1 Introduction

In this homework we perform image segmentation using the Otsu's Method. Otsu's method identifies the optimum grayscale level that separates the foreground from the background. To carry out image segmentation using Otsu's method we used two different approaches

1. In the first approach, **RGB based Image Segmentation**, we apply the Otsu's method separately to the three channels, RGB, of the image and then combine the three segmented images to get the final result.

2. In the second approach, **Texture based Segmentation**, we first use texture-based characterization of the pixels and then apply the Otsu's method to the different characterizations treating them as color channels.

In the following sections we first describe the Otsu's method and then discuss the two approaches used for carrying out image segmentation in this study.

2 Otsu's Method

The Otsu's method an be described by the following steps:

1. In the first step we calculate the pixel grayscale level histogram. For the 256 gray levels in range [0, 255], we construct the probability distribution function,

$$p_i = \frac{n_i}{N}, \tag{1}$$

where n_i denotes the number of pixels at i^{th} grayscale level and $N = n_1 + n_2 + n_3 + ...n_{256}$ is the total number of pixels in the image.

2. If k denotes the grayscale level that separates the foreground from the background then we denote the pixels with grayscale level less than k by class C_0 and for pixels with grayscale level greater than k we use class C_1. Let ω_0 and ω_1 denote the probability of class C_0 and C_1 respectively.

$$\omega_0 = \sum_{1}^{k} p_i, \tag{2}$$

$$\omega_1 = 1 - \omega_0. \tag{3}$$

3. Next, we calculate the mean for each class μ_0, μ_1 and the between-class variance σ_b^2,

$$\mu_0 = \frac{1}{\omega_0} \sum_{i=1}^{k} i p_i, \tag{4}$$

$$\mu_1 = \frac{1}{\omega_1} \sum_{i=k+1}^{255} i p_i, \tag{5}$$

$$\sigma_b^2 = \omega_0 \omega_1 (\mu_0 - \mu_1)^2. \tag{6}$$

4. We select the optimum threshold k^* as the one which which maximizes σ_b^2.

5. In the next step we carry out masking of the image in such a way that the pixel value of the pixels with grayscale level lower than k^* are changed to 0 to mark the background and pixels with grayscale level greater than k^* are changed to 1 to mark the foreground.
6. To improve the segmentation results, we iterate from Step 1 to Step 5. The number of iterations is a hyperparameter.

3 RGB based Segementation

The RGB based segementation using the Otsu's method is carried out in the following manner:

1. In the first step, we separate the colored image into the three RGB channels.
2. Next, we obtain the foreground mask for each channel using the Otsu's method described in Section 2. Let these masks be mask_r, mask_g, mask_b for the RGB channels respectively.
3. We now have to combine the masks for all the three channels to obtain the final mask which separates the foreground from the background. The combination logic depends on the characteristics of the image.
   (a) For the red lighthouse image we need the red lighthouse as the foreground and the blue and green channels as the background. So we find the combined mask using, **mask = mask_r AND (NOT mask_b) AND (NOT mask_g)**
   (b) For the cute baby image, there is not particular dominant color channel. So we calculate the combined mask as **mask = mask_r AND mask_b AND mask_g**.
   (c) For the jumping man image also there is not dominant color so the combined mask is, **mask = mask_r AND mask_b AND mask_g**.

4 Texture based Segementation

The texture based segmentation consists of the following steps:

1. In the first step we convert the colored image into its grayscale form.
2. In the second step we create a grayscale image such that each pixel represents the variance of the grayscale images of the $N \times N$ window around the corresponding pixels of the original grayscale image. This is done for three different values of $N = 3, 5, 7$ so that we get three different textures of the original image.
3. Treating the three different textures of the images as three different channels, we apply the Otsu's method to each channel to get three different foreground masks mask_3, mask_5 and mask_7 for values of $N = 3$, $N = 5$ and $N = 7$ respectively.
4. In the next step we merge the foreground and the background mask using the variance properties in each of the images.
   (a) The red lighthouse image does not have significant variance in pixels of the foreground. Hence we should unmask all the the three foreground masks mask_3, mask_5 and mask_7 to get the final foreground mask, **mask = NOT mask_3 AND (NOT mask_5) AND (NOT mask_7)**.

(b) For the cute baby image, there significant variance on the pixels for the foreground so all the channels are treated equally to give the final mask as, **mask = mask_3 AND mask_5 AND mask_7**.

(c) For the jumping man image also there is variance in the pixel values for the foreground so the final mask is, **mask = mask_3 AND mask_5 AND mask_7**.

5 Contour Extraction

After using either of the methods described in Section 3 and Section 4 to segment the image, it is necessary to extract the contour of the foreground mask from the background mask by tracing a border around the foreground mask. The algorithm used in this study is based on using the 8-neighbors of each pixel. The steps involved in this algorithm are as follows:

1. If the pixel value is 0, it is treated as background and not selected for contour extraction.
2. If the pixel value is 1 and all its 8-neighbors are 1, then it indicates that the point is inside foreground mask and does not represent the contour border.
3. If the pixel value is 1 and at least one of its 8-neighbors is 0, then the point is selected as a contour border point.

Having discussed the methodology for carrying out the image segmentation we now discuss the implementation in the code in the following section.

6 Implementation in Code

All the methods discussed in the previous sections has been implemented in the Python script **hw6_TejasPant.py**. The structure of the code is as follows:

1. In the first step we read the image using **cv2.imread**.
2. Next we carry out the RGB based Segmentation in the method **RGBSegment** which has the input arguments number of iterations per image channel **niterChannels** and the flag to indicate if the foreground mask of each channel needs to be inverted or not **channelInvert**. The Otsu's algorithm is called in the method **RGBSegment**.
3. The Otsu's algorithm has been implemented in the method **OtsuMethod**. This image returns the foreground mask with pixel values 1 and background mask with pixel values 0.
4. For the Texture based Segmentation method, we first extract the texture features using the method **getImageTexture**. Three window sizes 3 x 3, 5 x 5 and 7 x 7 are used to extract the features by calculating the variance of the pixels in the window. The three variances obtained by using the three window sizes at each pixel location now represents the texture measure at that pixel location.
5. In the next step we again use the method **RGBSegment** to segment the image with the input image being the 3D characterization of the image being the variance values instead of the RGB channels.

6. Once the segmentation is carried out using either of the two segmentation methods, the segmented image will have some noise. We remove the noise using dilation and erosion. This is done using the methods **cv2.dilate** and **cv2.erode**
7. In the last step, we extract the contour of the foreground mask. This is implemented in the method **getForegroundContour**. The algorithm for this method is discussed in Section 5

7 Parameters Used

The parameters used in the code for the different images and different methods is described below.

