

maatras clipping algorithm for adding Indic script support to the Tesseract engine

7th June 2008

by Debayan Banerjee

Background: The Tesseract OCR engine currently does not recognise connected script such as Hindi, Bengali, and all such similar devnagri script. This algorithm aims to add Indic script support to the existing OCR engine by trying to modify the input image suitably and then submitting it to the existing Tesseract engine for segmentation, recognition etc.

Modus Operandi: The Tesseract OCR engine currently segments individual characters and then puts a bounding box around them. The engine depends on the characters being isolated 'blobs' of black pixels to be able to box them. Devnagri script fails to be segmented by the engine because entire words in this script are connected by what we call a *maatras* (Fig. 2).

A simple way to get by this problem would be to clip the *maatras* between successive characters, so that the connectedness is broken, and I assumes characteristics of scripts the existing Tesseract engine can recognise.

Input to the algorithm: The thresholded, unskewed and noise free image to be OCRed.

Output of the algorithm: The *maatras* clipped image.

Algorithm

- Start from 0,0 (Fig 1). Count the number of black pixels in each row of the image.
- If number of black pixels is greater than a certain number ($.05 * \text{width of page}$), store the corresponding y co-ordinate and the black pixel count in an array named 'blackpixels[][]'.
- After the entire page has been scanned in this fashion, process 'blackpixels[][]' to find rows which contain the *maatras* (Fig 2).
- The *maatras* is found in this way: Between two successive regions of zero-black-pixel-count rows (Fig. 3), the row with the highest number of black pixels is the row going through the *maatras*.
- Once a *maatras* is found store it in array 'maatras[][]'. Store the y co-ordinate, the black pixel count of the row, and the maximum character height encountered in that sentence. Do this for the entire 'blackpixels[][]' array.
- Now go through the 'maatras[][]' array. We need to find x co-ordinates for corresponding y co-ordinates in 'maatras[][]' so we can clip the *maatras* at that position.
- For each y co-ordinate (*maatras*) in the *maatras* array, look for the abscissa under which there is a considerable ($>.8 * \text{character_height}$) run of white pixels (Fig. 4). This means that below this spot in the *maatras*, there is no character and hence this is a region between two characters. We now clip the *maatras* at all such positions.

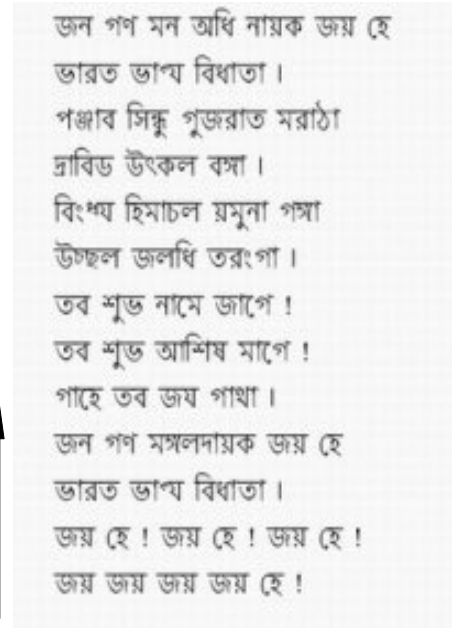


Fig. 1

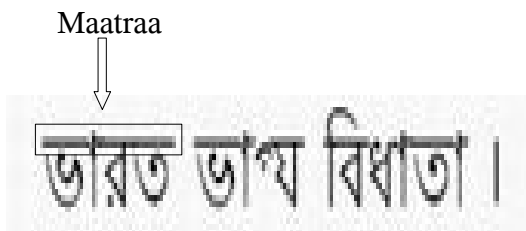
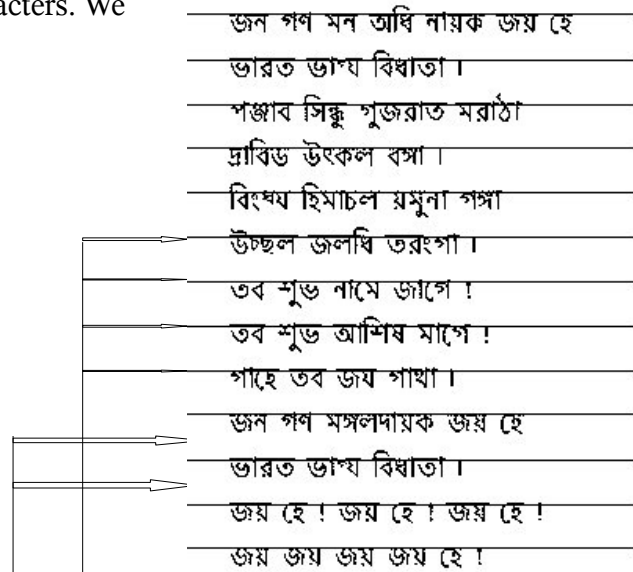


Fig. 2

Two successive regions of zero-pixel-count rows



A few rows with mataraas Fig. 3

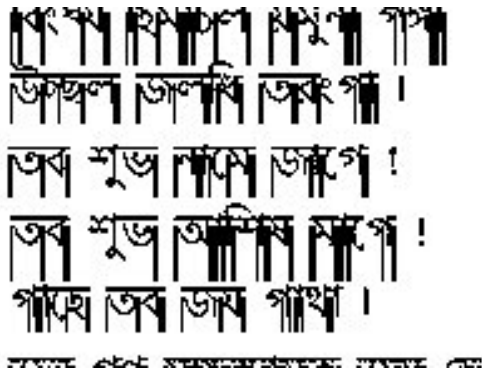


Fig. 5

In the image above, the black columns signify abscissae where we can remove the portion of the *maatras* from the top. The two circled regions in the image show an anomaly. It occurs because the loop below the first character overlaps the ordinate of the second character.

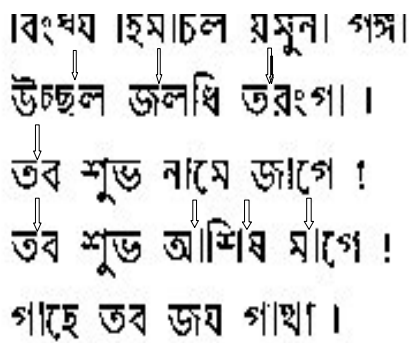


Fig. 6

The image above is the final output of the algorithm. Some of the clipped portion of the *maatras* are showed by small arrows. Compare with Fig. 5 and observe the results.

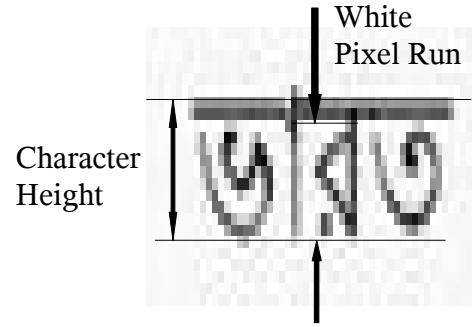


Fig. 4