



## Extracting a Bounded Region from a Map Using Flood Fill Algorithm

S. M. Abdullah Al Shuaeb<sup>1\*</sup>, Md. Kamruzzaman<sup>2</sup> and Mohammad Hazrat Ali<sup>1</sup>

<sup>1</sup>Instructor (Tech) Computer Technology, Tangail Polytechnic Institute, Directorate of Technical Education, Tangail, Bangladesh.

<sup>2</sup>Workshop Super (Tech), Electronics Technology, Tangail Polytechnic Institute, Directorate of Technical Education, Tangail, Bangladesh.

### Authors' contributions

This work was carried out in collaboration among all authors. Author SMAAS designed the study and analyzed the algorithm and author MK managed the literature review and author MHA prepared the dataset for this work. All authors read and approved the final manuscript.

### Article Information

DOI: 10.9734/AJRCOS/2021/v7i130170

#### Editor(s):

(1) Dr. Stephen Mugisha Akandwanaho, University of KwaZulu-Natal, South Africa.

#### Reviewers:

(1) P. Natarajan, Vellore Institute of Technology, India.

(2) N. Sharmili, GVP College of Engineering for Women, India.

Complete Peer review History: <http://www.sdiarticle4.com/review-history/64295>

Original Research Article

Received 25 October 2020  
Accepted 29 December 2020  
Published 11 January 2021

### ABSTRACT

Extracting the needed portion from a bounded region is an important task in image processing. Editing a map and extracting a region from the map is challenging. It is useful in some contexts to have a region in a separate sheet. In this image processing, we have used the Flood Fill algorithm to extract a region from the image map. To achieve that goal, we had worked in our study to separate a bounded region on a map. Usually, a scanned map may contain a lot of useless information. So we have to process the image to remove useless information from the map. We had quantized the image to a binary one. In the second phase, we have applied a gray color to separate the desired position from a map. Our main objective of the study to extract a bounded region from mapping an image that contains useless information and removes it. We have experimented with several maps and it works successfully.

*Keywords: Image processing; map; extract; bounded region; flood-fill algorithm.*

\*Corresponding author: E-mail: [nixon.cse28@gmail.com](mailto:nixon.cse28@gmail.com);

## 1. INTRODUCTION

### 1.1 Prelude

Extracting a fixed area from an image is an important topic in computer vision and image processing fields. At present image processing has a very huge appliance and nearly all of the technical fields are dominated by image processing. In [1], an image consists of a two-dimensional array of numbers. The color or gray shade represented for a given picture element (pixel) rely on the number stored in the array for that pixel. The ordinary type of image data is black and white. It is a binary image since each pixel is either 0 or 1. The more complex type of image data is gray-scale, where each pixel takes on a value between zero and the number of gray-scales or gray levels that the scanner can record.

These images become visible like similar black-and-white photographs they are black, white, and shades of gray. Most gray-scale images today have 256 shades of gray. People can characterize about 40 shades of gray, so a 256-shade image. The most critical type of image is color. Color images are the same as gray-scale except that there are three bands, or channels, same to the colors red, green, and blue. Thus, each pixel has three values combined with it. A color scanner uses red, green, and blue filters to generate those values. Images are available via the Internet, scanners, and digital pictures. Any pictures shown on the internet can be downloaded by pressing the right mouse button when the pointer is on the image. This contracts the image to the user's PC usually in a JPEG format.

In this work, we have used the flood fill algorithm for extracting bounded regions from the original image map. A scanned map may contain a lot of useless information. Often the useless content interspersed with the main content. It's a great problem to use the map. We have processed the image of the map to remove useless information from that and extract the bounded region.

### 1.2 Motivation

The scientific system of image processing is concerned with the knowledge behind spurious systems that extract significant information from images. Extracting meaningful information from a map is so much interesting task in image processing. The image data can contain many forms, such as video sequences, views from

multiple cameras, 2D images, 3D scanners, various medical scanning devices. The technical order of image processing seeks to apply its knowledge and models to the working of computer vision systems. Image processing and image analysis tend to focus on 2D images, how to transform one image to another by pixel-wise operations such as grayscale conversion, contrast enhancement, local operations such as edge extraction or geometrical transformations removal, or extract meaningful information, and also remove unnecessary information. The characterization signifies that image processing.