1. **Image 1: Lighthouse Image**
   (a) RGB based Segmentation
      i. Number of iterations for Otsu's method for blue channel = 1
      ii. Number of iterations for Otsu's method for green channel = 1
      iii. Number of iterations for Otsu's method for red channel = 1
      iv. Kernel size for erosion and dilation of foreground = 2 x 2
      v. Kernel size for erosion and dilation of background = 4 x 4
   (b) Texture based Segmentation
      i. Window sizes for texture operators = 3 x 3, 5 x 5 and 7 x 7
      ii. Number of iterations for Otsu's method for blue channel = 1
      iii. Number of iterations for Otsu's method for green channel = 1
      iv. Number of iterations for Otsu's method for red channel = 1
      v. Kernel size for erosion and dilation of foreground = 2 x 2
      vi. Kernel size for erosion and dilation of background = 4 x 4
2. **Image 2: Cute Baby Image**
   (a) RGB based Segmentation
      i. Number of iterations for Otsu's method for blue channel = 2
      ii. Number of iterations for Otsu's method for green channel = 2
      iii. Number of iterations for Otsu's method for red channel = 2
      iv. Kernel size for erosion and dilation of foreground = 2 x 2
      v. Kernel size for erosion and dilation of background = 4 x 4
   (b) Texture based Segmentation
      i. Window sizes for texture operators = 3 x 3, 5 x 5 and 7 x 7
      ii. Number of iterations for Otsu's method for blue channel = 1
      iii. Number of iterations for Otsu's method for green channel = 1
      iv. Number of iterations for Otsu's method for red channel = 1
      v. Kernel size for erosion and dilation of foreground = 2 x 2
      vi. Kernel size for erosion and dilation of background = 4 x 4
3. **Image 3: Jumping Man Image**
   (a) RGB based Segmentation

   i. Number of iterations for Otsu's method for blue channel = 1
   ii. Number of iterations for Otsu's method for green channel = 2
   iii. Number of iterations for Otsu's method for red channel = 1
   iv. Kernel size for erosion and dilation of foreground = 2 x 2
   v. Kernel size for erosion and dilation of background = 4 x 4

(b) Texture based Segmentation

   i. Window sizes for texture operators = 3 x 3, 5 x 5 and 7 x 7
   ii. Number of iterations for Otsu's method for blue channel = 1
   iii. Number of iterations for Otsu's method for green channel = 1
   iv. Number of iterations for Otsu's method for red channel = 1
   v. Kernel size for erosion and dilation of foreground = 2 x 2
   vi. Kernel size for erosion and dilation of background = 4 x 4

8 Observations

Following are the key observations made in this study:

1. It can be observed that the performance of the two different methods for image segmentation significantly depends on the characteristics of the image and it is not possible to conclude if one method is better than the other.

2. For the lighthouse image, the RGB based method outperforms the Texture based method. This is because in this image there is strong gradient of the grayscale level near the borders of the lighthouse making it easier to use the RGB color scheme to separate the foreground of the lighthouse from the background.

3. For the other two images namely, cute baby and jumping man image, the Texture based method is able to segment the image relatively better than the RGB based method. This can be attributed to the fact that in these images the foreground has more texture in comparison to the background.

4. Since many of the real-life images might not have strong gradients of the grayscale levels at the border of the foreground, the Texture based segmentation method can be suggested to be a more general method.

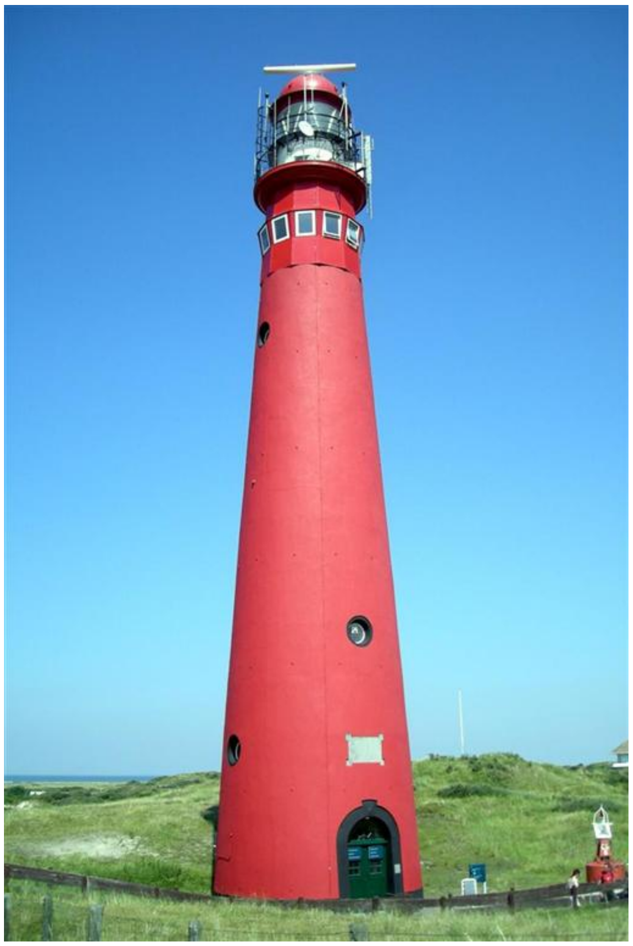

Figure 1: Original red lighthouse image.

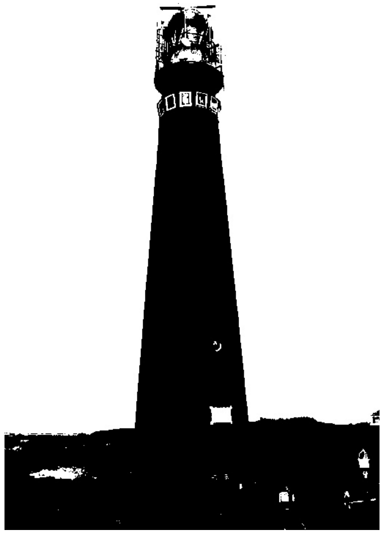
Figure 2: Foreground mask using **blue channel** of the image for 1 iteration Ostu's Method using **RGB based Segmentation**.

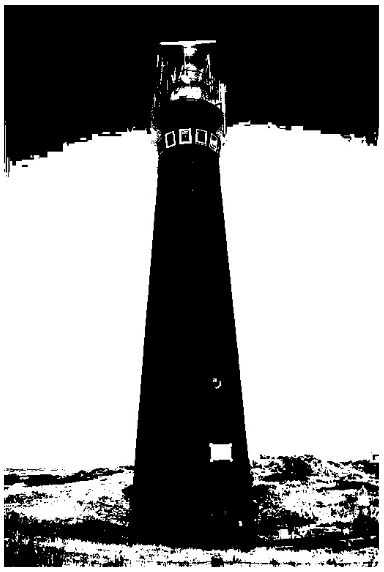
Figure 3: Foreground mask using **green channel** of the image for 1 iteration Ostu's Method using **RGB based Segmentation**.

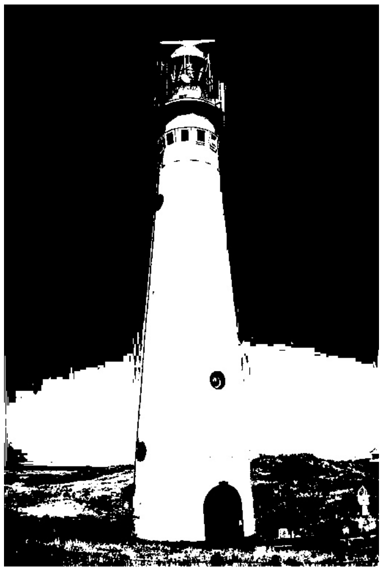

Figure 4: Foreground mask using **red channel** of the image for 1 iteration Ostu's Method using **RGB based Segmentation**.

