

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/315834331>

# Overview of Tesseract OCR engine

**Technical Report** · December 2016

CITATIONS

0

READS

825

**1 author:**



[Akhil s Nair](#)

National Institute of Technology Calicut

**2** PUBLICATIONS **0** CITATIONS

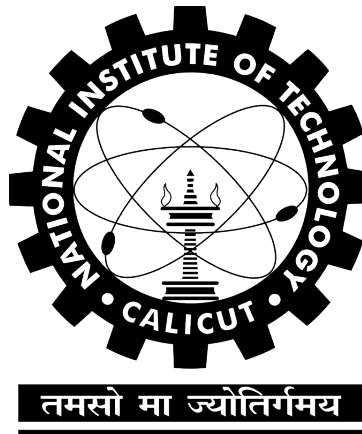
SEE PROFILE

# An overview of Tesseract OCR Engine

A  
Seminar Report

*by*

Akhil S  
B130625CS



Department of Computer Science and Engineering  
National Institute of Technology, Calicut  
Monsoon-2016

National Institute of Technology, Calicut  
Department of Computer Science and Engineering

*Certified that this Seminar Report entitled*

**An overview of Tesseract OCR Engine**

*is a bonafide record of the Seminar presented by*

**Akhil S**  
**B130625CS**

*in partial fulfillment of  
the requirements for the award of the degree of  
Bachelor in Technology  
in  
Computer Science and Engineering*

---

**T.A Sumesh**  
*Department of Computer Science and Engineering*

## **Acknowledgement**

I would like to express my special gratitude towards my seminar coordinator, Mr Sumesh T A , for giving me the opportunity to do this seminar on Optical Character Recognition.

This is seminar helped me to increase my knowledge in the field of Computer Vision and new trends in the area of Character Recognition. I was able to explore a lot on these two topics going through various research papers and experiments.

I would also like to thank the Head of the Department of Computer Science department Dr Saidalavy Kalady for his guidance and support.

## **Abstract**

Machine replication of human functions, like reading, is an ancient dream. Optical character recognition is the machine replication of human reading and has been the subject of intensive research for more than three decades. It can be described as Mechanical or electronic conversion of scanned images where images can be hand written, type-written or printed text. This paper presents Google's open source Optical Character Recognition software Tesseract. We will give an overview of the algorithms used in the various stages in the pipeline of Tesseract. In particular we will focus on the aspects that are novel or at least unusual in Tesseract compared to other OCR engines.

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Architecture</b>	<b>6</b>
<b>3</b>	<b>Adaptive Thresholding</b>	<b>6</b>
<b>4</b>	<b>Page Layout Analysis</b>	<b>7</b>
<b>5</b>	<b>Baseline Fitting and Word Detection</b>	<b>8</b>
<b>6</b>	<b>Word Recognition</b>	<b>9</b>
<b>7</b>	<b>Character Classifier</b>	<b>11</b>
7.1	Static Classification . . . . .	11
7.2	Adaptive Classification . . . . .	12
<b>8</b>	<b>Demonstration</b>	<b>12</b>

# 1 Introduction

Optical character recognition is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text [8]. It is the easiest method of digitizing printed and handwritten texts so that they can be easily searched, stored more compactly, displayed and edited online, and used in various other processing tasks such as language translation and text mining.

Tesseract is an open source optical character recognition engine. Tesseract began as a PhD research project in HP Labs, Bristol. It was further developed at HP in between 1984 to 1994 [4]. It was modified and improved in 1995 with greater accuracy. In late 2005, HP released Tesseract for open source and is now available at [2].

The first part of this paper presents the overview of the architecture of Tesseract. In the later sections we will see how the major stages in the Character Recognition problem - preprocessing, segmentation and recognition is dealt by Tesseract. The final part presents the usage and a short demonstration of the software.

## 2 Architecture

The pipeline of Tesseract OCR engine is given in fig1.

The first step is Adaptive Thresholding, which converts the image into a binary version using Otsu's method [3]. The next step is page layout analysis, which is used to extract the text blocks within the document as shown in fig 2. In the next stage the baselines of each line are detected and the text is divided into words using definite spaces and fuzzy spaces [4].

In the next step, the character outlines are extracted from the words. Recognition of text is then started as two-pass process. In the first pass, word recognition is done using the static classifier. Each word passed satisfactory is passed to an adaptive classifier as training data [4]. A second pass is run over the page, using the newly learned adaptive classifier in which words that were not recognized well enough are recognized again.

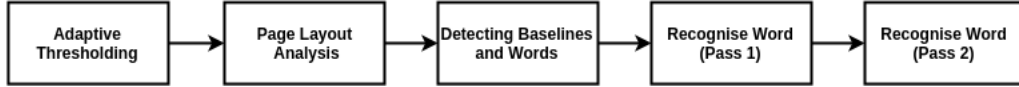


Figure 1: Architecture of Tesseract OCR engine.

