E9-241 DIGITAL IMAGE PROCESSING

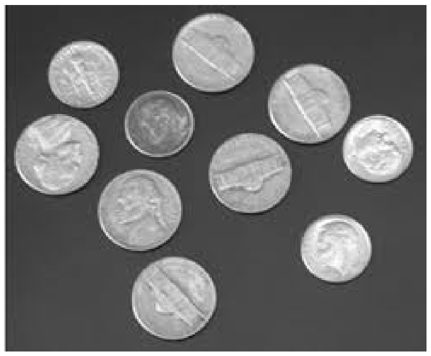
Assignment #1

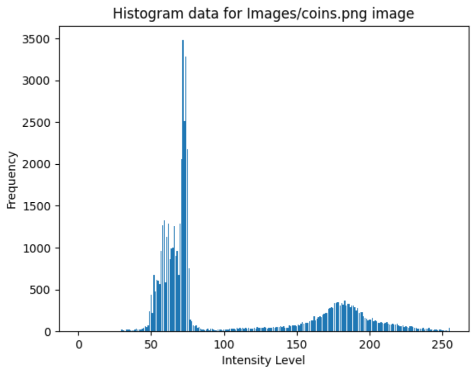
Sai Teja Kuchi (21317)
kuchisai@iisc.ac.in

*Some of the plots/intermediate outputs that are present in different sections of this report are not generated if the python file attached with the report is run. They are either commented out/removed as they are not specified to be included when the python file is run for output verification. An additional package **collections** is used for deque data structure. It's **part of the standard python package** but in-case it's throwing an error, please install the package and run the python file.

1 Histogram Computation

Computing histogram data on the image (1a), we will get the result image (1b).


(a) Image used for histogram computation



(b) Histogram Data

Figure 1: Image Data

From the image (1b), we can observe that there are 2 different distributions of intensities in the image which can be separated using thresholding.

Computing the average intensity using the two different methods we have,

$$\text{Average Intensity using histogram} = 103.30500158906722$$
$$\text{Actual Average Intensity} = 103.30500158906722$$

From the above result, we can conclude that the Average Intensity computed using the two different methods is indeed the same.

2 Otsu's Binarization

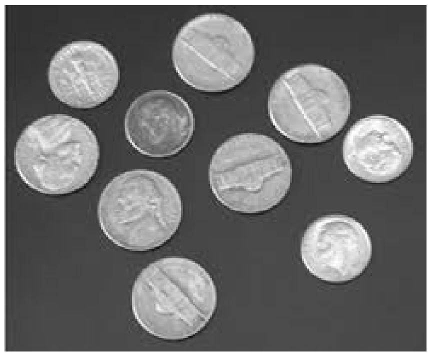
- To find the optimal threshold we sweep through values from t = 0 to 256 (both inclusive). The intensity data of the image will be split based on the threshold as described below
- At threshold t = 0,
  - Background/class1 has no intensity data.

- Foreground/class2 has the complete intensity data.

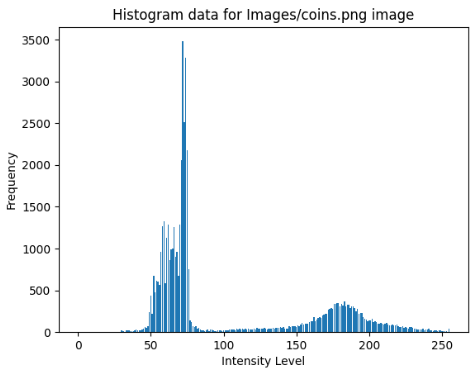
- At threshold t = 1,
  - Background/class1 has intensity data up to intensity 0.
  - Foreground/class2 has the intensity data starting from intensity 1 to intensity 255 both inclusive.
- At threshold t = 2,
  - Background/class1 has intensity data up to intensity 1.
  - Foreground/class2 has the intensity data starting from intensity 2 to intensity 255 both inclusive.