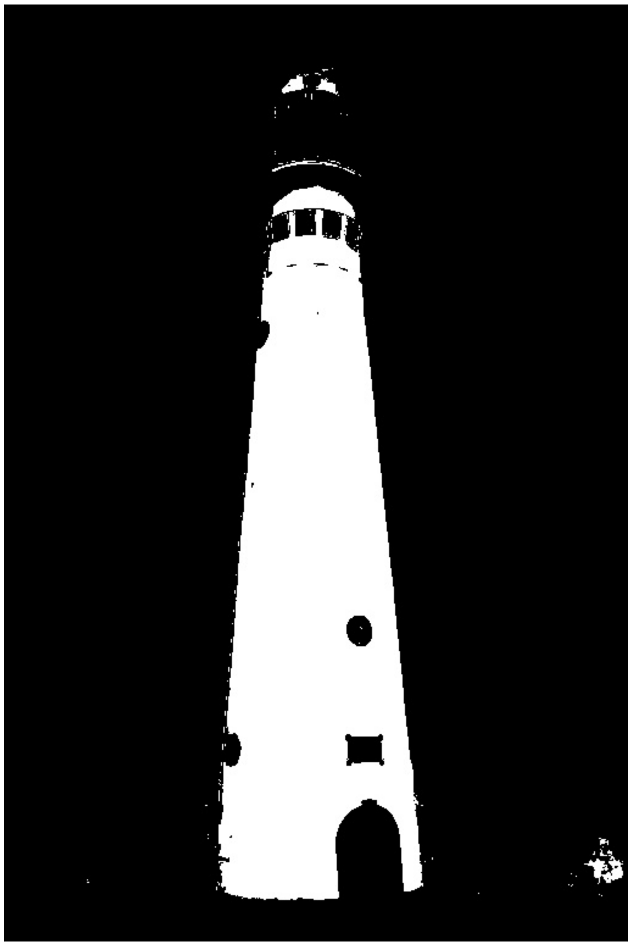

Figure 5: The overall foreground mask **with noise** using the masks of the **three channels** of the image using **RGB based Segmentation**.

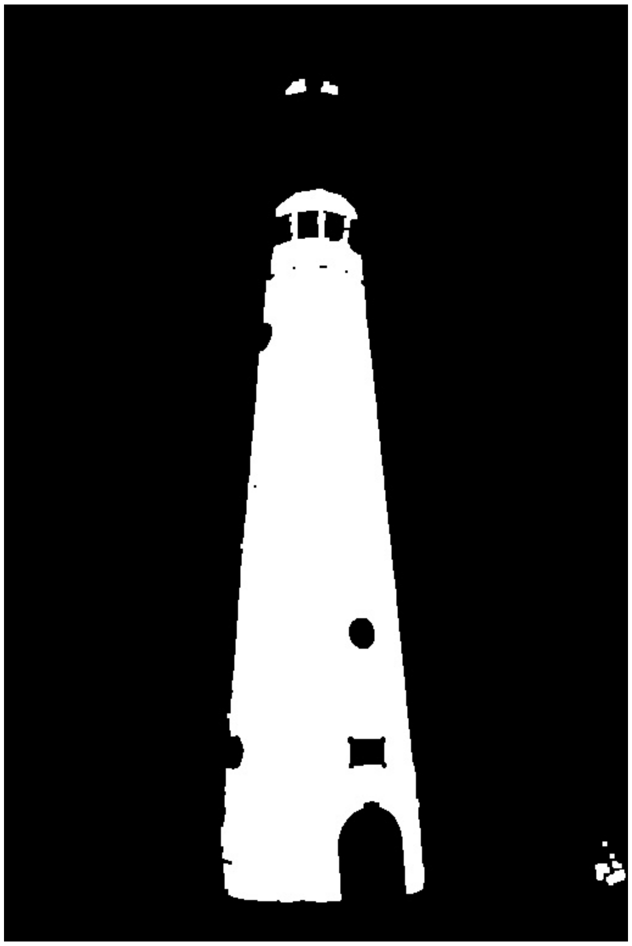

Figure 6: The overall foreground mask after **removing noise** using **erosion** and **dilation** for the **RGB based Segmentation**.

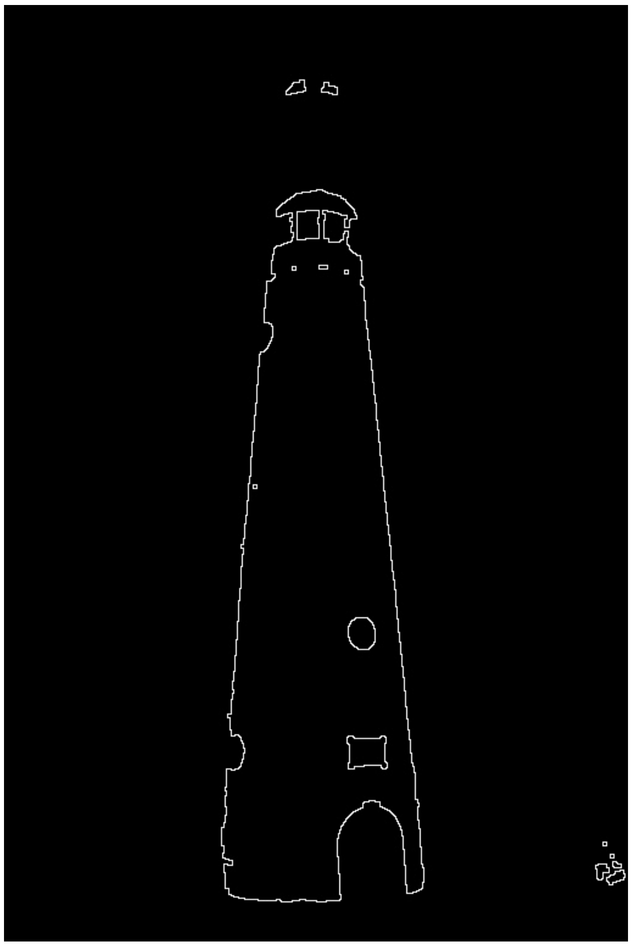

Figure 7: The final contour of the image for the **RGB based Segmentation**.

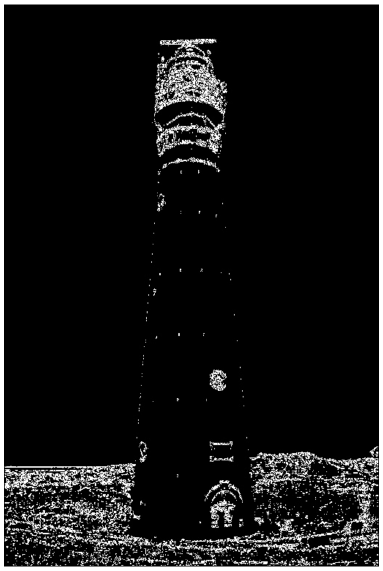

Figure 8: Foreground mask using window size **N = 3 X 3** for 1 iteration Ostu's Method using **Texture based Segmentation**.

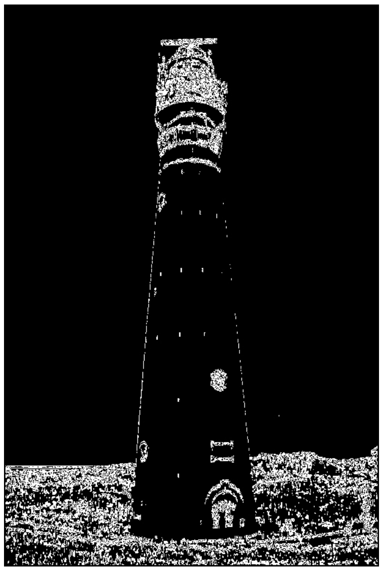

Figure 9: Foreground mask using window size **N = 5 X 5** for 1 iteration Ostu's Method using **Texture based Segmentation**.

