Angle slope detection on scanned documents

Valeriy Dmitriev
Ufa, Russia
ufabiz@gmail.com

Abstract

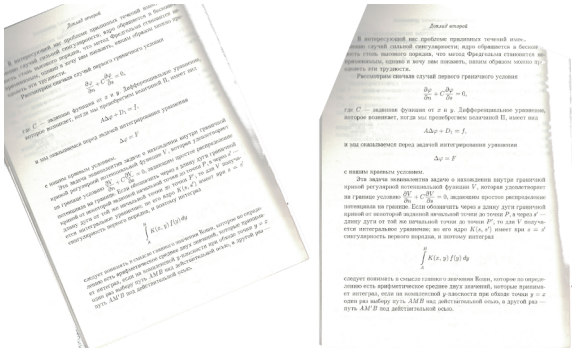
The quality of an optical text recognition depends on whether the text is sloped horizontally in the document or not. Aligned documents have noticeably better recognition quality. So there is a practical need to means of automatically aligning the slope of the text.

This paper proposes a simple, universal and quite effective text slope alignment algorithm based on the idea of minimizing the average entropy of rows and columns of bitmap images.

1 Idea

The basic algorithm idea is that if a text is rotated on a scanned image, the average entropy of pixel distribution by rows and columns will increase.

Let us consider a scan of a black-and-white image, where each pixel can take only two values: 0 or 1. As a known fact, the entropy of the uniform distribution is maximal. If an image is sloped, then the pixel distribution across rows (and columns) will on average be closer to uniform one than that of an unrotated image. For an aligned image, the pixel distribution is on average less uniform.



The hypothesis is to calculate the average entropy of pixel distribution by rows and columns for different rotation angles and then to find the angle at which the minimum value of the averaged entropy is reached.

In order to check the hypothesis, a dataset of various types of scanned images was collected from the Internet. Subsequently the assumption was proven experimentally.

The approach which was resorted to in experiment is efficient. It enables us to correctly determine the rotation angle in 83 % cases. And to determine the rotation angle with an error of 1° in 98 % cases.

Although the Shannon entropy is well suited for this problem, it would be reasonable to consider the whole spectrum of Rényi entropies. To choose the optimal parameter of the Renyi entropy, let us also take into account the computational complexity.

The Rényi entropy is defined with the following expression [1]:

$$R_\alpha = \frac{1}{1-\alpha} \log \left(\sum_{i=1}^{n} p_i^\alpha \right), \tag{1}$$

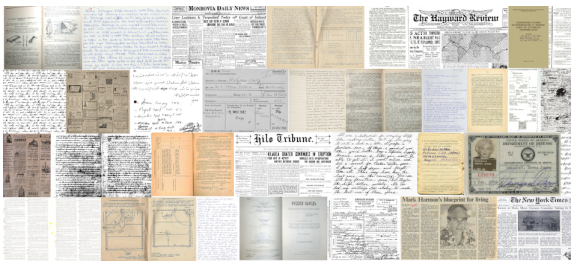
where p_i are the probabilities corresponding to the distribution. In our case, these are the frequencies of black and white pixels.

In the case of $\alpha = 1$, the expression (1) turns into the Shannon entropy:

$$R_1 = H = -\sum_{i=1}^{n} p_i \log(p_i). \tag{2}$$

2 Experiment

A set of various documents was collected on the Internet. Then each document of the set was turned by a random angle in the range from -45° to 45°. And finally the angle of slope was calculated using the algorithm proposed below.



The table below shows the results for various values of the Rényi entropy parameter α:

Rényi entropy parameter α	1/8	1/4,	1/2	3/4	1	2	5
Average absolute deviation	0.498	0.299	**0.211**	0.283	0.240	5.827	41.099
Part of total matches (accuracy)	0.822	**0.834**	0.828	0.815	0.805	0.641	0.009
Part of matches with an error no more then 1°	0.942	0.969	**0.980**	**0.982**	**0.983**	0.822	0.015
Part of matches with an error no more then 2°	0.967	0.983	**0.991**	**0.993**	**0.994**	0.841	0.015

A total of 1665 documents were processed.

It can be concluded from the table above that the smallest average absolute deviation is achieved with $\alpha = \frac{1}{2}$.

The best accuracy (proportion of total matches) is achieved with $\alpha = \frac{1}{4}$.

The best acceptable accuracy, as a proportion of matches with an error no more then 1°, is achieved with $\alpha \in \{1, \frac{3}{4}, \frac{1}{2}\}$.

And the best accuracy, as a proportion of matches with an error no more then 2°, is achieved with $\alpha \in \{1, \frac{3}{4}, \frac{1}{2}\}$ too.

In order to consider the proposed method as a part of the optical document recognition complex, the best value of the parameter α is $\alpha = \frac{1}{2}$.