⋮

- At threshold t = 256,
  - Background/class1 has the complete intensity data.
  - Foreground/class2 has no intensity data.
- This way of thresholding might be the reason that my result varies by a value (+1) when compared with others as I am considering 2 additional edge cases at threshold values t = 0 and t = 256.



(a) Image used for histogram computation



(b) Histogram Data

Figure 2: Image Data

From the histogram data (2b) of the image (2a), we can observe that there are 2 different distributions of intensities in the image. Thus Otsu Binarization can be used in this case to find the optimal threshold for binarization of the image.

2.a Minimizing the within class variance

On applying Otsu Binarization on the image (2a) by minimizing the within-class variance, we will have

$$\text{Optimal Threshold (t)} = 126 \tag{1}$$

$$\text{Within-Class Variance } (\sigma_w^2(t = 126)) = 265.102457 \text{ (rounded to 6 decimal digits)} \tag{2}$$

$$\text{Time-taken} = 7.175207ms \tag{3}$$

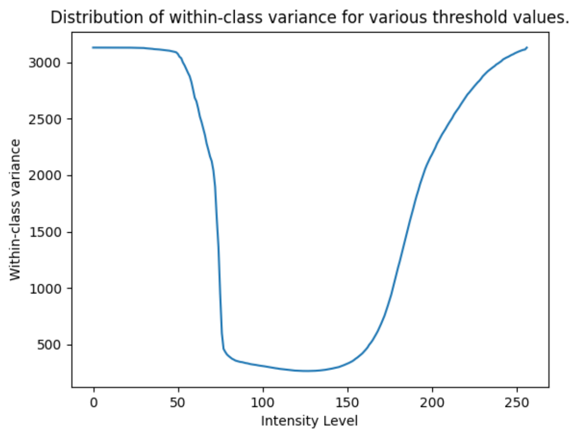


Figure 3: Within-class variance distribution across different threshold values.

2.b Maximizing the between class variance

On applying Otsu Binarization on the image (2a) by maximizing the between-class variance, we will have

$$\text{Optimal Threshold (t)} = 126 \tag{4}$$

$$\text{Between-Class Variance } (\sigma_b^2(t = 126)) = 2865.701764 \text{ (rounded to 6 decimal digits)} \tag{5}$$

$$\text{Time-taken} = 3.979444ms \tag{6}$$

2.c Inferences

From equations (1) and (4), we can see that the optimal threshold value from both methods turned out to be the same.
$\therefore$ The optimal threshold for binarization in this case is,

$$t = 126 \tag{7}$$

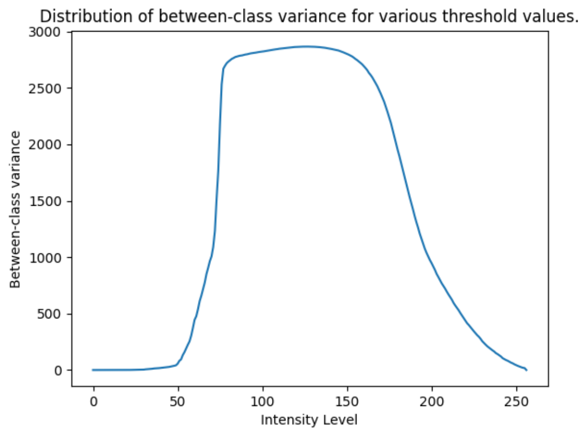


Figure 4: Between-class variance distribution across different threshold values.

