2$$

can be detected by using the approach just discussed.

An edge-linking approach based on the Hough transform is as follows:

1	**Compute the gradient of an image and threshold it to obtain a binary image.**
2	**Specify subdivisions in the $\rho\theta$-plane.**
3	**Examine the counts of the accumulator cells for high pixel concentrations.**
4	**Examine the relationship between pixels in a chosen cell.**

It is a common practice to use schemes that compact the data onto representations after obtaining raw data in the form of pixels along a boundary. These representations contain data more useful in the computation of descriptors. Chain codes represent a boundary by connecting a sequence of straight-line segments of specified length and direction. Usually, Chain codes representations are based on 4- or 8-connectivity of the segments. The direction of each segment is coded by using a numbering scheme such as the ones shown in Figure 5.



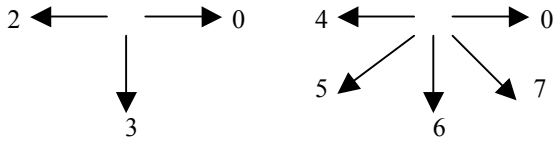


Figure 5. Direction numbers for a 4 – and 8 – directional chain codes.

The codes that are based on this scheme are referred to as Freeman chain codes.

DESIGN OF THE ALGORITHM

Besides line detection on a digital image, Hough transform and line detection algorithm compute both end-points of each line segment, the distance between each pair of these points, angle of the line segment (Hough transform bin, degrees) and position of Hough transform bin (rho-axis). This algorithm consists of four functions: *hough, houghpeaks, houghpixels and houghlines*.

First function computes Hough transform of the image *f*. It takes two more input parameters which are *dtheta* and *drho*. If the last two inputs are not provided, they are both set to 1.

Second function, *houghpeaks,* is used for finding a meaningful set of distinct peaks in the Hough transform matrix *h* passed to the function as an input parameter. The output parameters of this function are *r* and *c*. These parameters are the row and column coordinates of the identified peaks.

Third function, houghpixels, obtains pixels of the image belonging to the transform bin. Its output parameters, *r* and *c* are row-column indices for all nonzero pixels in image F that map to a particular transform bin.

The last function, *houghlines,* extracts line segments based on the Hough transform. The only output parameter of this function is the collection of line segments found on the original image.

In this solution, the Hough line-detecting algorithm is used to retrieve the properties of the lines located at the edges of a document with text. Despite its promising solution, line-detecting algorithm does not take for consideration the possibility of presence of lines that may present noise or should not be taken for account. Such lines could be a grid on a background of a page, line segment stretched across the page separating paragraphs and such. These segments could be interpreted as additional information that may help to detect the direction of the text, but they could not be relied upon during design of an algorithm for while detected on some documents they may not be present on the others.

The solution to this problem was found in function *boundaries*. This function traces the exterior boundaries of the objects in digital image f. Just us the Chain codes described before, this function can be based on 4– or 8- connectivity of the segments. It traces the boundaries of the objects either in a clockwise (*cw*-the default) or counterclockwise (*ccw*) directions. Assuming that the boundaries of the document with text are the longest and unique, we obtain them from the set of boundaries returned and ordered by their size. This allows as not only filter noise but the unwanted boundaries as well.

We apply function *bound2im* to the selected boundary to obtain a binary image of size $M \times N$ with 1s for boundary points and a background of 0s.

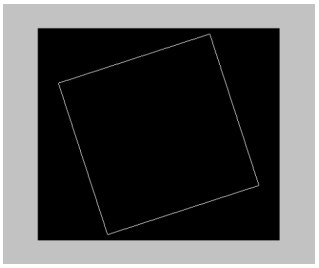


Figure 6. Boundary of the skewed document

Hough line-detector is applied to the resulting image and retrieves properties of the desired boundary. Thus, having the θ of the picked by the algorithm single line, we calculate the θ_{inv} by: $\theta_{inv} = 2\pi - \theta$. We use function *imrotate* of the IPT toolbox which rotates image by applying:

The translation of a bounding box of the document is calculated by obtaining difference between the center of the entire image and the centroid of the boundary. The centroid is calculated by the function *regionprops.* The document is

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where coordinate equations are:

$$x = \omega\cos\theta - z\sin\theta$$

$$y = \omega\sin\theta + z\cos\theta$$

centered in the image by applying IPT functions *translate* and *imdilate*. The first function creates the transformation matrix for image translation:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \partial_x & \partial_y & 1 \end{bmatrix}$$

where ∂_x and ∂_y are the transformation coefficients of the x and y coordinates. The *imdilate* function applies the transformation matrix to the image. To fit a bounding box to the text and erase all image information outside the text box we determine the scale of an image by calculating ratio of the width and height of the original image to the width and height of the bounding box. Then, knowing the coordinates of the bounding box we extract the image within in and create a new image. The scaling transformation is applied

to the resulting image with produces the desired effect. The transformation matrix for the image scaling has the following form:

where s_x and s_y are scaling factors of x and y coordinates. The IPT function used for generation of scaling matrix was *maketform*. The IPT function used to apply scaling to an image was *imtransform*.