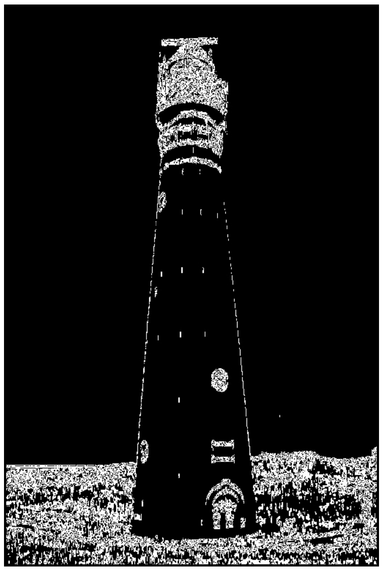

Figure 10: Foreground mask using window size **N = 7 X 7** for 1 iteration Ostu's Method using **Texture based Segmentation**.

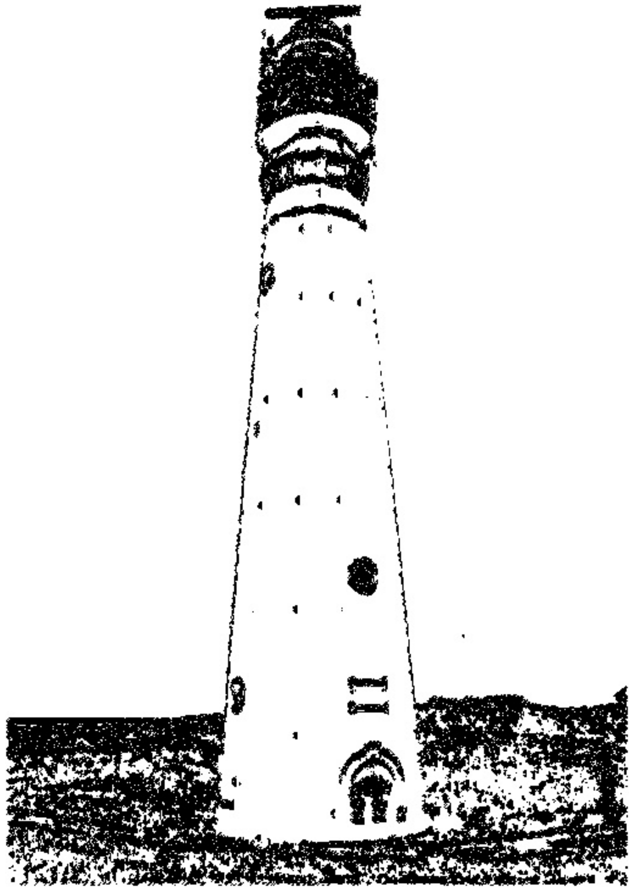

Figure 11: The overall foreground mask **with noise** using the masks of the **three window sizes** using **Texture based Segmentation**.

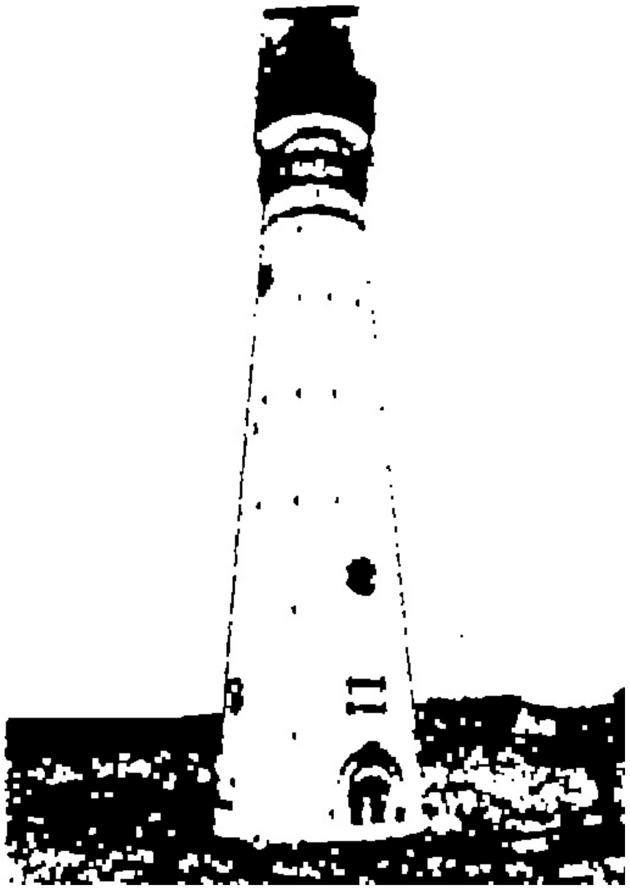

Figure 12: The overall foreground mask after **removing noise** using **erosion** and **dilation** for the **Texture based Segmentation**.

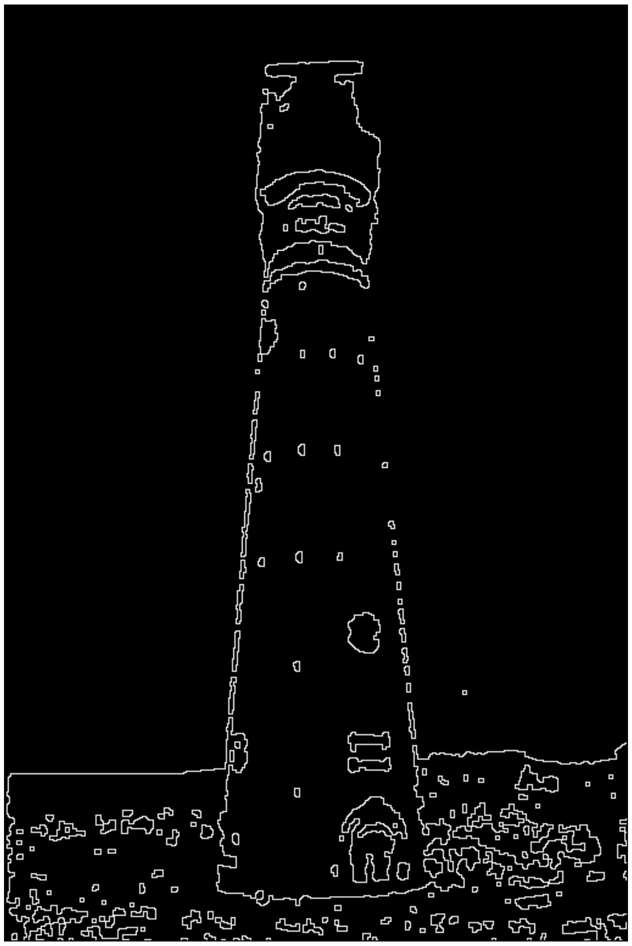

Figure 13: The final contour of the image for the **Texture based Segmentation**.

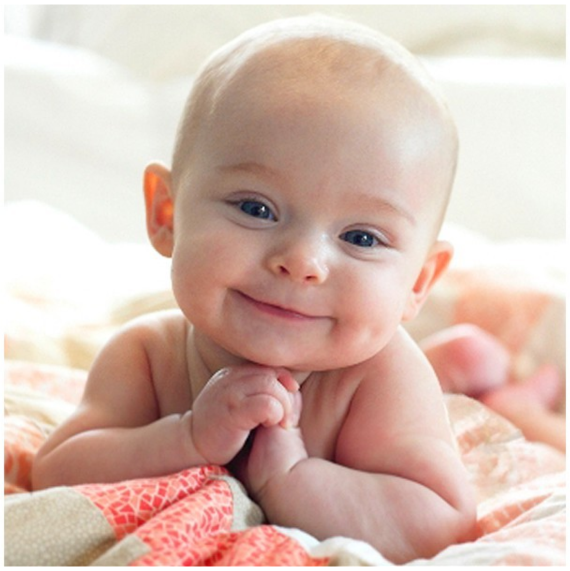

Figure 14: Original cute baby image.