On calculating the global-variance (σ_T^2), we get

$$\sigma_T^2 = 3130.804221 \text{ (rounded to 6 decimal digits)}$$

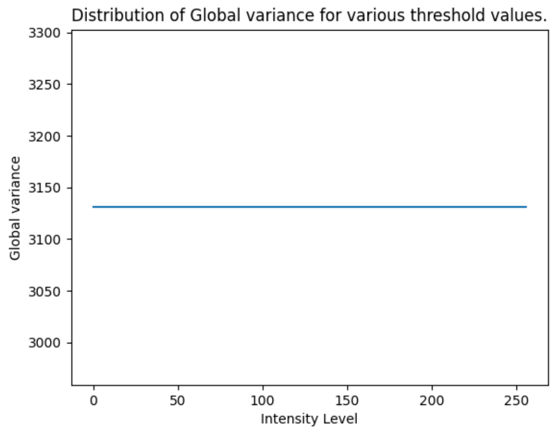


Figure 5: Global variance distribution across different threshold values.

From equations (2) and (5), we can see that the sum of the class variances at t = 126 is equal to the global variance.

$$i.e.\ \sigma_w^2(t) + \sigma_b^2(t) = 265.102457 + 2865.701764 = 3130.804221 = \sigma_T^2 \tag{8}$$

Equation (8) is indeed true for all values of threshold ranging from 0 to 256 and the same has been asserted in the python code attached along with the report and can be seen from the images (3), (4) and (5)

From equations (3) and (6), we can observe that the between-class variance (σ_b^2) takes less amount of time compared to that of the within-class variance (σ_w^2). This is because when calculating the within-class variance we also calculate the additional variance (has an power operator) term for the 2 classes, while the same is not calculated in the case of the between-class method.

A speedup of ∼1.8x (mean value of 5 different runs) can be achieved if between-class is used for finding the optimal threshold instead of within-class as both of the methods produces the same results.

Below is the binarized image obtained by thresholding the image (2a) on the optimal threshold that is obtained from Otsu Binarization (7).

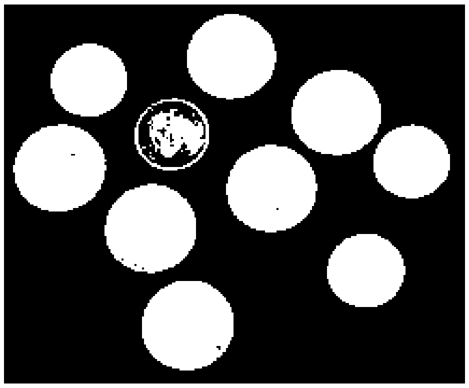
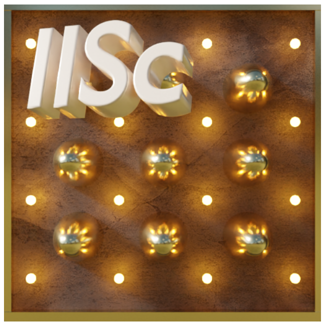


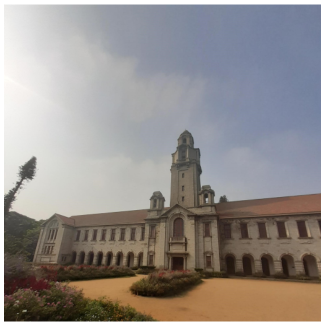
Figure 6: Binarized image using the optimal threshold

From the above image (6), we can observe that the threshold value that is obtained by using Otsu Binarization is able to differentiate the coins from the background due to the difference in the intensity levels between the 2 classes, but it's not able to pickup the design that is present on most of the coins for which techniques like edge detection can be used.

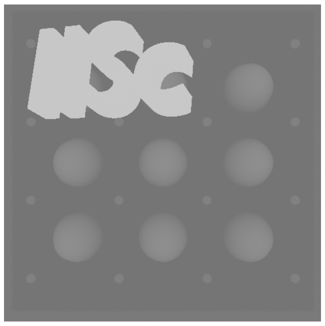
3 Depth based Extraction



(a) Text Image from which the text **IISc** needs to be extracted from.



(b) Background Image where the text **IISc** needs to be added.



(c) Depth Image which is used for extraction of the text **IISc** from Text Image.