With $\alpha = \frac{1}{2}$, the mean absolute deviation is only 0.211°. This achieves an optimal acceptable proportion of matches with an accuracy of 1°.

There is another reason to choice $\alpha = \frac{1}{2}$: this value achieves optimal computational complexity.

Below are the results of the multiple calculation benchmark of entropies (1) for various values of the parameter α [3].

α	nanoseconds	miliseconds	% of Shannon
1	10249895206	10249	100
1/2	8677368472	8677	**84.66**
1/4	10421639934	10421	120.1
1/8	13235709810	13235	127
3/4	11403406522	11403	86.16
2	7245386547	7245	**63.54**
5	7771674801	7771	107.26
10	10809162384	10809	139.08

The benchmark was calculated for 1000000000 samples [1].

The table above shows us that of the values of α that are suitable for us the best performance is achieved with $\alpha = 1/2$, which is quite expected.

3 Algorithm

Remark. The algorithm proposed below [2] assumes that we use a binary black-and-white bitmap image to determine the rotation angle, in which each pixel can take only two values: 0 or 1.

Therefore, first it is necessary to obtain a binarized copy of the image in order to apply the algorithm.

I have implemented the algorithm with the *libleptonica* library to manipulating bitmap images, because it is used in TesseractOCR [2].

For this I used sequential conversion pixContrastTRC with $contrast\_factor = 1.0$ and then pixConvertTo1 with $threshold = 170$.

Let h is the hight and w is the width of the original image.
And let $d = \sqrt{w^2 + h^2}$ is the diagonal length.

We will rotate the image by an angle ϕ relative to the center of the image and calculate the average entropy over rows and columns.

In order not to go beyond the boundaries of the image as a result of rotation, we mentally expand the canvas to a size with height and width equal to d.

Define the integers x_{from}, x_{to}, y_{from} and y_{to} as the following expressions:

$$\begin{aligned} x_{from} &= \frac{d}{2} - \frac{h|\sin(\phi)| + w|\cos(\phi)|}{2}, \\ x_{to} &= d - x_{from}, \\ y_{from} &= \frac{d}{2} - \frac{h|\cos(\phi)| + w|\sin(\phi)|}{2}, \\ y_{to} &= d - y_{from}, \end{aligned}$$

where x_{from} and x_{to} define the left and the right boundaries of the rotated image, respectively.

And y_{from} and y_{to} define the upper and the lower boundaries of the rotated image, respectively.

We need to calculate the average entropy by rows and columns. For example, let's show how to calculate by rows. The calculation by columns is similar.

Let $V(x, y)$ be the color of a point with coordinates (x, y) (takes values in $\{0, 1\}$).

And let $R(\{p, q\})$ be the entropy for the distribution $\{p, q\}$.

The algorithm below calculates the average entropy S_ϕ of a bitmap binary image rotated by an angle ϕ.

[1] The benchmark source code can be found on the page

Algorithm 1 Calculation of the average rows entropy

$S_\phi = 0$ ▷ The average rows entropy.
$y = y_{from}$
while $y < y_{to}$ **do**
  $b = 0$ ▷ The number of black pixels in a row.
  $x = x_{from}$
  while $x < x_{to}$ **do**
    $\widetilde{x} = x - \frac{d}{2}$ ▷ Setting the origin
    $\widetilde{y} = y - \frac{d}{2}$ ▷ to the canvas center.
    $x' = \widetilde{x} \cdot \cos(\phi) - \widetilde{y} \cdot \sin(\phi) + \frac{w}{2}$ ▷ Rotate the canvas by the angle ϕ .
    $y' = \widetilde{x} \cdot \sin(\phi) + \widetilde{y} \cdot \cos(\phi) + \frac{h}{2}$
    if $x' \geq 0$ AND $x' < w$ AND $y' \geq 0$ AND $y' < h$ **then**
      $b = b + V(x', y')$; ▷ Counting black pixels.
    end if
    $x = x + 1$
  end while
  $p = \frac{b}{d}, q = 1 - p$
  $S_\phi = S_\phi + \frac{R(\{p,q\})}{d}$
  $y = y + 1$
end while

The resulting value of S_ϕ will be the desired average entropy for a rotation angle ϕ.

To find the required rotation angle ϕ_0, we need to find the minimum of S_ϕ:

$$\phi_0 = -\arg\min_\phi S_\phi.$$

The "−" sign is taken because to align the image, we need to rotate it in the opposite direction.

References

[1] WikiPedia "Rényi entropy" https://en.wikipedia.org/wiki/Rényi_entropy

[2] The algorithm source code on C++ https://github.com/valmat/rotate_detection

[3] The entropy performance benchmark source code on C++ https://gist.github.com/valmat/6a737cc3783449c4f7a829e77c77393e

`https://gist.github.com/valmat/6a737cc3783449c4f7a829e77c77393e`

[2]The algorithm source code: `https://github.com/valmat/rotate_detection`