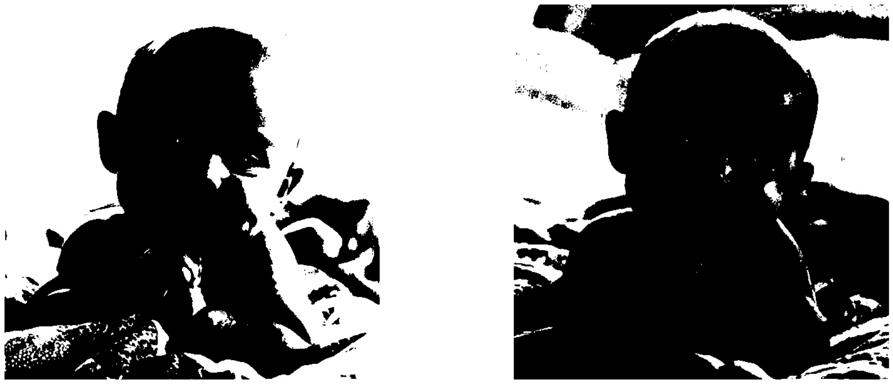

Figure 15: Foreground mask using **blue channel** of the image for 2 iterations Ostu's Method using **RGB based Segmentation**.

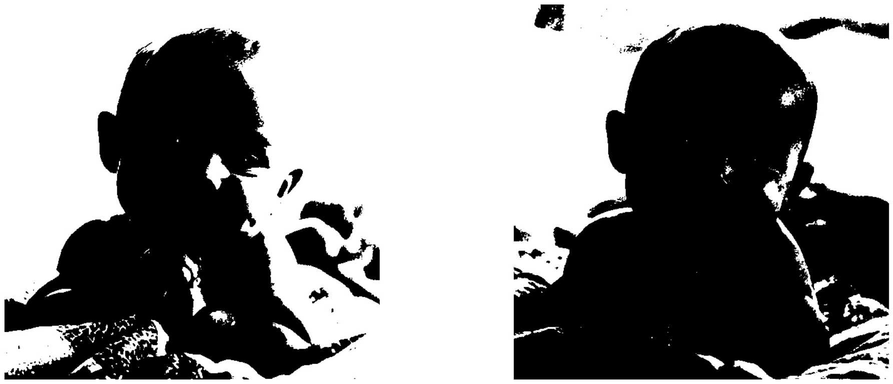

Figure 16: Foreground mask using **green channel** of the image for 2 iterations Ostu's Method using **RGB based Segmentation**.

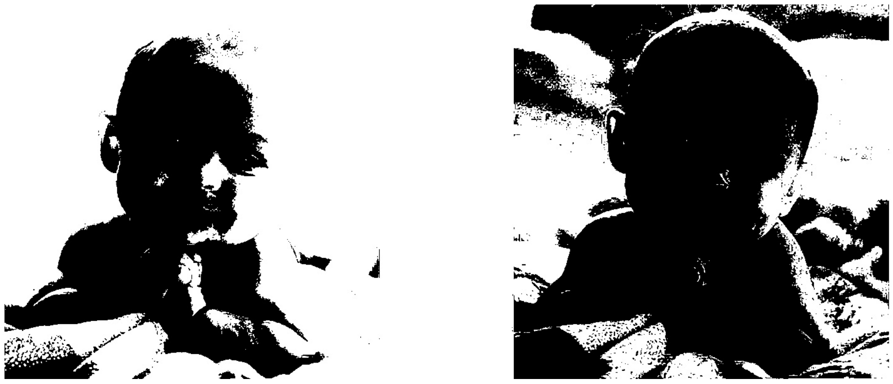

Figure 17: Foreground mask using **red channel** of the image for 2 iteration Ostu's Method using **RGB based Segmentation**.

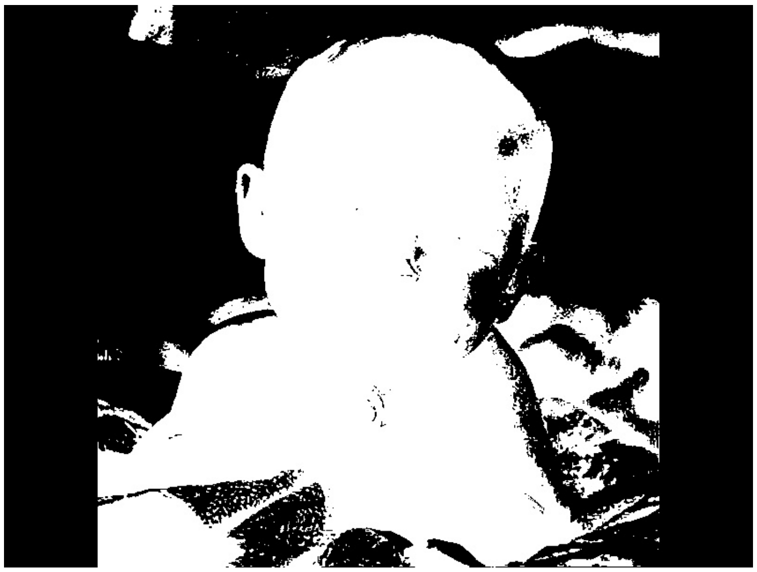

Figure 18: The overall foreground mask **with noise** using the masks of the **three channels** of the image using **RGB based Segmentation**.

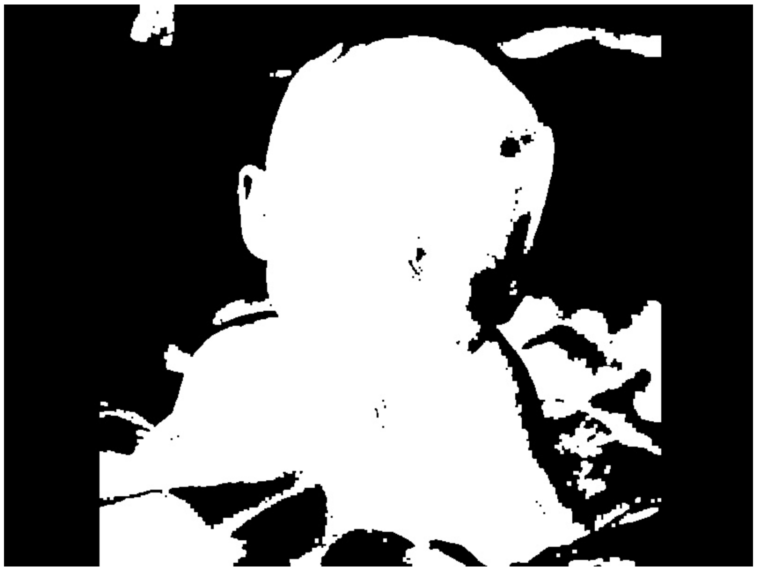

Figure 19: The overall foreground mask after **removing noise** using **erosion** and **dilation** for the **RGB based Segmentation**.