### 1.3 Objectives the Work

- ✓ To extract the bounded region from a map.
- ✓ To remove unnecessary information from a map.
- ✓ To extract meaningful information from a map.

## 2. RELATED WORK

In [2], the authors compared to the Flood Fill algorithm, the edited Flood-Fill algorithm offers an important reduction in cell distance updates. With a lesser number of distances to update, the micro mouse which uses the modified flood fill can traverse from cell to cell at a higher speed. In [3], image processing image may contain a lot of noisy information. To avoid this noise, we have applied the flood-filling algorithm that begins from a known pixel in a bounded region and recursively identifies all the pixels within the adjacent region of the known pixel. To touch all pixels in each region by a 4-connected or 8-connected pixel a stack may be needed, and pixels within the region will be coming recursively. In [4], computer graphics there are many algorithms to fill the polygon. This work they show the comparative performance both flood-fill and boundary-fill algorithm. We have realized that both algorithms have optimized some real applications. It is suitable for filling large polygon. We also see that flood-fill algorithms represent efficient results than the boundary-fill algorithms in this study. In [5], we have been realized that to find the object using the flood-fill algorithm. Image segmentation is used to identify objects and boundaries in images. This paper shows the most identical regions identify using the flood-fill algorithm. The image is first segmented by mean shifted segmentation. The users only require to exactly identify the original features of objects. In [6], the flood-fill algorithm is used robotics path identification. FF dividing matrices allows and

development of path planning of mobile agents. This algorithm creates a short movement that is important to reduce the energy costs by the path of the robot.

### 3. METHODOLOGY

#### 3.1 Introduction

In this chapter, the algorithms and datasets are described. These algorithms are used to extract meaningful information from the map using the flood-fill algorithm. The map images of different types are provided as the dataset in these algorithms. A simple map image is presented below: -

#### 3.2 Proposed Work

The proposed system of an algorithm is shown here. To process an image, it is very much important to configure the OpenCV (open-source computer vision library). It can work with Visual C++. It can use the Intel Image Processing libraries (IPL). The configuration process of that interesting library is not so hard.

##### Algorithm

- |   |  |
|---|--|
| 1 | Read images from the dataset.  |
| 2 | Calculate the height, width, step width, and channel of the stored image.                  |
| 3 | Pre-process images like Make the gray-scale image completely black and white               |
| 4 | If (RGB < 100)<br>Then make the Color White.<br>Else<br>Make the color black.              |
| 5 | Apply the FloodFill Algorithm to select the expected region and change the color to black. |
| 6 | Set the path where want to save.   |
| 7 | Exit   |

#### 3.3 Overview of the Flood Fill Algorithm

In this section, we have described the Flood fill algorithm. It is useful in cases where there no single color boundary for the polygon that is the boundary has multiple colors. It is a very simple algorithm and efficient for correctness but not efficient large polygons. In [7], the flood fills algorithm instead of filling color till we appointment a specific boundary color we just fill the pixels with the default color. It is used function to fill connected, same-colored areas with a different color. In the flood-fill algorithm,

there are two methods in which the flood-fill algorithm can be implemented these are 4-connected pixel and 8-connected pixel. In this work, we have used the 4-connected pixel method and we check 4 pixels adjacent to the current pixel namely towards the top, bottom, left, right. So we fill the area with the 4-connected pixel method. The working procedure and figure are given below: -

##### 3.3.1 Working procedure:

**Step 1:** First initialization the 4 values namely x, y, fill-color, and default-color.

Where x and y are coordinated positions of the initial interior pixel we start the flood fill algorithm and fill-color is the color we want to fill default-color is the color of the interior pixel.

