Automated Amharic Braille Recognition Using Image Processing and Contour Detection

Amir Ahmedin, Amir Fasil, Tsehay Goremes, Henok Belayneh, Tamiru Alemnew

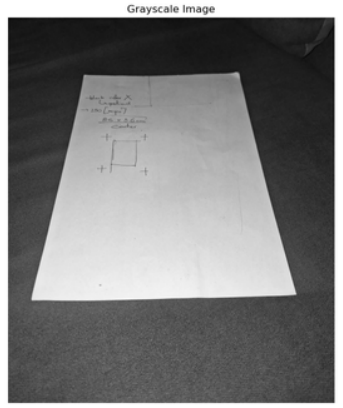
Advisor: Surafel Kindu

PROBLEM STATEMENT

Amharic Braille is essential for visually impaired individuals who read and write in Amharic. This project focuses on developing a computer vision-based solution to automatically detect and interpret Amharic Braille characters.

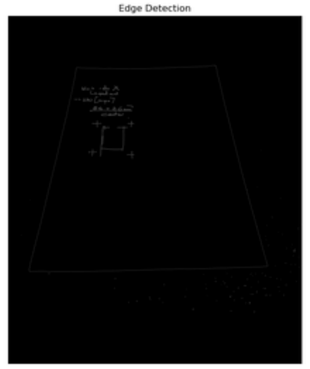
Step 1: IMAGE PROCESSING

Convert the Image to Grayscale

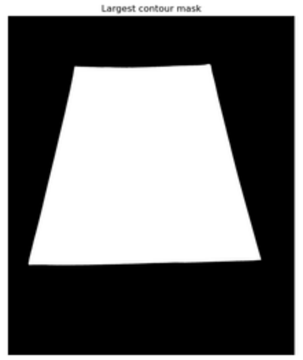


Grayscale Image

Apply Bilateral Filtering, Find Contours in the Image, Identify the Largest Contour & Create a Mask



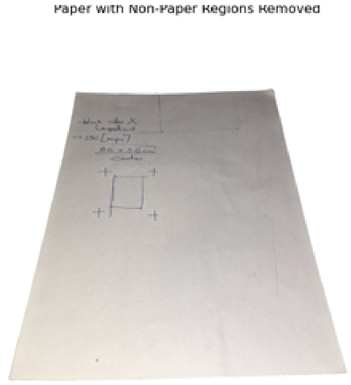
Edge Detection



Largest contour mask

Replace Non-Paper Areas with Zero:

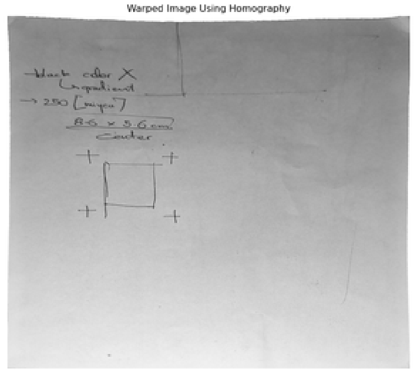
The areas outside the paper boundary are set to zero using the mask. This step removes unwanted background noise.



Paper with Non-Paper Regions Removed

Apply the Homography Matrix Using warpPerspective:

The homography matrix is used to transform the image using OpenCV's warpPerspective function, resulting in a deskewed and aligned Braille image.



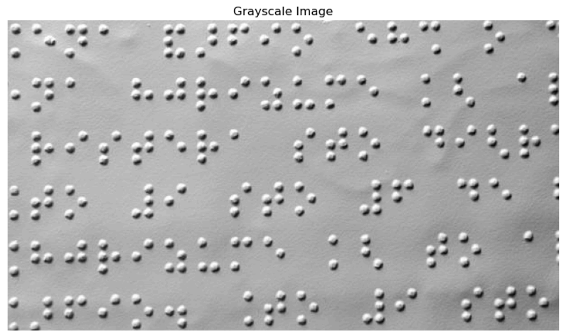
Warped Image Using Homography

Step 2: BRAILLE DOT DETECTION

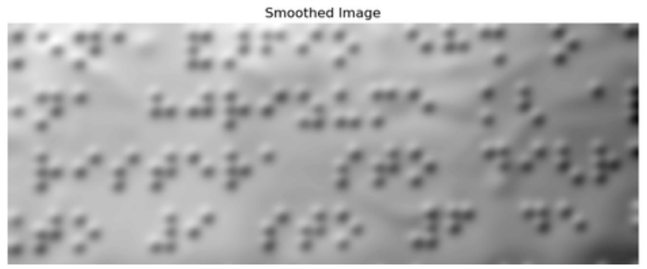
2.1. Smoothing the image:

Formula

Smoothed Image = GaussianBlur(I , ksize=(5,5), σ=1.5)