Figure 7: Input Images

The histogram data for the depth image (7c) can be seen below,

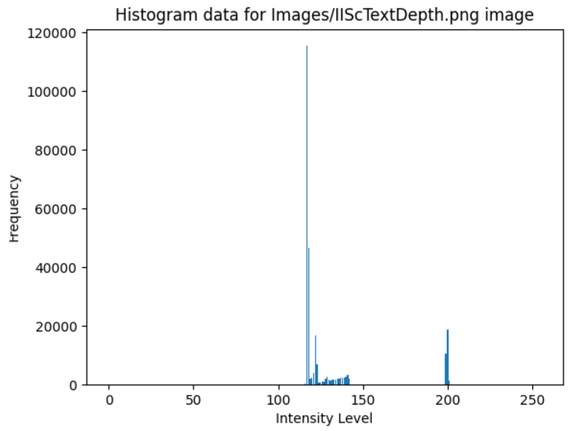


Figure 8: Histogram data

As seen from the above image (8), it contains 2 different distributions for intensities.Thus applying Otsu Binarization on the image (7c) gets us the optimal threshold value.

In this case the optimal threshold value is t = 144 and the binarized image is shown below

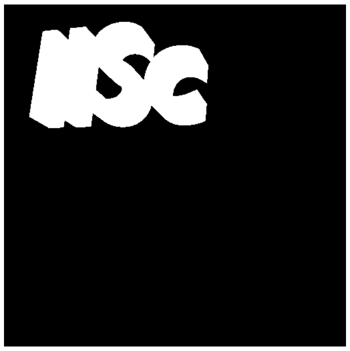


Figure 9: Binarized Image using Otsu Binarization on (7c)

Use the inverse of the above binarized image (9) (also need to stack the image because we are applying it on an color image) on the background image (7b) to create empty space for the text. The result obtained is shown below

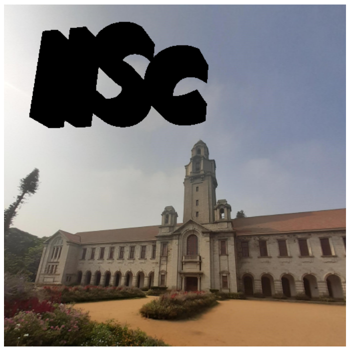


Figure 10: Background image with text space.

Use the binarized image (9), to extract the text from the Text Image (7a). The result obtained is shown below

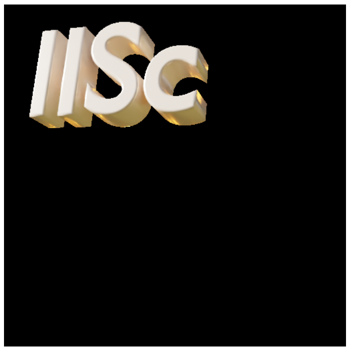


Figure 11: Text extracted from Text Image (7a) using the binarized image (9).

Combining the images from (10) and (11) provides us with the final result image shown as below.

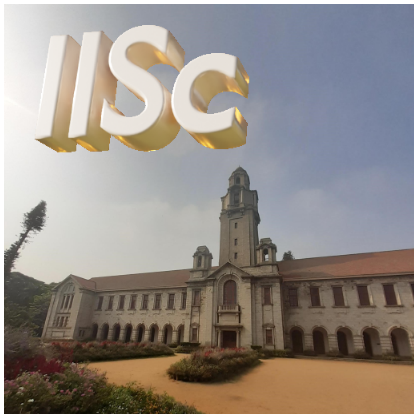
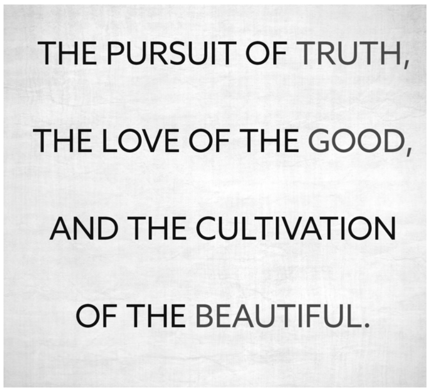


Figure 12: Final Result Image

4 Connected Components

Computing histogram data on the image (13a), we will get the result image (13b).



