CONTOUR BASED OPTICAL BRAILLE RECOGNITION FOR SINGLE SIDED ENGLISH BRAILLE TEXT

Divya Antony J.R.

March 2020

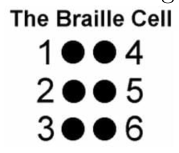
1 Introduction

Optical Braille Recognition(OBR) is the process or task of recognizing braille text from a simple image. It is analogous to **Optical Character Recognition(OCR)**. For this problem we propose simple image processing instead of using complex structures like *Artificial Neural Networks*. Some studies have used ANNs but we see this unnecessary since single sided english braille is not something like handwritten digits which is very distinct, It has some deformities due to wear and tear but braille characters most probably will not be handwritten and it also has a standard format to follow.

Thus we use shopisticated image processing and contours to decode the braille text. We see **OpenCV** is the best fit for implementing the proposed idea.

2 Anatomy of Braille Text

Figure 1: The structure of a single braille character



Braille is a tactile writing system used by people who are visually impaired. It is traditionally written with embossed paper. These characters have **rectangular blocks called cells that have tiny bumps called raised dots**. The number and arrangement of these dots distinguish one character from another.

Since we are only looking at **English Braille**, the braille characters are very predictable. A single braille character can be contained in a **3x2 Matrix**, (i.e) A single braille character in English braille text is a grid of 3 rows and 2 columns. This gives an opertunity to make the segmentation and processing optimal.

It is also worth to note that in our problem of recognizing English braille text, the first column of the braille in a single braille character will have atleast one dot present. This makes the segmenting of the braille character much more accurate

Thus a braille text contains a collection of braille cells as shown in figure 1 to form the entire text.

3 Preprocessing of the Braille Text Image

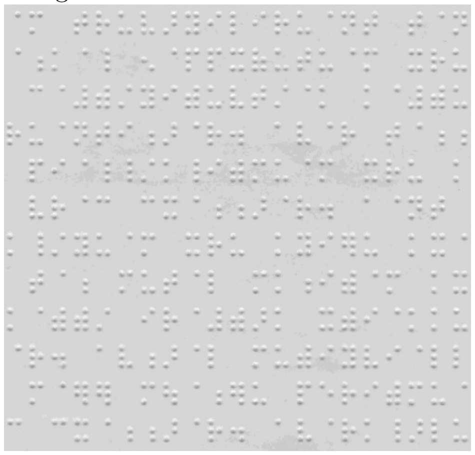
First we do various image morphing techniques to reduce noise and increase sharpness of the dots in the braille text, We do this by the following steps,

1. Conversion of **BGR/RGB scale to Gray scale**.
2. Apply segmentation using **Binary Thresholding and OTSU Thresholding**.
3. Remove Noise using **Blurring**
4. Repeat the above process for **3 iterations**.
5. Convert the result of the final thresholding to **binary image**.

Now with the final binary image we find contours.

3.1 Conversion to Gray scale

Figure 2: A image of Braille Text converted from RGB to Gray



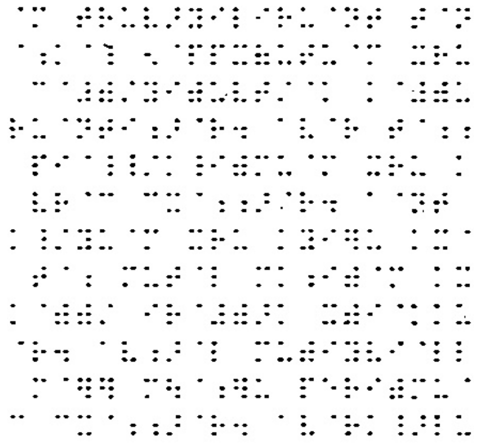
Using the following equation we convert the **RGB/BGR** to Gray channel.

$$Y = (0.299 * R) + (0.587 * G) + (0.114 * B) \tag{1}$$

Here **Y** is the Gray channel which can be used to construct the Gray scale image.

3.2 Segmentation using Binary Thresholding and OTSU Thresholding

Figure 3: Binary Thresholding of Braille Text



Thresholding is the **simplest form of segmentation**. Segmentation is the process in which different regions of a image is identified. By identifying different regions, we can effectively find **the impressed dots in the image by segmenting it from the background**.

In our proposed solution we use a combination of *Binary Thresholding* and *OTSU Thresholding (invented by Nobuyuki Otsu).*

$$Destination_{(x,y)} = \begin{cases} maxval & : Source(x,y) > T(x,y) \\ 0 & : Otherwise \end{cases} \tag{2}$$

Where **T(x,y)** is the threshold of the pixel at x,y with respect to 2d cartesian plane.

Equation **(2)** is used for binary thresholding.

