OCR MATHEMATICAL MODEL

The Tesseract Algorithm is described as follows:

1. **Line and word finding**

   The text line detection is designed for documents of varying orientations, backgrounds, and heterogenous layouts. Then the document is binarized to locate text elements. The binarization method is a scale-space adaptation of Su's method which can identify varying font sizes. Subsequently, the connected components are merge into "words" and then are clustered as text lines.

   Text line detection:

   The text line detection presented is a bottom-up approach which utilizes profile boxes of words as basic entity. These words are merged according to their minimal distance. Then, line rectangles are calculated using PCA for a robust line orientation estimation. A recti linearity hypothesis test removes words at either end of a text line that impair the result due to mutually inconsistent locations. Having located the text lines, a back-propagation voting is applied which joins local classification results with global knowledge of the current document. Thus, a single word classified as handwriting in the context of printed words may change its label to print depending on its classification accuracies.

   One assumption is incorporated into the text line detection:
   Each word is either the last word of a text line or it is succeeded by exactly one word. This applies to all horizontal and can be easily adapted to vertical writing systems. It implies that maximally two words are merged at the same time and noise in the text region (e.g. quotes or dots) are not merged with any text line. Furthermore, larger irregular spaces between words can be bridged resulting in fewer split errors.
   As previously mentioned, the inputs for the text line detection are labelled profile boxes which represent entities like words.
   First, a distance matrix containing connecting distances d(p, q) of all rectangles is created.

   Hence, the detection process is carried out in a rotationally invariant manner. In order to minimize the chance of false vertical merges, caused by low line spacing, the connecting distance of two rectangles is weighted by:

   $$d(p, q) = e(p, q) \cdot (1 + C\, g\sigma(\cos(\theta 1 - \theta 2))) \quad (2)$$

where e(p, q) is the minimal Euclidean distance between the rectangles' left p and right q points (upper, middle and lower).

θ1 is the angle of the left rectangle and θ2 the angle of the line which connects both rectangles.

gσ(x) is a Gaussian distribution with σ = 1/3 and C is a constant which weights the angle distance depending on the rectangle's class label (based on experiments, we chose C = 15 for printed and C = 4 for handwritten text). Figure 3 shows a sample image where the Euclidean distance would lead to a false word merging (dashed line), while the distance measure d(p, q) proposed correctly merges the words. Note that in this case "and" and "Ocean" are not merged anyway because of the global minimization. Having created the distance matrix, all words with a minimal distance are connected. The decision of connecting neighbouring words is based on the global distance minimization. In the end solely words at the beginning or end of a line and noise within the line are left.

In order to reject these connections, either a global threshold or an adaptive threshold calculated by means of robust statistics on the distance matrix can be applied.

The orientation of the text line rectangle is estimated using PCA (see Figure 4). First, the upper and lower line of all words merged to one text line are sampled equidistantly so that larger words have greater significance. The PCA is then computed for all sample points.

Finally, the line's orientation is determined by:

$$\theta l = \tan{-1}\,(e1, e2) \quad (3)$$

where θl represents the resulting text line angle and e1, e2 are the first and the second Eigenvectors respectively.

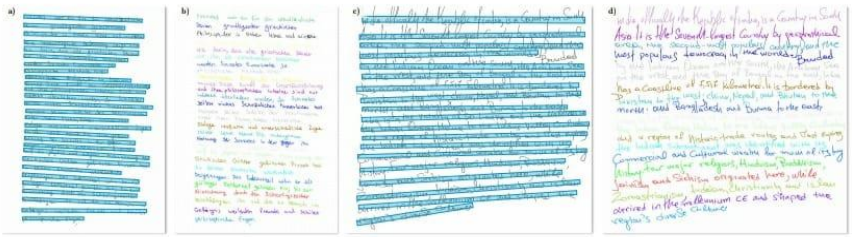


Figure 5. Sample images from the ICDAR 2009 (a, b) and ICFHR 2010 (c, d) datasets. The missing word *"Bounded"* in (c) gets corrected during labeling (d). The FM of (c) is 63.16% by reason of falsely labeled ascenders and descenders especially in the first four text lines (d).

2. Pitch estimation

three different approaches to estimate the pitch of a text line efficiently and reliably, even when several of the characters are broken or touching: autocorrelation, Fourier transform, and peak-valley analysis.

The Radon transform, the vertical projection profile f(x) of a text Img(x, y) is defined as:

$$f(x) = \sum_{y} Img(x, y).$$

For pitch-based segmentation, one also needs to know the offset of the text, as well as its pitch. the offset is defined to be the distance between the left margin and the left edge of the imaginary box in which the first character is written. These imaginary fixed-width boxes, starting from the offset, will be called the pitch windows. The Radon transform of a fixed-pitch word with pitch p is roughly periodic with period p, though this is less true for the whole text where there are spaces or small punctuation characters. By applying the limiting transformation, we can remove some of the irrelevant information and bring out the periodicity further. This limiting transformation also reduces the adverse effect of large noise regions such as vertical frames, and has been found useful in increasing the overall pitch estimation accuracy. we calculate a rough initial estimate of the pitch, from the initial connected component analysis. This value is used for estimating a few non-sensitive threshold values, as well as narrowing down the possible pitch range in the pitch estimation process.