**Step 2:** if the color of the node is not equal to the default-color, return.

**Step 3:** Set the color of the node to the replacement-color.

**Step 4:** Set the color of the node to the bottom of the node to the replacement-color.

**Step 5:** Set the color of the node to the top of the node to the replacement-color.

**Step 6:** Set the color of the node to the left of the node to the replacement-color.

**Step 7:** Set the color of the node to the right of the node to the replacement-color.

**Step 4:** Exit.

The flood-fill (FF) also starting with seed within the region. It checks to see if the pixel has the region's original color. If the answer is yes, it fills the pixel with a new color and uses each of the pixels adjacent as a new seed in a recursive call. If the answer is no, it returns to the caller. FF method shares key similarities in its working rule with the boundary-fill algorithm.

### 4. RESULTS AND DISCUSSION

In this section, we have discussed the result of the task. The target of our experiment is to extract the bounded region we find out the image data and convert the gray-scale image into the black and white images. Experiment results are shown in different steps these are given below.

**Step1:** At first to display the image data we need an image as an input. This image is as follows:

After processing this image, we can see the height, width, step width, and channel on a DOS text window. Fig.5. shows the height, width, step width, and channel of the image of Fig.4.

**Step2:** For displaying the gray-scale Image completely black and white image with image data we use the image of Fig. 4. as an input. After processing this image, we can see the completely black and white image on the specified window. The specified window for this image is "BG". Fig. 4 shows the completely black and white image of Fig. 5.

**Step3:** For successful completion of extract the bounded region we use the image of Fig. 4. as an input. After processing this image, we can see the specified part of the image on the defined window "BG". We declared the part of the image using function cv\_Point (100, 100). Fig. 5. display the final processed image of Fig. 6.