The Otsu's algorithm tries to find a threshold value (t) which minimizes the **weighted within-class variance** given by the relation in equation **(3)** below,

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \tag{3}$$

Where,

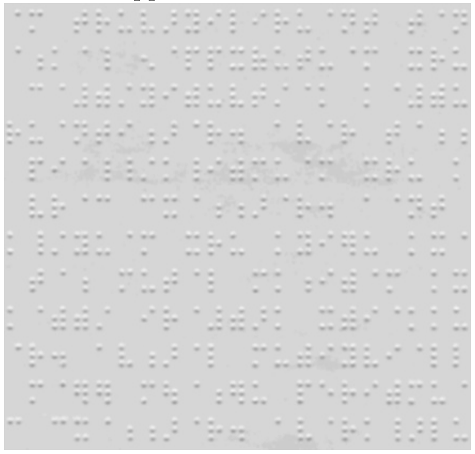
$$q_1(t) = \sum_{i=1}^{t} P(i) \quad \& \quad q_2(t) = \sum_{i=t+1}^{I} P(i)$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

3.3 Removing Noise

Figure 4: Gaussian blur applied to the Braille Text Gray Scale image



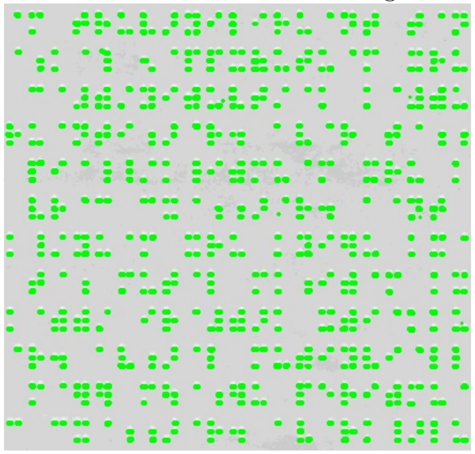
Using **Gaussian Blurring and Median Blurring** we reduce noise in the captured image as much as possible.

3.4 Conversion to Binary Image

Finally after all the processing, the final thresholded image is now converted to a binary image. (i.e) The image now has only two possible values for a single pixel.

A high value represents a white pixel and a low value represents a black pixel

Figure 5: Contours drawn on the original image



4 Contours

Contours are simple curve joining all the continuous points (along the boundary), having same color or intensity. The contours are useful tool for shape analysis and object detection and recognition. Contours can be extracted from the binary image which was obtained from the preprocessing step.

We find contours by a method proposed by **Satoshi Suzuki et el**. This method is already implemented in **OpenCV** so we need not define the entire algorithm as it exceeds the scope of this study.

Once we find the contours we do the **Braille Character Segmentation**.

5 Braille Character Segmentation

Segmentation is very simply built and unique to our study. The main feature about this segmentation is that this research can be further improved with Artificial Neural Networks if desired as this segmentation allows us to separate individual braille character with all the pixel data required for Convolutional Neural Networks.

The Segmentation process is as follows,

1. Calculate the **Mean Radius and Diameter** of each dots in each Braille Character using the Contours.

2. Using the Radius and Diameter, Find the **Center Point** of all dots in each Braille Characters.

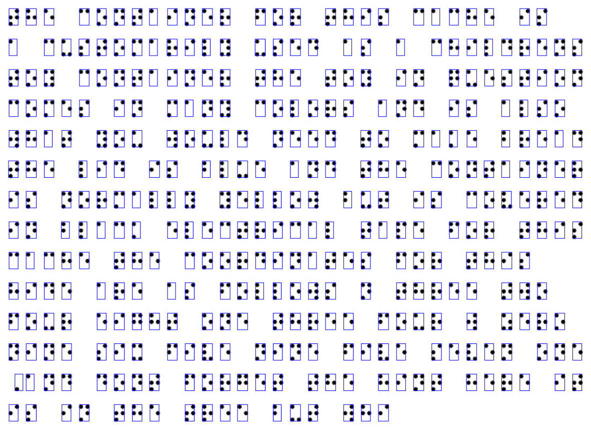
3. Find the smallest **Center Point** of all dots in each Braille Characters in both X and Y Axis.

4. With the X and Y points find all possible Braille Characters as **3x2** matrix.

5. Get each Braille Character bounding Rectangle Coordinates.
6. With the bounding Rectangle Coordinates, Each Braille Character can be Segmented.

With each Braille Character segmented as separate images we can do simple image processing to convert the image to a **3x2** matrix, this matrix can be converted to **vector with single dimension**. Each combination of such vectors will represent a character in English, Thus a sentence can be constructed.

Figure 6: Braille Characters Bounded by a Bounding Rectangle



6 Future Works

In future we propose to use Convolution Neural Network to classify each Braille Image for better accuracy and classification instead of using a Combination Hash Table as used in our Study.

7 Conclusion

Our study was able to do the segmentation and Braille Recognition in real time and at a state-of-the-art speed compared to any Artificial Neural Network based Segmentation methods. This Braille Recognition can be done in low powered devices also since we do not use complex system like Convolutional Neural Network. The accuracy is on par with any Aritficial Neural Network based Braille Recognition to the best of our knowledge with **98.457%**.

We believe that this study can be deployed in Mobile Systems with ease compared to Artificial Neural Network based Braille Recognition.

8 Miscellaneous

The source code of this entire study can be found at github which is licensed under **The BSD 3-clause "New" or "Revised" License.**

A application to demonstrate this study is hosted at heroku.