Fig. 1. Sample fixed-pitch text

Fig. 2. Sample proportional-font text

Fig. 3. Sample dot-matrix text

Fourier Transform approach:

Due to the roughly periodic nature of the Radon transform of a fixed-pitch text, we expect that, in the Fourier representation, the sinusoids which have a period equal to the pitch, would correlate highly with the Radon transform. To find the sinusoids with the highest correlation, we need to find the magnitude of the corresponding complex coefficients. Using the Discrete Fourier Transform (DFT), the Fourier coefficients an and bn of the sinusoids can be calculated as:

$$an = (f(x) \times \cos(2\pi n/Nx))$$

$$bn = (f(x) \times \sin(2\pi n/Nx))$$

where N is the number of data points and the index n corresponds to the pitch equalling N/n. From these, we can estimate the magnitude of the complex coefficients cn as:

$$| cn | = Math.pow((an2 + bn2), 1/2)$$

Finally, the pitch is computed as $N/n*$ where N is the number of data points and $n*$ is the index of the coefficient with the maximum magnitude. Without using an FFT implementation, the

time necessary to estimate the pitch, as described above, is on the order of P N, where N is the length of the line and P is the number of possible pitches considered. However, the trigonometric function evaluations are more time consuming than the calculations used in the auto-correlation method. The offset, on the other hand, is calculated in a single step, as:

arctan bn∗/an∗

The performance of the Fourier analysis for estimating the pitch was similar to the auto-correlation method. One type of error, also common to the peak-valley analysis, occurs when there are too few characters on the text line and most of them have dominating side strokes (e.g., “H, N, U”) and/or broken centers, resulting in an incorrect pitch estimate that is roughly half the correct pitch.

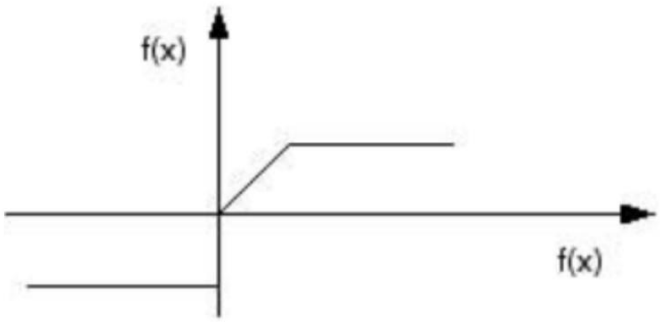
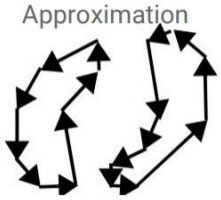


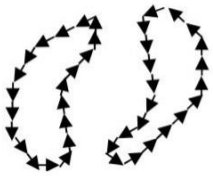
Fig. 7. Limiting function applied to the Radon transform, $f(x)$

3. Feature detection and extraction, Classification of characters

Segments of the polygonal Approximation



Even smaller features

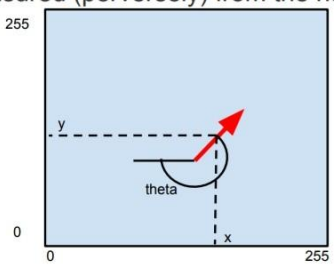


Statistical: $argmax(k)\prod_{l,i}\frac{1}{\sigma_{ijk}}\exp[-\frac{1}{2}(\frac{x_{il}-\mu_{ijk}}{\sigma_{ijk}})^2]$

Geometric: $argmin(k)\frac{1}{M+J_k}(\sum_{l,i}(x_{il}-\mu_{ijk})^2+\sum_{j,i}(x_{il}-\mu_{ijk})^2)$

Features from unknown

- Multiple features extracted from a single unknown
- Each feature is a short, **fixed length**, directed, line segment, with (x,y) position and theta direction making a 3-D feature vector (x, y, theta) from an integer space [0, 255]
- Direction is measured (perversely) from the negative x-axis!

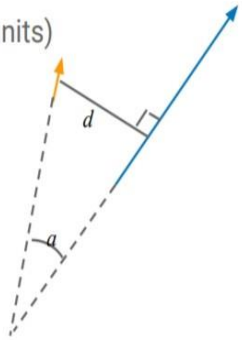


Features in training data

- Elements of the polygonal approximation, clustered within a character/font combination.
- x,y position, direction, **and length** (as a multiple of feature length)

Distance Function: Single Feature to Single Prototype

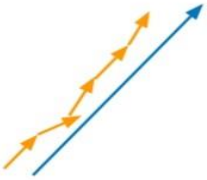
d = perpendicular distance of feature f from proto p
a = angle between feature f and proto p
Feature distance $d_{fp} = d^2 + a^2$ (in appropriate units)
Feature evidence $e_{fp} = 1^* / (1 + kd_{fp}^2)$



Feature evidence and Proto evidence

$$\text{Feature evidence } e_f = \max_{\text{p in config}} e_{fp}$$

$$\text{Proto evidence } e_p = \sum_{\text{top } l_p} e_{fp} \quad (\text{Proto p is of length } l_p)$$



Distance Function: Unknown char to Prototype

$$d = 1 - \max \frac{\sum_f e_f + \sum_p e_p}{N_f + \sum_p l_p} \qquad d' = \frac{dl_o + kc}{l_o + k}$$

Feature-proto distance *CN correction*

l_o = Length of outline
c = Char position feature distance (CN feature)
k = classify_integer_matcher_multiplier (arbitrary constant = 10)

Rating = $d'l_o$
Certainty = $-20d'$

Character Classifier