Grayscale Image

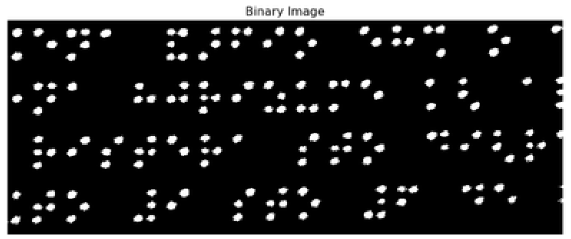


Smoothed Image

2.2. Binary Image Conversion:

Formula

AdaptiveThreshold(I, blockSize=11, C=2)

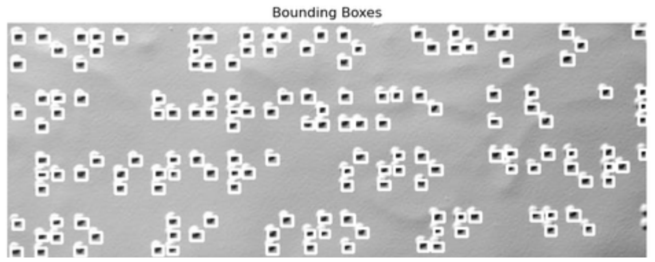


Binary Image

2.3. Contour Detection and Filtering:

Code Snippet

```
dot_contours, _ = cv2.findContours(binary,
cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```



Bounding Boxes

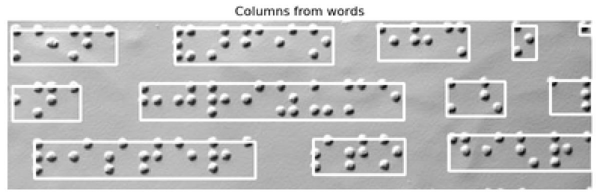
2.4. Grouping Contours:

Thresholds:

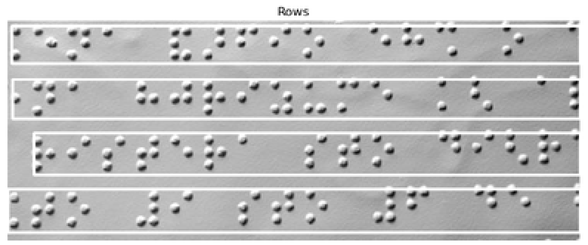
dot_spacing_threshold = 20

letter_spacing_threshold = 20

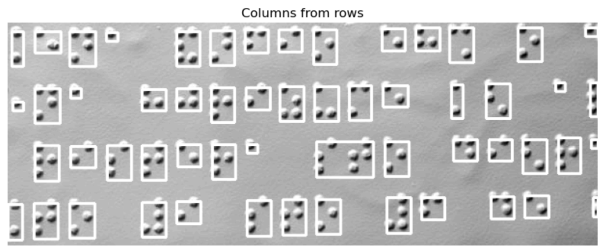
word_spacing_threshold = 50



Columns from words



Rows



Columns from rows

Step 3: Braille Grid Formation and Matching

3.1. Forming the grids:

Detected dots mapped to a **2x3 grid** for Amharic Braille.

Normalization formula:

col = int(grid_col_size * (x - min_x) / width)

row = int(grid_row_size * (y - min_y) / height)

Example:

Input (letter bounding boxes): [(4, 35, 12, 10), (5, 9, 11, 10)]

Output: [[1 0] [0 0] [1 0]]

3.2. Matching with Amharic Braille Dictionary:

Each formed grid is compared with a predefined dictionary of Amharic Braille patterns to determine the corresponding character.

Amharic Dictionary Example:

```
amharic_braille_dict = {
“:.” : { ":." :  'ሀ', "::" : 'ሁ',  ".:" : 'ሂ', ".." : 'ሃ',  ":." : 'ሄ', "::" : 'ህ', ".:" : 'ሆ' }
}
```

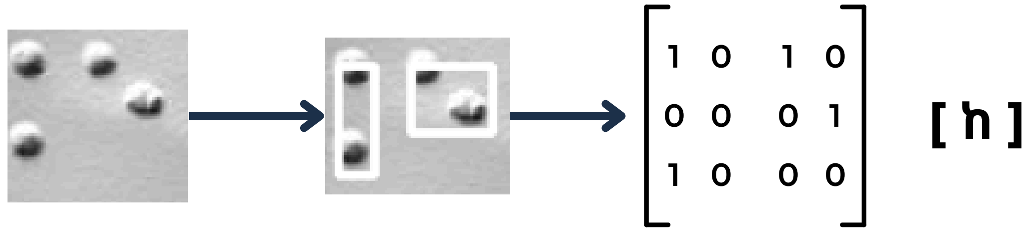
Example:

Input (letter bounding boxes):

[(4, 35, 12, 10), (5, 9, 11, 10)] + [(28, 9, 12, 9), (43, 22, 12, 9)]

Output Grid: [[1 0] [0 0] [1 0]] + [[1 0] [0 1] [0 0]]

Output Alphabet: [ከ]



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

[ከ]

Conclusion and Future Work

This work demonstrates an efficient pipeline for recognizing Amharic Braille characters using computer vision.

Future improvements include handling multi-line Braille documents, improving robustness for noisy or skewed inputs, and supporting real-time recognition.