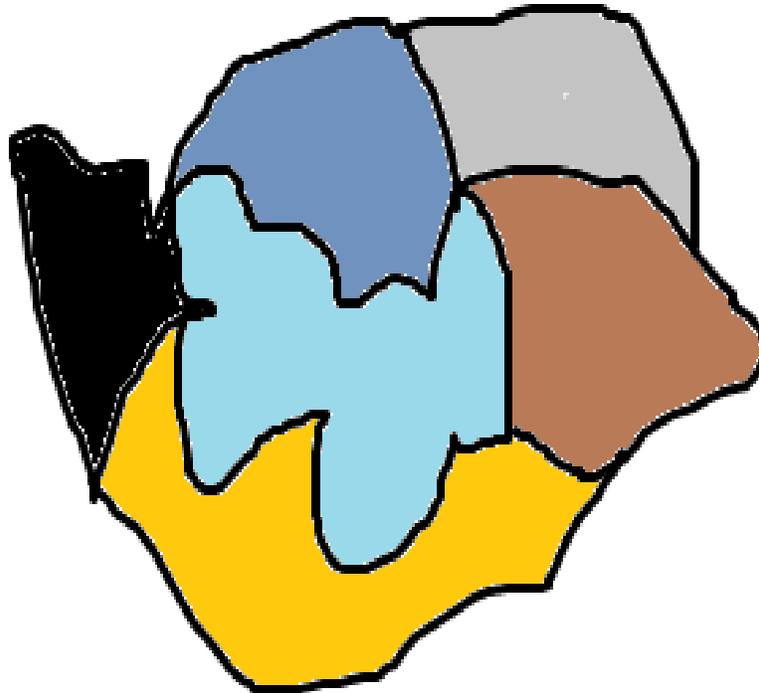


Fig. 1. Sample map image

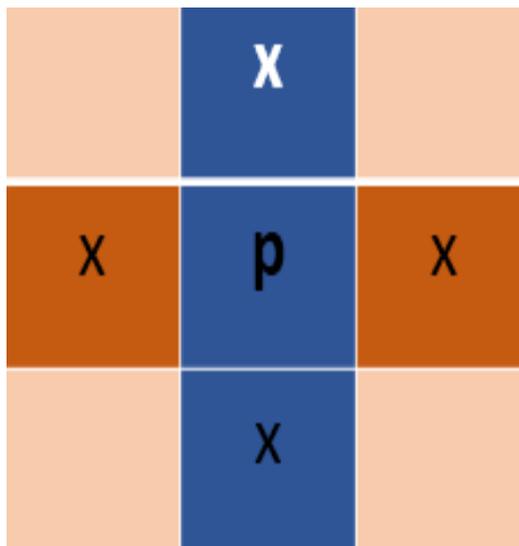


Fig. 2. 4-Connected pixel

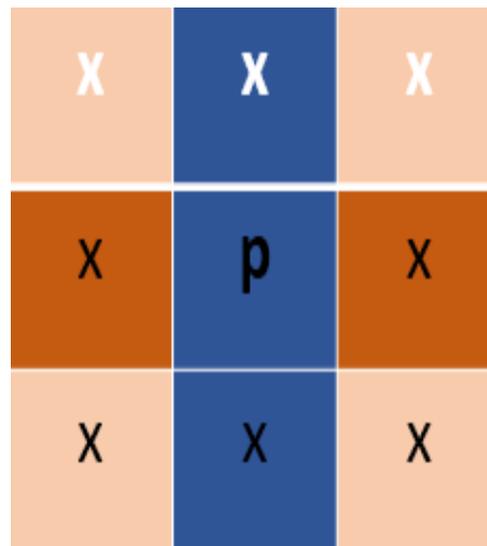


Fig. 3. 8-Connected pixel

