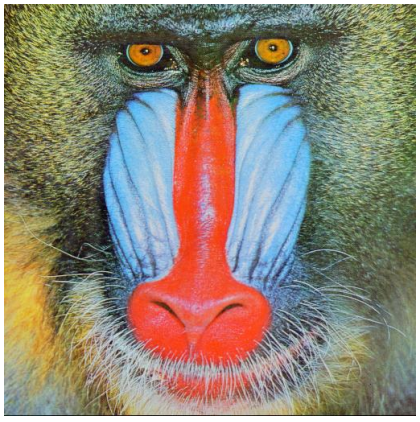
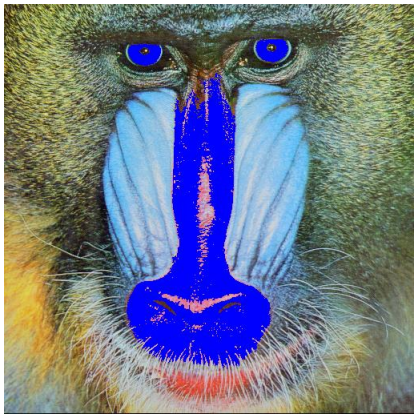




ELEC 474

Prelab 2 – Image Segmentation

Flood fill Tool - Region Growing

Kevin Huang and Michael Greenspan

Contents

1. Region Growing 1
2. Submission 3

1. Region Growing

For this prelab you will be implementing ***flood fill***, which is a region growing function that you have probably used before (e.g. also known as *paint fill* in Microsoft Paint). For this application, you will need to follow a variation of these steps:

1. Retrieve an initial ***seed pixel*** by selecting a pixel in your image, using a mouse callback function.
2. Create a ***stack*** to keep track of possible candidates to fill.
3. ***Push*** the seed pixel onto the stack as the initial input pixel.
4. ***Pop*** a pixel from your stack, which is the ***current pixel***.
5. Compare the value of the current pixel to the seed pixel, using a defined homogeneity criterion. If their values are similar enough, then the current pixel will join the region that is being grown, and so:
    a. Set the current pixel's value to the filled region color.
    b. Check the current pixel's neighbors, and add them to the stack.
6. Repeat steps 4, 5 and 6, until the stack is empty.

Notes:

A. Your application requires mouse clicks on your image. Use OpenCV's ***setMouseCallback()*** function to listen for mouse clicks in your OpenCV image window. The flood fill routine should be invoked directly through a mouse click.

B. Implement a trackbar that modifies an intensity threshold value, which will be used in the homogeneity criterion to determine membership in the filled region.

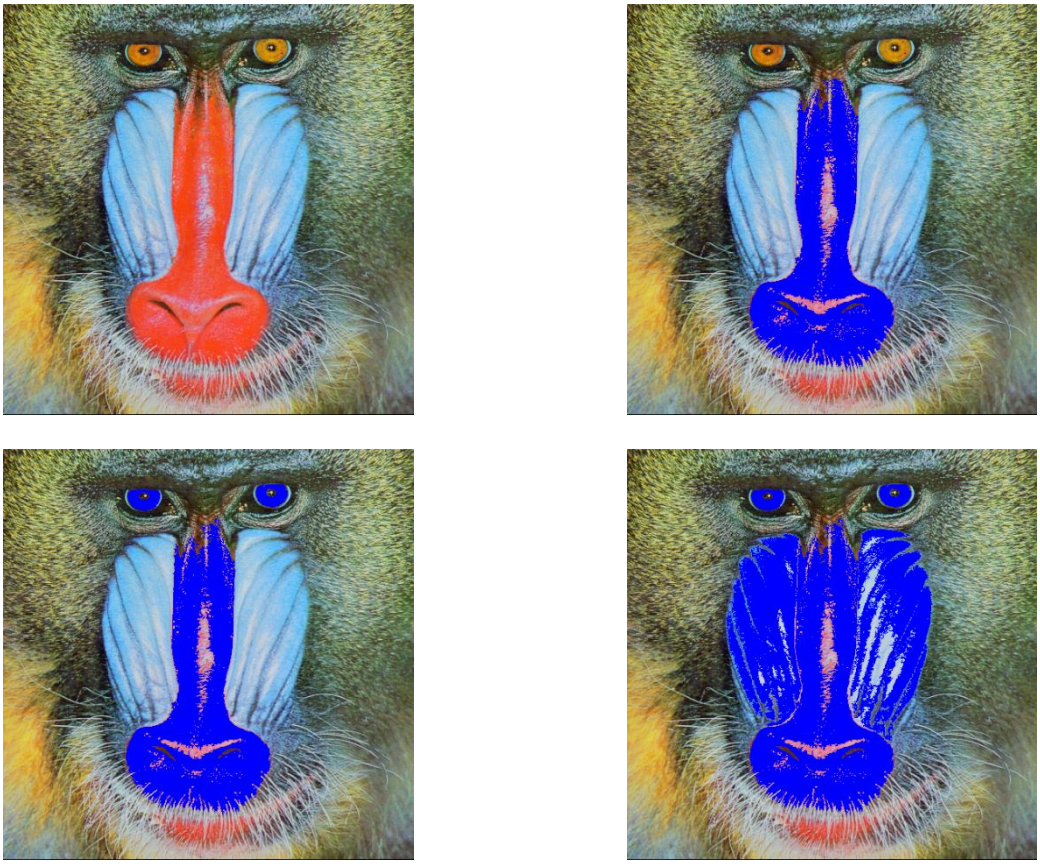
C. Use of global variables will be necessary for the mouse and trackbar callback functions.

D. Remember that your RGB image is being loading in as a BGR image in OpenCV. The intensity data type is ***uint8*** (unsigned int 8-bit) so they cannot be negative. Keep this in mind when performing any arithmetic on your intensity values.

E. In the above algorithm, you will have to make sure that you don't visit a pixel more than once, which can lead to an infinite loop. There are a few ways to do this, one of which is to keep a list or hash table of previously visited pixel values.

F. When expanding the neighborhood of the current pixel, make sure that you refer to image size, so that you don't try and access a pixel out of bounds.

G. Examples on how your application should work:



Baboon with blue flood filled regions at a color intensity threshold of 50

2. Submission

The submission for this lab should include a .zip file containing:

- An .ipynb file that includes:
    - Your code for flood fill, that runs on “baboon.png” and an image of your choosing.
    - Comments (in markdown cells) that indicate:
        - The type of neighborhood connectivity you applied.
        - The definition of your homogeneity criterion.
- At least one additional image of your choosing.

Your code will be run in Jupyter Lab to test for functionality.

When submitting your lab, make sure that you follow the formatting instructions that are listed on the onQ site under ‘Resources->lab submission instructions’.

The marking rubric is as follows:

Item	mark
1. Flood fill method correct and fully functional	1
2. Homogeneity criterion correct and threshold value adjustable with trackbar	1
3. Submission format correct	1
Total:	3