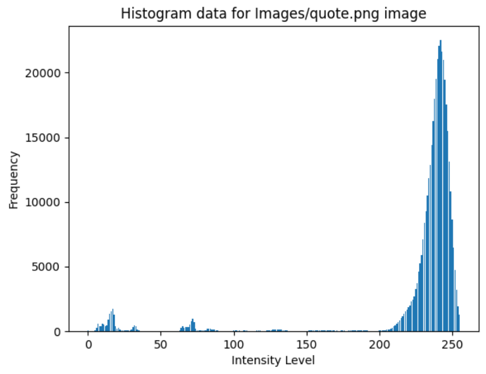
THE PURSUIT OF TRUTH,

THE LOVE OF THE GOOD,

AND THE CULTIVATION

OF THE BEAUTIFUL.

(a) Image with Quote.



(b) Histogram Data

Figure 13: Image Data

From the image (13b), we can observe that most of the intensity data is towards the right end of the distribution which is background data in this case. We can apply Otsu Binarization on the image to get the optimal threshold and separate the characters from the background data.

In this case, on applying Otsu binarization we get the optimal threshold value as t = 143 and the inverse binarized image (14) is shown below

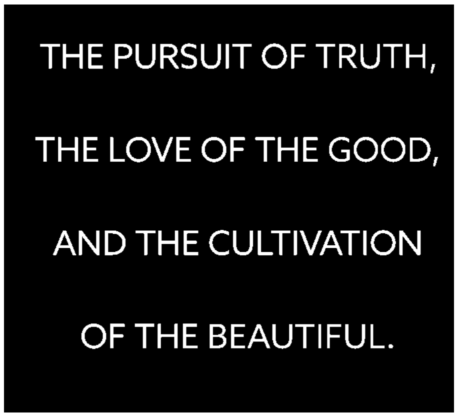


Figure 14: Inverse Binarized image of (13a).

As seen from the histogram data of the image (13b), most of the frequency of intensity levels is background noise. So when finding the connected components we can set a maximum-threshold such that if the region count for that connected component is greater than 1% of the number of pixels, we can reject such components as they are not related to the characters that we are looking for in the image.

On performing the connected-component analysis and plotting the distribution of the sizes of the characters (number of pixels in the image) found, we will be left with the below image,

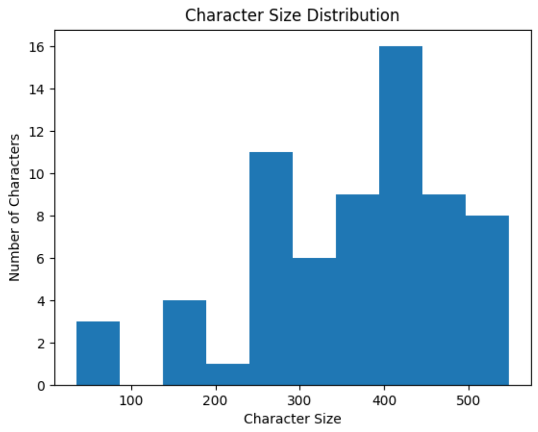


Figure 15: Character Size Distribution on using connected component analysis

From the above image (15), we can observe that there is an gap in the distribution and this is what is separating the actual characters from the punctuations. Further on performing empirical analysis on the distribution, using 20% of maximum character size from the distribution as the threshold helps in separating the characters from the punctuations.

Thus, we are using a total of 3 threshold values.

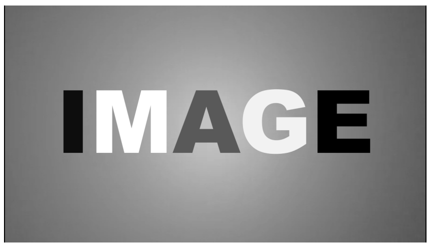
- optimal_threshold that is found using Otsu Binarization for getting binarized image (14) to perform connected-component analysis on it.
- maximum_threshold = 1% of total pixel count of the image. This is used for removing components which are not part of the characters in the image and is mostly background data.
- minimum_threshold = 20% of character size after applying the above 2 thresholds. This helps in excluding the punctuations from the actual characters.