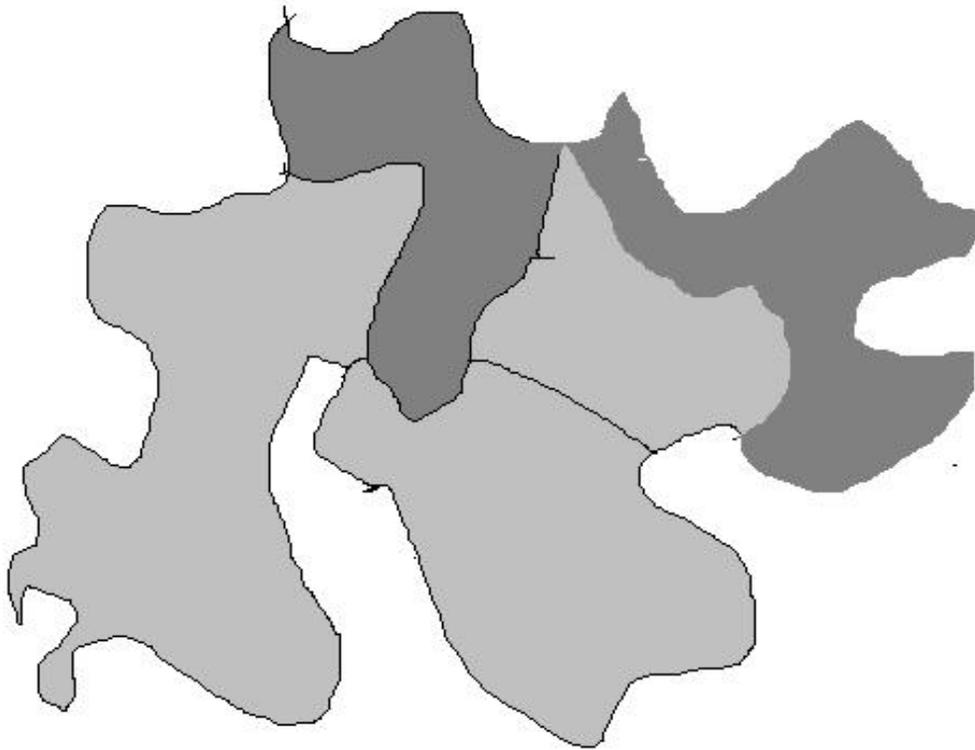


Fig. 4. Input image

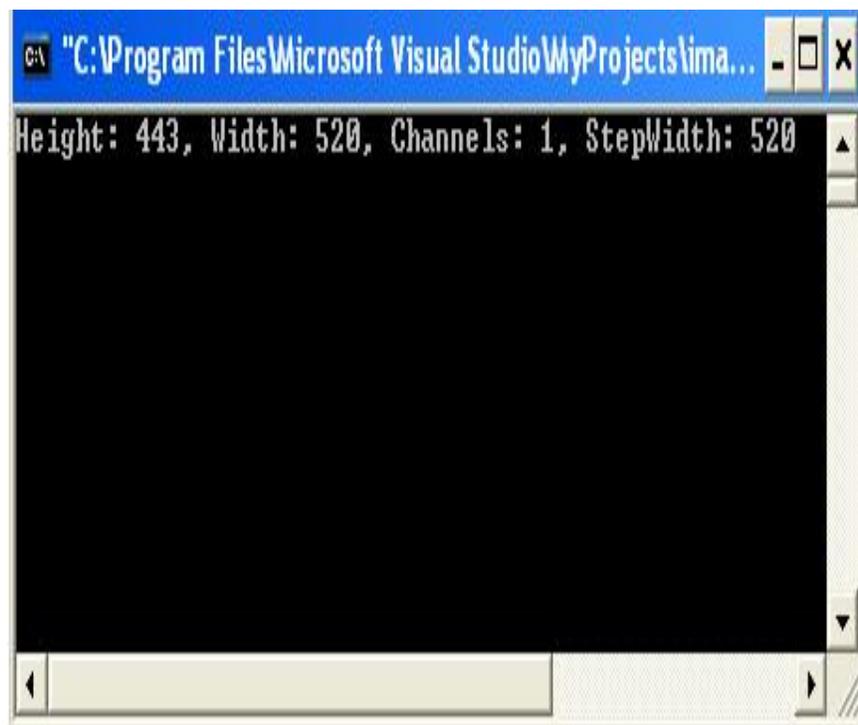
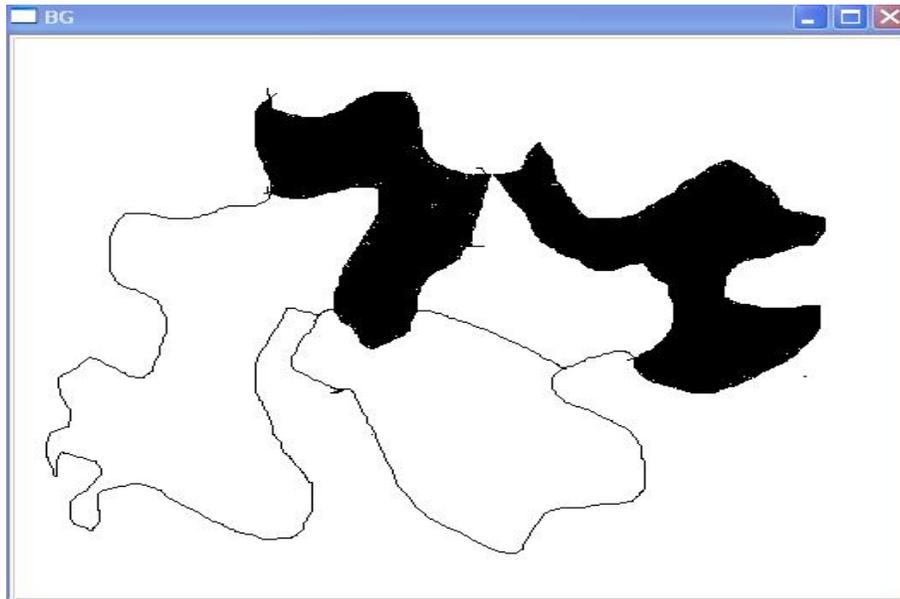
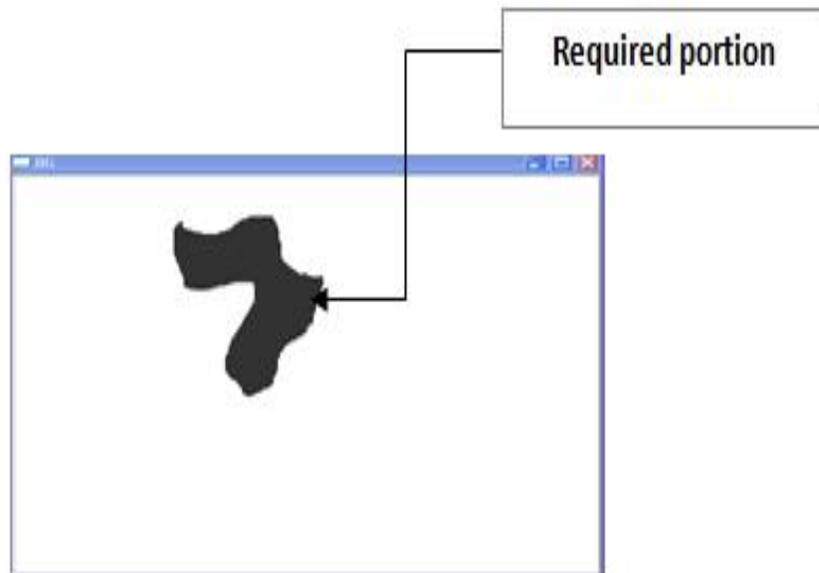