## 3 Adaptive Thresholding

In tesseract, Otsu's method [3] is used to perform clustering-based image thresholding. The pixels in the picture are represented in L gray levels [0,1, .. ,L] where each value corresponds to a potential threshold.

In Otsu's method we search for the threshold that minimizes the intra-class variance, defined as a weighted sum of variances of the two classes:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Weights  $\omega_0$  and  $\omega_1$  are the probabilities of the two color classes separated by a threshold t and square and  $\sigma_0^2$  and  $\sigma_1^2$  are variances of these two classes.

Otsu shows that minimizing the intra-class variance is the same as maximizing inter-class variance [3]

$$\sigma_b^2(t) = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2$$



where

$$\mu_0(t) = \sum_{i=0}^{t-1} \frac{p(i)}{\omega_0(t)}$$

$$\mu_1(t) = \sum_{i=t}^{L-1} \frac{p(i)}{\omega_1(t)}$$

All these values can be computed from the binary grayscale input given. The optimal threshold  $t^*$  that maximizes  $\sigma_b^2(t)$ , is selected by a sequential search over the different values of  $t$ .

To account for variations within the image, Local Adaptive thresholding is performed in tesseract, where Otsu's algorithm is applied to small sized rectangular divisions of the image.

## 4 Page Layout Analysis

Page layout analysis, one of the first steps of OCR, divides an image into areas of text and non-text, as well as splitting multi-column text into columns [7].

The page layout analysis in Tesseract is based on the idea of detecting tab-stops in a document image. It has the following steps[7]:

1. The morphological processing from Leptonica[1] detects the vertical lines and the images. These elements are removed from the input image before passing the cleaned image to connected component analysis.
2. The candidate tab-stop connected components that look like they may be at the edge of a text region are found and then grouped into tab-stop lines.
3. Scanning the connected components from left to right and top to bottom, runs of similarly classified connected components are gathered into Column Partitions (CP), subject to the constraint that no CP may cross a tab stop line.
4. Chains of text CPs are further divided into groups of uniform line-spacing, which make text blocks. Now each chain of CPs represents a candidate region. The reading order of the regions are determined by some heuristic rules.



Figure 2: Output of Page Layout Analysis.

## 5 Baseline Fitting and Word Detection

Tesseract uses a very novel algorithm for finding the rows of text on a page. The algorithm performs well even in the presence of broken and joined characters, speckle noise and page skew. It operates as follows[5]:

1. Connected Component Analysis is performed.
2. The median height is used approximate the text size in the region and components (blobs) that are smaller than some fraction of the median height mostly being punctuation, diacritical marks and noise are filtered out.
3. The blobs are sorted (into ascending order) using x-coordinate (of the left edge) as the sort key. This sort makes it possible to track the skew across the page.
4. For each blob in sorted order:  
Find the existing row which has most vertical overlap with the blob.

```

    If there is no overlapping row
    Then
        Make a new row and put the blob in it.

        Record the top and bottom
        coordinates of the blob as the
        top and bottom of the row.
    Else
        Add the blob to the row.

        Expand the top and bottom limits of
        the row with the top and bottom of
        the blob, clipping the row height to a limit.
    Endif

Endfor

```

Once the text lines have been found, the baselines are fitted more precisely using a quadratic spline by a least squares fit.

Tesseract does word detection by measuring gaps in a limited vertical range between the baseline and mean line[4]. Spaces that are close to the threshold at this stage are made fuzzy, so that a final decision can be made after word recognition.

## 6 Word Recognition

In this part the detected words are segmented into characters.

Tesseract tests the text lines to determine whether they are fixed pitch (having a constant spacing between word and characters) during word detection step. For the fixed pitch text, Tesseract chops the words into characters using the pitch. The rest of the word recognition step applies only to non-fixed-pitch text.

Fig.3 is a block diagram of the word recognizer [6].

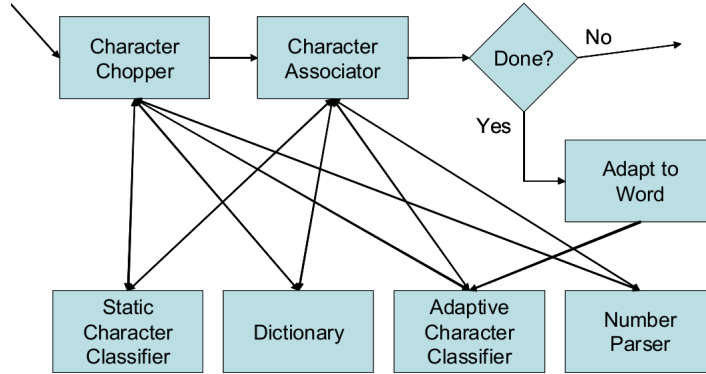


Figure 3: Block diagram of Word Recogniser.