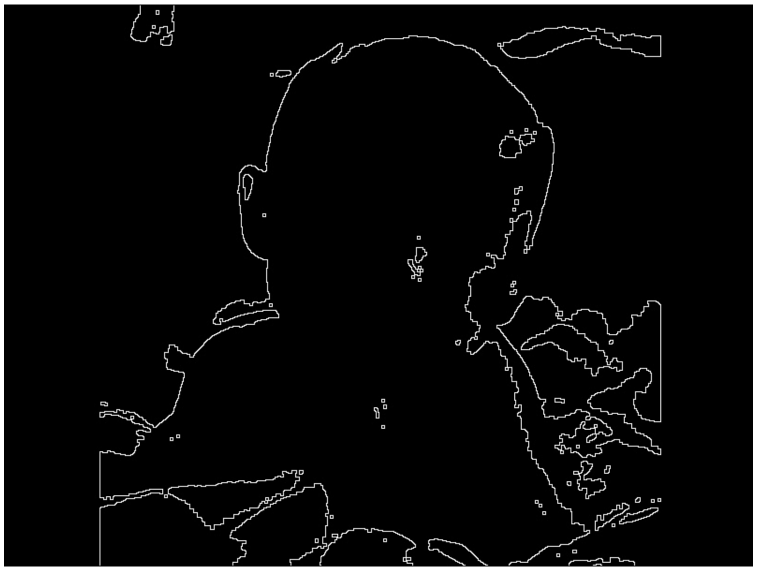
Figure 20: The final contour of the image for the **RGB based Segmentation**.

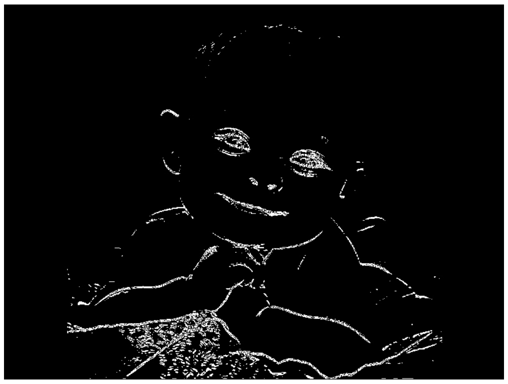
Figure 21: Foreground mask using window size **N = 3 X 3** for 1 iteration of Ostu's Method using **Texture based Segmentation**.

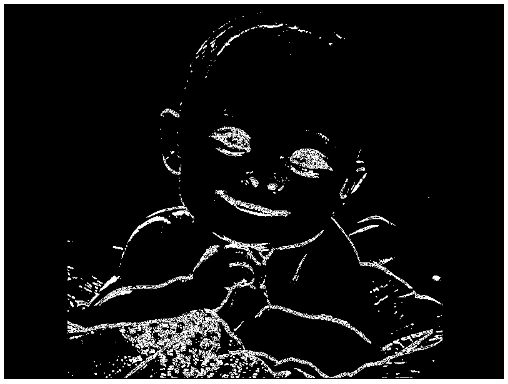
Figure 22: Foreground mask using window size **N = 5 X 5** for 1 iteration of Ostu's Method using **Texture based Segmentation**.

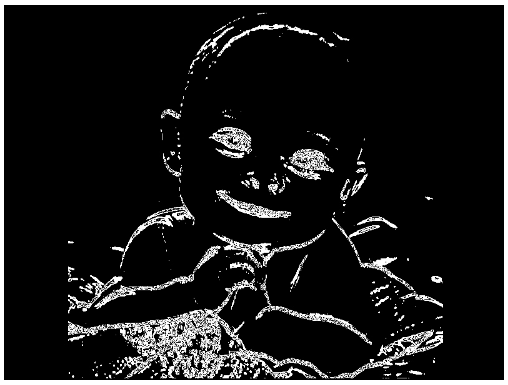

Figure 23: Foreground mask using window size **N = 7 X 7** for 1 iteration of Ostu's Method using **Texture based Segmentation**.

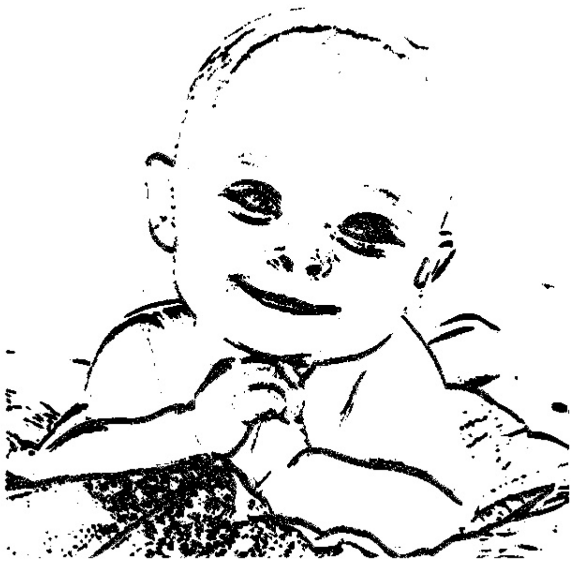

Figure 24: The overall foreground mask **with noise** using the masks of the **three window sizes** using **Texture based Segmentation**.

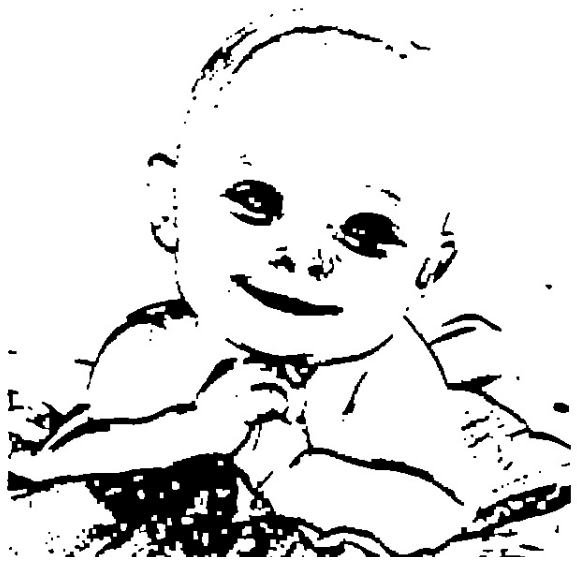
Figure 25: The overall foreground mask after **removing noise** using **erosion** and **dilation** for the **Texture based Segmentation**.

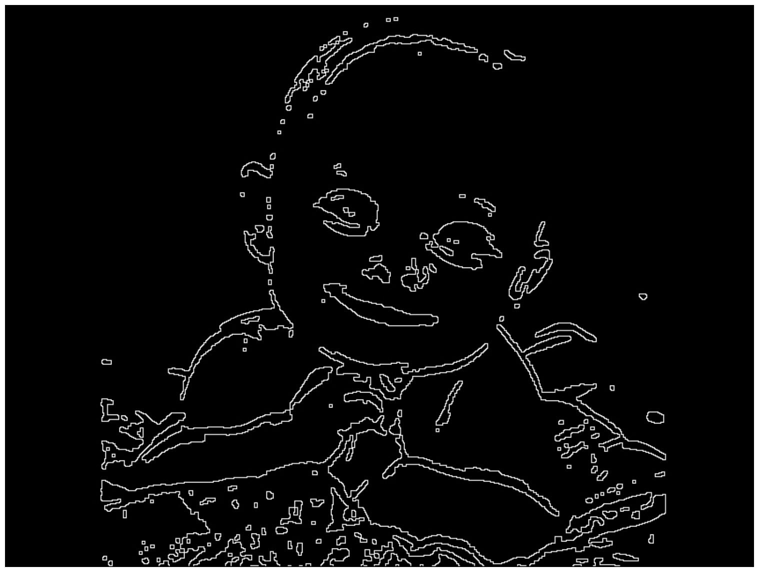
Figure 26: The final contour of the image for the **Texture based Segmentation**.