On performing the above analysis, we find that the total number of characters excluding punctuations for the image (13a) is **64**.

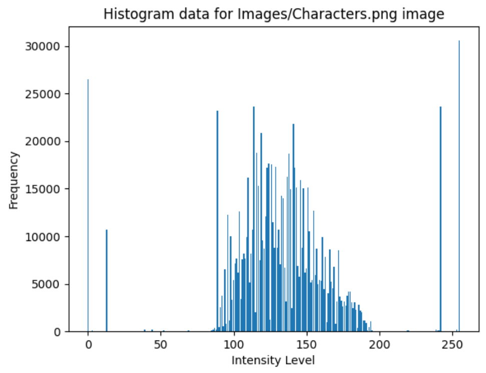
Same analysis with the above threshold conditions is performed on 2 more different images (attached with this report, present in Images folder named test_quote.png and test_quote1.png) which are similar to the input image (13a) provided. They also resulted in correct output for the number of characters excluding punctuations. So unless the characters sizes are too close to that of the punctuations this method works fine.

5 Maximally Stable Extremal Regions (MSER)

A total of $\sim$**24 mins** is taken for running this problem as it involves coming the connected-components for every threshold value followed by some post-processing.
Computing histogram data on the image (16a), we will get the result image (16b).



(a) Image used for histogram computation



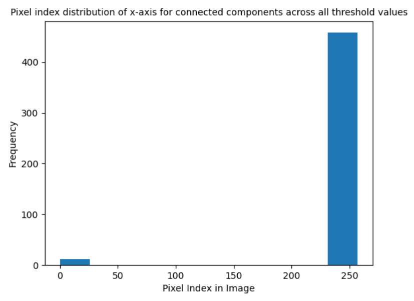
(b) Histogram Data

Figure 16: Image Data

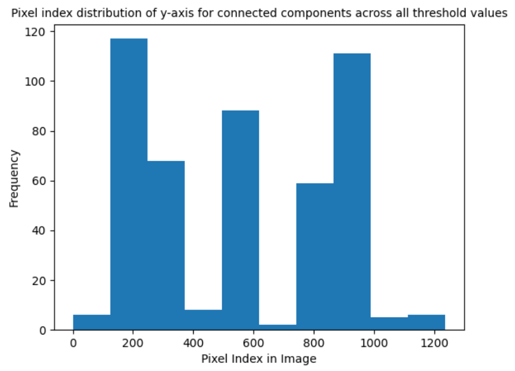
Applying Otsu binarization and finding connected-components on the image (16a) will not work because, when Otsu binarization is applied we will be having a single threshold which will divide the entire image into 2 different intensity levels, but as seen from the image (16b) we have multiple peaks with different intensity levels. Hence Otsu binarization will fail to pick up few characters which are not within the threshold that obtained through that method. As MSER sweeps through all possible threshold values, it will be able to identify the number of characters in the image.

Borrowing the ideas from the previous question. We are using 2 threshold values initially to filter noise from the connected-components.

- maximum_threshold = 4% of total pixel count of the image. This is used for removing components which are not part of the characters in the image and is mostly background data.
- minimum_threshold = 30% of character size after applying the above threshold. This helps in further reducing noise from the connected-components.