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

RESULTS AND DISCUSSION

A test function *skew* was created to allow testers to load an image and have that image skewed. After skewing, the resulting image can be passed to the function *orcprep* described above.

The set of images below shows starting from the left image original image, bounding box with bottom line selected (colored green). Green line indicates line selected by algorithm for rotation angle θ calculation. In this example the angle is approximately equals to 30^0. The bottom two images show image after the translation was performed, and text scaled to fit the entire image correspondingly.

Trial 1:

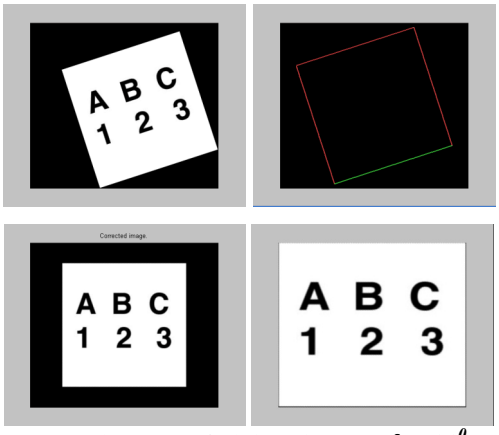


Figure 7. Original image: $\theta = 30^0$

The table below shows rotation, translation and scaling values obtained during program execution and applied during image transformation.

Rotation	Translation		Scale	
θ	X	Y	X	Y
342	19	-30	1.6273	1.5102

The next sequences of five images similarly demonstrate entire process of image segmentation and alignment of the text box including one extra image (1st in the second row) after application of rotation to the original image.

Trial 2:

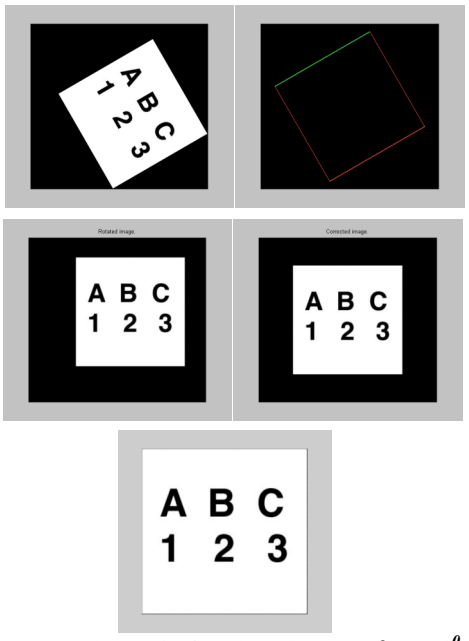


Figure 8. Original image: $\theta = -60^0$

Rotation	Translation		Scale	
θ	X	Y	X	Y
420	19	-31	1.6273	1.5102

Trial 3:

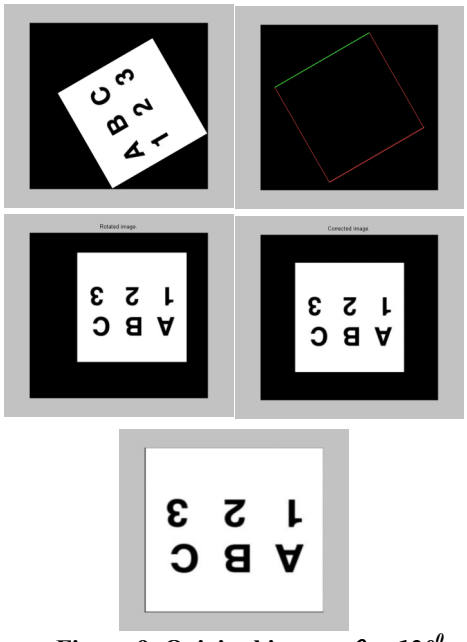


Figure 9. Original image: $\theta = 120^0$

Rotation	Translation		Scale	
θ	X	Y	X	Y
420	19	-30	1.6273	1.5102

In the following three trials, we will perform the same tests on a different image. That image contains a grid on its background and is filled with a handwritten text.