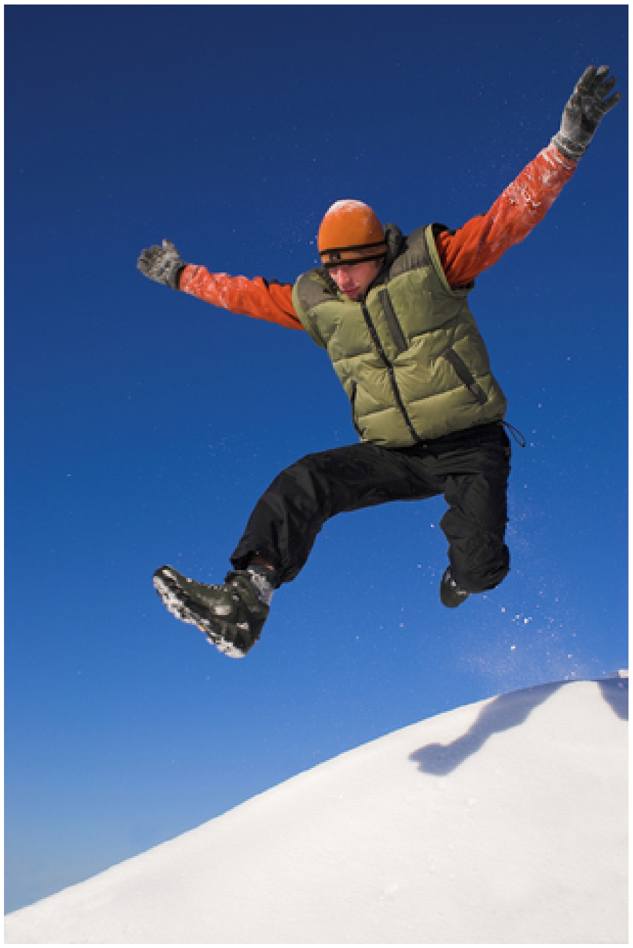

Figure 27: Original jumping man image.

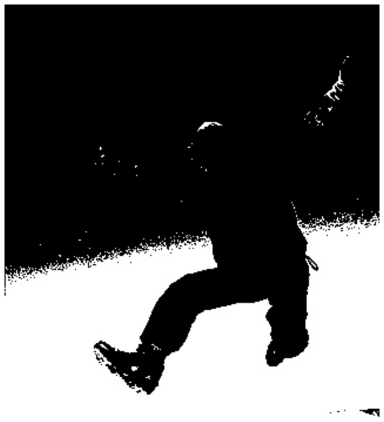

Figure 28: Foreground mask using **blue channel** of the image for 1 iteration Ostu's Method using **RGB based Segmentation**.

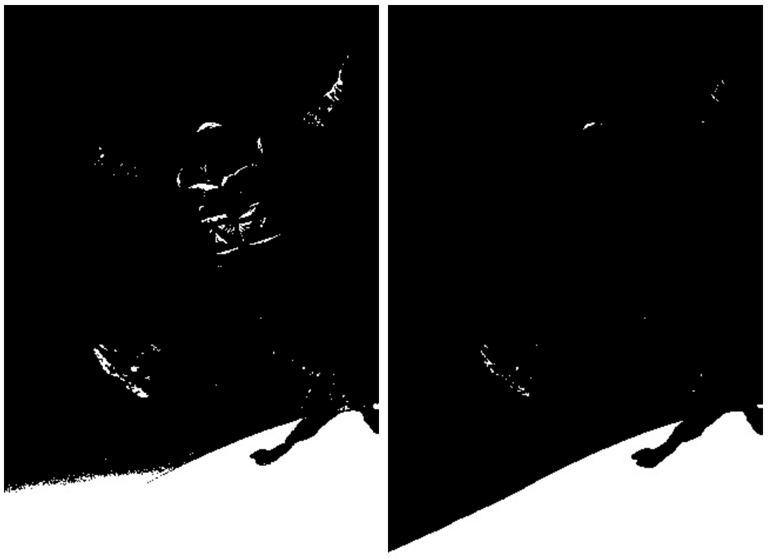

Figure 29: Foreground mask using **green channel** of the image for 1 iteration Ostu's Method using **RGB based Segmentation**.

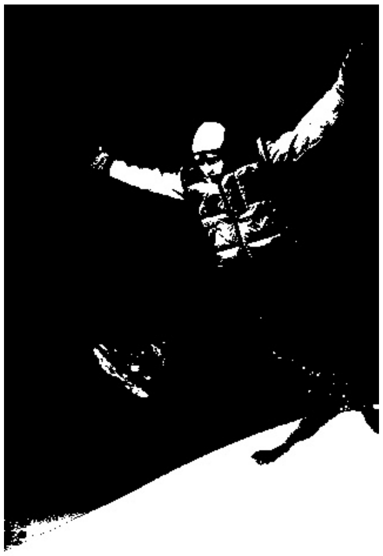

Figure 30: Foreground mask using **red channel** of the image for 1 iteration Ostu's Method using **RGB based Segmentation**.

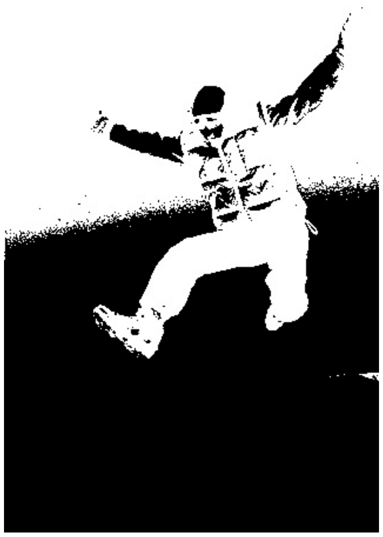

Figure 31: The overall foreground mask **with noise** using the masks of the **three channels** of the image using **RGB based Segmentation**.

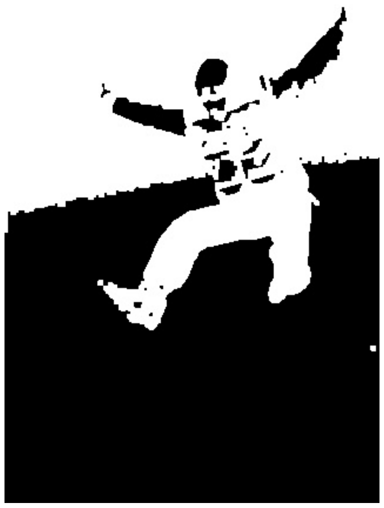

Figure 32: The overall foreground mask after **removing noise** using **erosion** and **dilation** for the **RGB based Segmentation**.