The word recognizer first classifies each blob, and presents the results to a dictionary search to find a word in the combinations of classifier choices for each blob in the word. While the word result is unsatisfactory, Tesseract chops the blob with worst confidence from the character classifier.

Candidate chop points(fig.4 [4]) are found from concave vertices of a polygonal approximation of the outline.

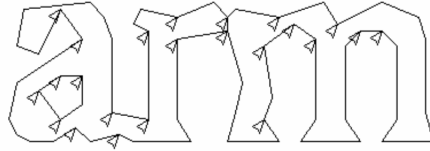


Figure 4: Candidate chop points.

After the chopping possibilities are exhausted, the associator makes an A\* (best first) search of the segmentation graph of possible combinations of the maximally chopped blobs into candidate characters [4]. At each step in the best-first search, any new blob combinations are classified, and the classifier results are given to the dictionary again.

The output for a word is the character string present in the dictionary that had the best overall distance-based rating.

## 7 Character Classifier

### 7.1 Static Classification

The features used in classification are the components of the polygonal approximation of the outline of a shape. In training, a 4-dimensional feature vector of (x, y-position, direction, length) is derived from each element of the polygonal approximation, and clustered to form prototypical feature vectors (Hence the name: Tesseract) [6].

In recognition, the elements of the polygon are broken into shorter pieces of equal length, so that the length dimension is eliminated from the feature vector. This process of small features matching large prototypes is easily able to cope with recognition of damaged images. Its main problem is that the computational cost of computing the distance between an unknown and a prototype is very high.

To reduce the computing time, in the first step of classification, a class pruner creates a shortlist of 1-10 character classes that the unknown might match using a method closely related to Locality Sensitive Hashing (LSH).

In the second stage, the classifier calculates the weighted distance  $d_f$  of each feature from its nearest prototype as follows [6]:

$$d_f = d^2 + w\theta^2$$

Where  $d$  is the Euclidean distance of the feature coordinates from the prototype line and  $\theta$  is difference of the angle from the prototype. The feature distance is converted to feature evidence  $E_f$  using the following equation [6]:

$$E_f = \frac{1}{1 + kd_f^2}$$

The constant  $k$  controls the rate at which the evidence decays with distance.

As features are matched to prototypes, the feature evidence  $E_f$ , is copied to the prototypes  $E_p$ . The sums are normalized by the number of features and sum of prototype lengths  $L_p$ , and the result is converted back into a distance [6]:

$$d_{final} = 1 - \frac{\sum_f E_f + \sum_p E_p}{N_f + \sum_p L_p}$$

The computed final distance is used by KNN algorithm for classification.

## 7.2 Adaptive Classification

A more font-sensitive adaptive classifier that is trained by the output of the static classifier is used to obtain greater discrimination within each document, where the number of fonts is limited [4]. Tesseract uses the same features and classifier of the static classifier for Adaptive classification.

Since the adaptive classifier learns during the first run, it can only make significantly less contribution near the top of the page if deployed during the first run. Therefore, a second pass is run over the page, in which words that were not recognized well enough in the first run are recognized again.

## 8 Demonstration

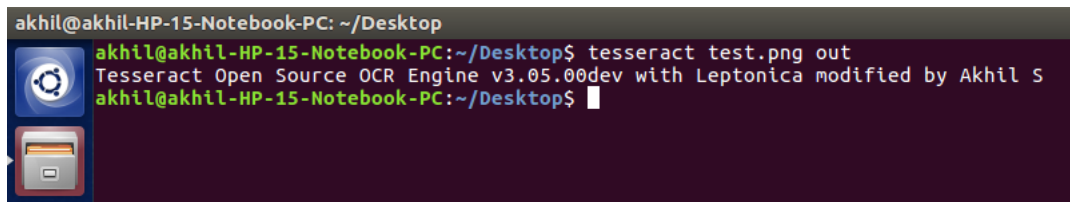
An image with the text (fig.5) is given as input to the Tesseract engine

### 6. CONCLUSION

Although Tesseract is command-based tool but as it is open source and it is available in the form of Dynamic Link Library, it can be easily made available in graphics mode. The results obtained in above sections are obtained by extracting vehicle number from vehicle number plate. So above results do not confirm that Tesseract is always better or faster than Transym but is it more accurate in extracting text from the vehicle number plate. The input images are specific, which are vehicle number plates, so in these specific images Tesseract provides better accuracy and in other kinds on images Transym might provide better accuracy than Tesseract. As we are interested in extracting vehicle number from vehicle number plate, we have considered both tools for serving this specific purpose.