Trial 4:

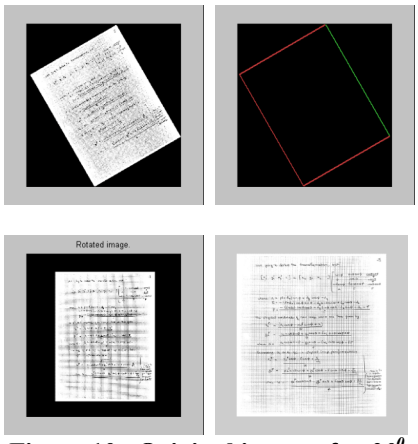


Figure 10. Original image: $\theta = 30^0$

Rotation	Translation		Scale	
θ	X	Y	X	Y
330	-11	-12	1.1997	1.6313

Trial 5:

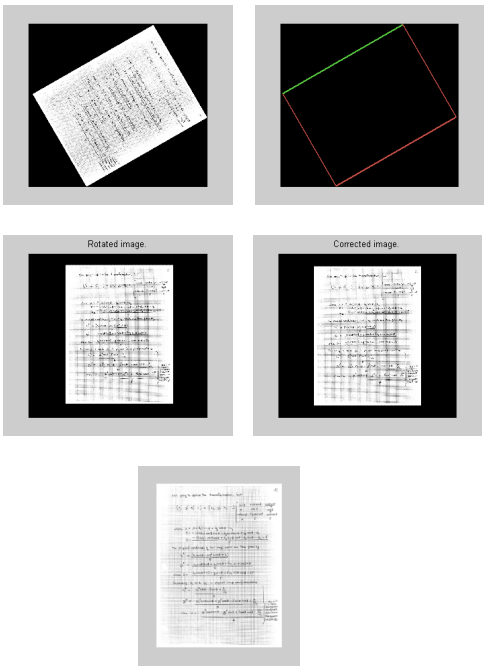


Figure 11. Original image: $\theta = -60^0$

Rotation	Translation		Scale	
θ	X	Y	X	Y
420	12	-11	1.2821	1.5247

Trial 6:

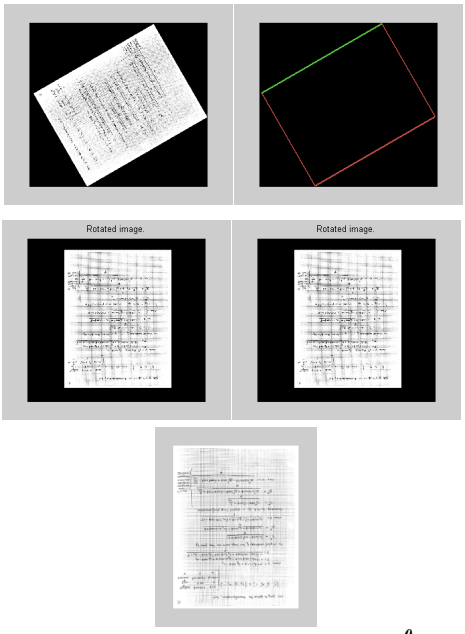


Figure 12. Original image: $\theta = 120^0$

Rotation	Translation		Scale	
θ	X	Y	X	Y
420	12	-12	1.2821	1.5247

All six trials demonstrated ability of the developed algorithm to identify object through segmentation. Trials of the second image have shown that existence of the undesirable boundaries is ignored and correct boundaries of the document are always obtained.

The developed algorithm successfully positioned page in the center of the image. It scaled images to the right dimensions, so the text occupied the entire image area. Moreover, the processed image appeared to have horizontally positioned lines of text in the document. In trials 3 and 6, text appeared to be turned upside down. In both of these trials, the θ if the original image was = 120^0. Both images were subjected to rotation of 420 degrees. In situations like this two, logic not allowing image to over-rotate or exceed certain degree of rotation could be introduced to the algorithm. Such logic could be achieved by implementation of validation check-points within existing solution or implementation of neural network responsible for making decisions. Image processing time significantly increased with increase of image complexity. Although, the equipment used to perform all six trials should be considered as a partial source of declining performance as well.

SUMMARY

The design of OCR preprocessor (Algorithm for Document Alignment) is presented. The implementation of the algorithm mainly concentrated on Global Processing via Hough

transform technique. Numerous functions provided by the Image Processing Toolbox were used in the current solution. Their usage was demonstrated and explored. Digital image diskewing (alignment) were achieved during all trials performed. Data used for image transformation was recorded.

A future study should concentrate on implementation of neural network able to identify text and evaluate its positioning. Such solution will enable precise text positioning either by algorithm itself or on demand. Further improvements should be explored to decrease processing time.

REFERENCES

[1] Rafael C. Gonzalez, Rechard E. Woods, Steven L. Eddins, "Digital Image Processing using MATLAB", Pearson *Prentice Hall*, 2004.

[2] Rafael C. Gonzalez. Richard E. Woods, "Digital Image Processing", Prentice-Hall, 2002