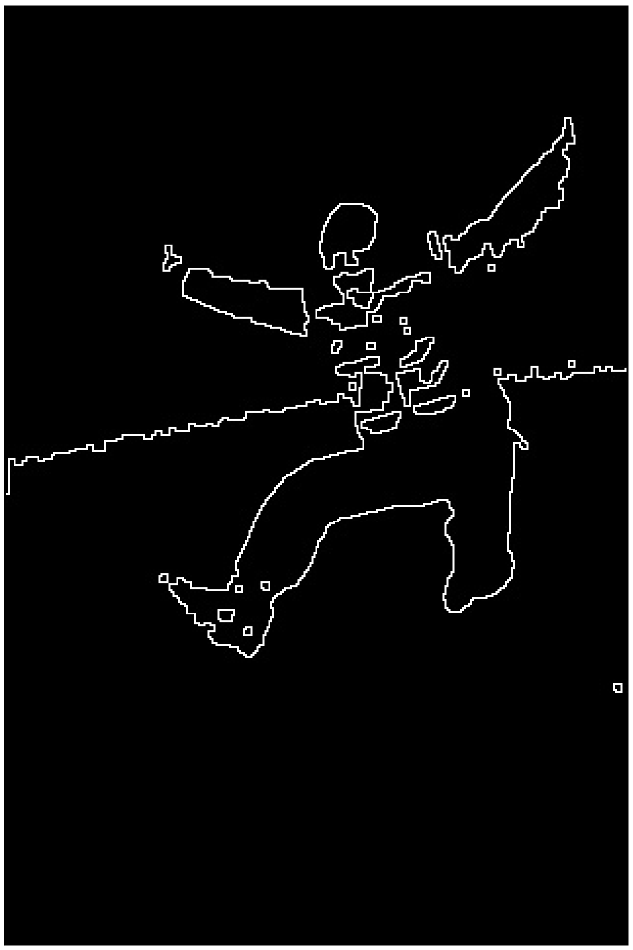

Figure 33: The final contour of the image for the **RGB based Segmentation**.

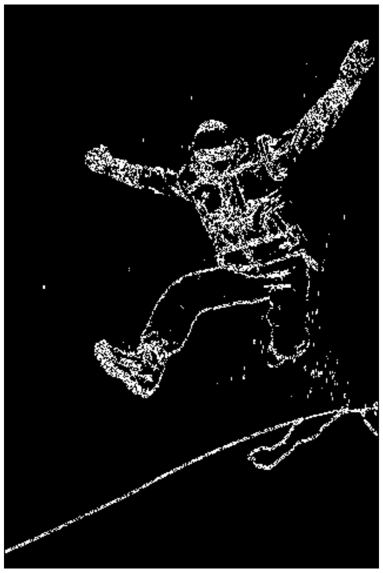

Figure 34: Foreground mask using window size **N = 3 x 3** with 1 iteration of Ostu's Method using **Texture based Segmentation**.

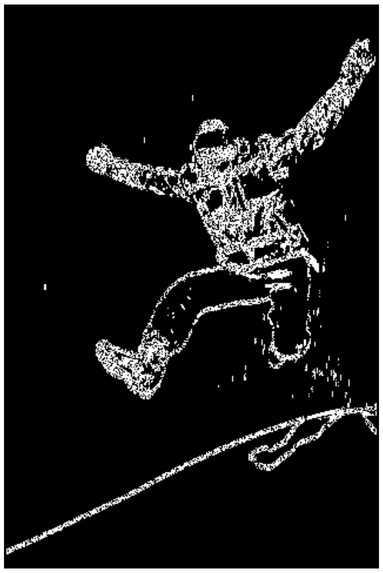

Figure 35: Foreground mask using window size **N = 5 x 5** with 1 iteration of Ostu's Method using **Texture based Segmentation**.

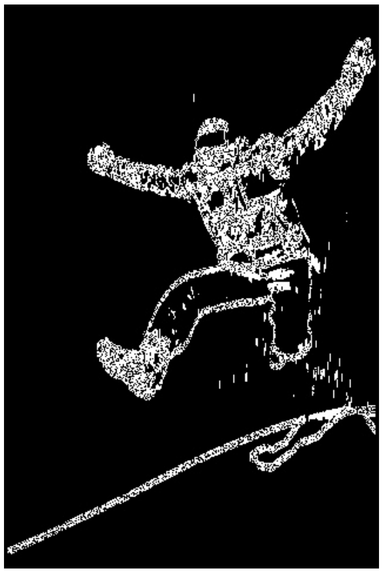

Figure 36: Foreground mask using window size **N = 7 x 7** with 1 iteration of Ostu's Method using **Texture based Segmentation**.

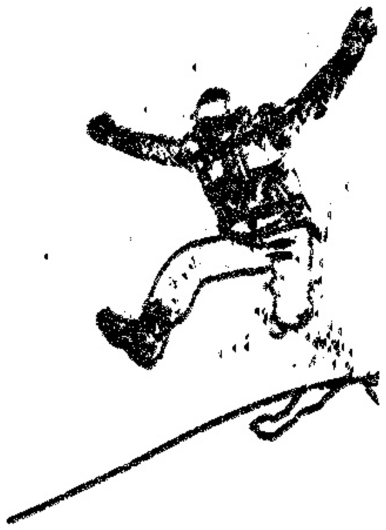

Figure 37: The overall foreground mask **with noise** using the masks of the **three window sizes** using **Texture based Segmentation**.

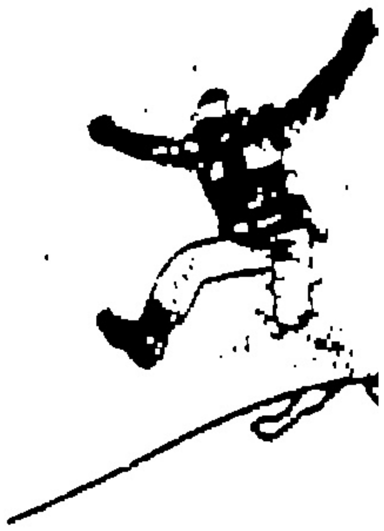

Figure 38: The overall foreground mask after **removing noise** using **erosion** and **dilation** for the **Texture based Segmentation**.

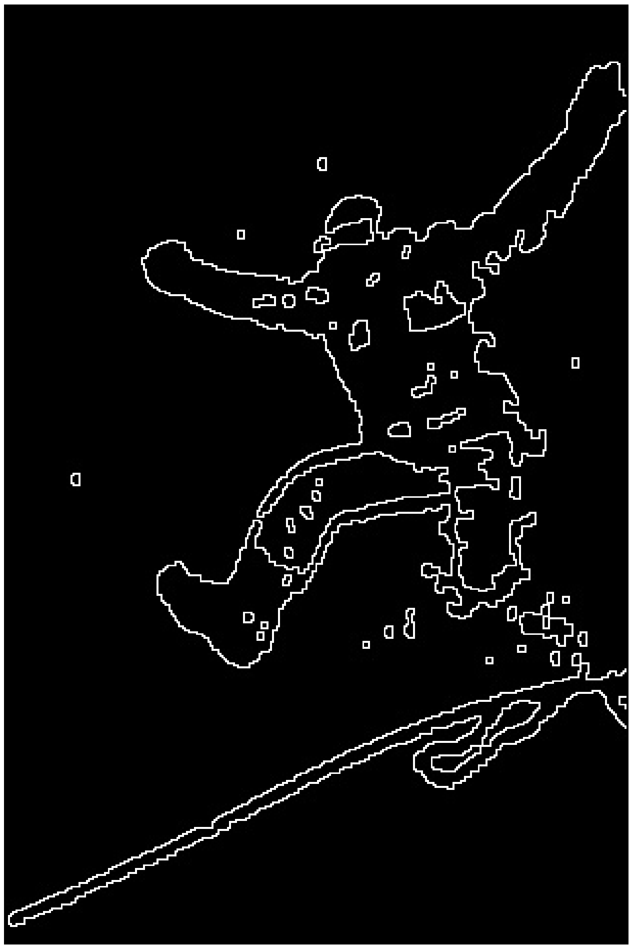

Figure 39: The final contour of the image for the **Texture based Segmentation**.