(a) Image used for histogram computation



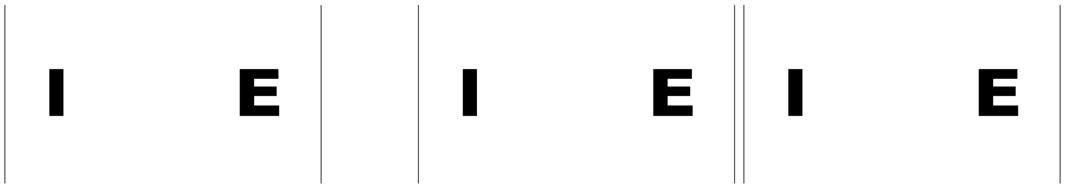
(b) Histogram Data

Figure 17: Image Data

From the image (17a) which shows the distribution of x-axis index for connected-components we can clearly see that most of the indexes are above 250 and some are below zero. The zero indexes are the one which are noise-data so we will be filtering them before proceeding with further analysis. Similarly, image (17b) shows the distribution of y-axis index for connected-components of the image.
We set the values $\epsilon = 4$ and $\delta = 20$ for finding the number of connected-components in the image and we found out that there are **5 characters** in the image.
Below are some the the stable thresholds found for the connected-components.

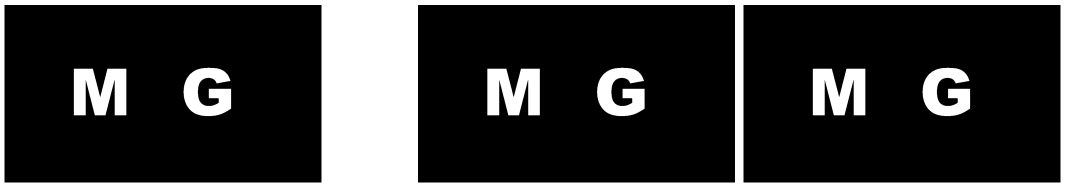
Connected-component number	Corresponding Character in Image	Stable Threshold value
1	A	125, 127, 140, etc
2	G	205, 206, 207, etc
3	I	15, 16, 17, etc
4	E	7, 8, 10, etc
5	M	230, 231, 232, etc

Below are the attached binarized images for each connected-component/characters which shows that the above results are valid.



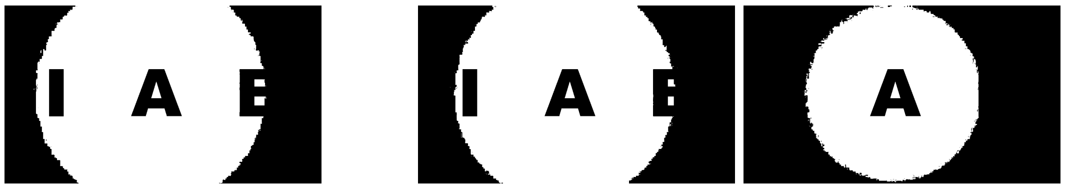
(a) Thresholded at 15 (b) Thresholded at 16 (c) Thresholded at 17

Figure 18: Binarized images for character I



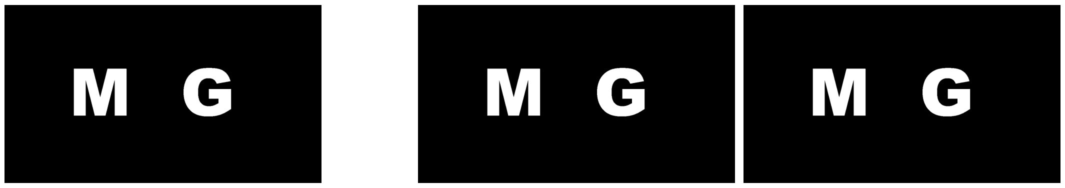
(a) Thresholded at 230 (b) Thresholded at 231 (c) Thresholded at 232

Figure 19: Binarized images for character M



(a) Thresholded at 125 (b) Thresholded at 127 (c) Thresholded at 140

Figure 20: Binarized images for character A



(a) Thresholded at 205 (b) Thresholded at 206 (c) Thresholded at 207

Figure 21: Binarized images for character G

(a) Thresholded at 7 (b) Thresholded at 8 (c) Thresholded at 10

Figure 22: Binarized images for character E