Figure 5: Test Image

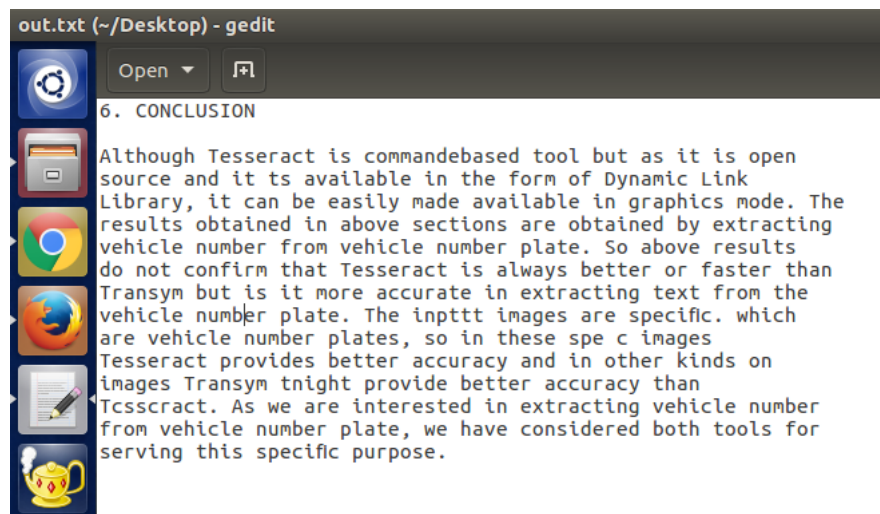
Tesseract command takes two arguments: First argument is image file name that contains text and second argument is output text file in which, extracted text is stored (fig.6).

A terminal window with a dark background. The prompt is 'akhil@akhil-HP-15-Notebook-PC: ~/Desktop'. The command 'tesseract test.png out' has been entered. The output shows 'Tesseract Open Source OCR Engine v3.05.00dev with Leptonica modified by Akhil S' followed by a new line.

```
akhil@akhil-HP-15-Notebook-PC: ~/Desktop
akhil@akhil-HP-15-Notebook-PC:~/Desktop$ tesseract test.png out
Tesseract Open Source OCR Engine v3.05.00dev with Leptonica modified by Akhil S
akhil@akhil-HP-15-Notebook-PC:~/Desktop$
```

Figure 6: Running the Tesseract engine via Terminal

Fig.7 shows the output in the file out.txt .

A Gedit window titled 'out.txt (~/Desktop) - gedit'. It shows the text of a conclusion section. The text discusses the accuracy of Tesseract compared to Transym for vehicle number plate extraction.

```
out.txt (~/Desktop) - gedit
6. CONCLUSION

Although Tesseract is commandbased tool but as it is open
source and it ts available in the form of Dynamic Link
Library, it can be easily made available in graphics mode. The
results obtained in above sections are obtained by extracting
vehicle number from vehicle number plate. So above results
do not confirm that Tesseract is always better or faster than
Transym but is it more accurate in extracting text from the
vehicle number plate. The inpttt images are specific. which
are vehicle number plates, so in these spe c images
Tesseract provides better accuracy and in other kinds on
images Transym tnight provide better accuracy than
Tcsscract. As we are interested in extracting vehicle number
from vehicle number plate, we have considered both tools for
serving this specific purpose.
```

Figure 7: Output of processing done by Tesseract on fig.5

## References

- [1] Leptonica image processing and analysis library. <http://www.leptonica.com>.
- [2] Tesseract ocr, github:. <https://github.com/tesseract-ocr>.
- [3] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

- [4] R Smith. An overview of the tesseract ocr engine. in proceedings of document analysis and recognition.. icdar 2007. In *IEEE Ninth International Conference*, 2007.
- [5] Ray Smith. A simple and efficient skew detection algorithm via text row accumulation. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 1145–1148. IEEE, 1995.
- [6] Ray Smith, Daria Antonova, and Dar-Shyang Lee. Adapting the tesseract open source ocr engine for multilingual ocr. In *Proceedings of the International Workshop on Multilingual OCR*, page 1. ACM, 2009.
- [7] Raymond W Smith. Hybrid page layout analysis via tab-stop detection. In *2009 10th International Conference on Document Analysis and Recognition*, pages 241–245. IEEE, 2009.
- [8] Wikipedia. Optical character recognition. [https://en.wikipedia.org/wiki/Optical\\_character\\_recognition](https://en.wikipedia.org/wiki/Optical_character_recognition).