Fig. 5. The image data of image 4.1



**Fig. 6. The completely black and white image**



**Fig. 7. The final processed image**

## 5. CONCLUSION

In this work, we present an effective region filling algorithm for free regions by using a color scheme to assign different labels to adjoin 4-connected regions. The FF algorithm identifies through all pixels in the image, automatically sets the seeds for flood filling. At first, we use the method of reading the image data and display them on the DOS text window. Then make a

gray-scale image completely black. Then select the expected point from the inputted image and give this area a specific color applying the cvFloodFill method. After processing this image, we can see the processed image on the defined window. In our future, work we hope to utilize this concept in the color image. However, we think this algorithm to be competent in giving some probable potential indentation to overcoming the related graphical geometrical matter.

## ACKNOWLEDGMENT

The authors wish to thank Tangail Polytechnic Institute and its research lab for their helpful guidance and useful supports.

## COMPETING INTERESTS

Authors have declared that no competing interests exist.

## REFERENCES

1. Pan C, Tabatabaei Yazdi SMH, Tabatabaei SK, Hernandez AG, Schroeder C, Milenkovic O. Image processing in DNA. *Bio Rxiv*; 2019. DOI: 10.1101/2019.12.15.877290.
2. Law G. Quantitative comparison of flood fill and modified flood fill algorithms. *Int J Comput. Theory Eng.* 2013;5(3):503–508. DOI: 10.7763/ijcte.2013.v5.738.
3. He Y, Hu T, Zeng D. Scan-flood fill(SCAFF): An efficient automatic precise region filling algorithm for complicated regions. *Ar Xiv*; 2019.
4. Kumar B, Tiwari UK, Kumar S, Tomer V, Kalra J. Comparison and performance evaluation of boundary fill and flood fill algorithm. *Int J Innov Technol. Explor. Eng.* 2020;8(12s3):9–13. DOI: 10.35940/ijitee.I1002.10812s319.
5. Muthukumar K, Poorani S, Sindhu S. Color image segmentation using similarity-based region merging and flood fill algorithm. 2016;5(6):40–46.
6. Esteban J, Benavides H, Eduardo D, Corredor E. Flood fill algorithm dividing matrices for robotic path planning. 2018; 13(11):8862–8870.
7. Xiang Z. *Computer graphics - Schaum series.* pdf. 2002;342.

© 2021 Shuaeb et al.; This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

*Peer-review history:*  
*The peer review history for this paper can be accessed here:*  
<http://www.sdiarticle4.com/review-history/64295>