ESS201: C++ Programming

Jaya Sreevalsan Nair *
International Institute of Information Technology, Bangalore

Term I: 2017-18 (Project from 2017-11-15 to 2017-11-29)

The project will be done by pre-determined groups of 6 members each. Given are 8 tasks where each group member will perform a task, different from the rest of his/her group. Each group will be given a unique combination of tasks which need to be integrated to give a common application.

Goal: Your group will be building a small-scale image processing toolkit.

Tasks: Assume `input.ppm` is a file input to the running of the toolkit. The input image is of size, `image_width x image_height`.
Compositing two images. A and B, means creating a new image by linear combination of the two given images. Compositing A with B gives $C = (1 - \alpha) * A + \alpha * B$, where floating point value $\alpha \in [0, 1]$. Create a class for `CompositeImage` which *is-a* `Image`, thus using inheritance.

The list of tasks are as follows (the number given in the beginning of each item is the corresponding task ID):

1. (a) Generate an image of size `(image_width x image_height)` of the geometric primitive, circle. The center of the circle must be `floor(image_width*0.5),floor(image_height*0.5)`. Use `min(floor(image_width*0.125), floor(image_height*0.125))` as the radius of the circle.

   (b) Any pixel distant from the center by the radius (in units of number of pixels) *or lesser* is to be considered black or else white (the circle is thus a filled primitive).

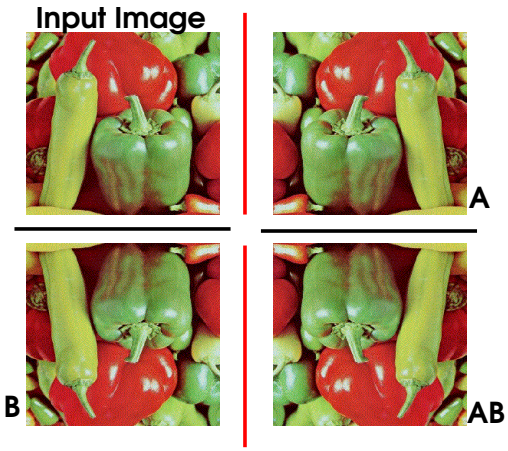
   (c) Construct similar circles in the image in the center of the four quadrants of the images (created using x- and y- axes intersecting at the center of the image). With every new addition of the circle, only the pixels that have to be colored white, should change the corresponding pixel color from black to white.

   (d) Now, combine the input image with the generated image of the circle to *clip* the input image, i.e. retaining only pixels of the input image that coincide with the filled circle. The rest of the image is white. Say, you get an image C.

   (e) Perform operations #1-a to #1-d, substituting the input image with its reflection, A, shown in Figure 1-(a). Except, now the clipping is like stenciling, where the filled primitive stays white, and the rest of the image retains values from the reflection A of the input image. Say the new image is D. The intended result of clipping and stenciling are shown with a single circle in Figure 2.

   (f) Composite C with D with $\alpha = 0.5$.

*(jnair@iiitb.ac.in)



(a)

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

(b)

Figure 1: (a) Reflection of an image about y axis is A, and that about x axis is B. (b) Gaussian filter applied to a pixel using its 8-neighborhood, with $\sigma \approx 0.1$. (Image courtesy: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm)

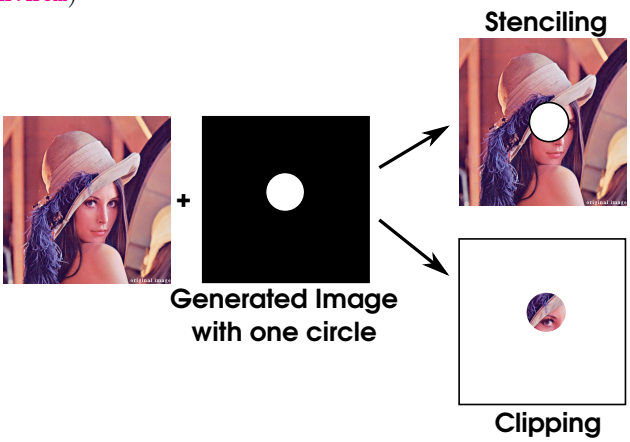


Figure 2: Intended results for clipping and stenciling when using a generated image with one circle.

2. (a) Consider the reflection A of an image about y axis, as shown in Figure 1-(a).
   (b) Find the set of pixels which have local minima in 8-neighborhood of blue component in A, say L.
   (c) Use the pixel locations in L to generate an image C of the same size as the input image, where pixels in C corresponding to the locations in L and its respective 24-neighborhoods are colored white, whereas all other pixels are colored black. RGB value for white is (255,255,255), and black is (0,0,0) (If there are pixels which are common in neighborhoods of multiple locations in L, set them to be white).
   (d) Apply Gaussian filter (Figure 1-(b)) to locations in L in C (For overlapping regions, take average of the values from the different Gaussian filters). In our case, "applying Gaussian filter" implies that the color of a pixel colored white currently is the product of the corresponding weight, as shown in Figure 1-(b), and the color white.
   (e) Composite C with A, with $\alpha = 0.75$.

3. (a) Take the reflection B of an image, which is about x axis, as shown in Figure 1-(b).

   (b) For each pixel in the image, convert RGB to HSV color model (conversion formulae are given in https://en.wikipedia.org/wiki/HSL_and_HSV).

   (c) Find the pixels which are local maxima in its 8-neighborhood, for the value of saturation (in HSV model). Let the set of local maxima pixels be L.

   (d) Use the locations in L to generate an image C of the same size as the input image, where pixels in C corresponding to the locations in L and its respective 24-neighborhoods are colored white, whereas all other pixels are colored black. RGB value for white is (255,255,255), and black is (0,0,0) (If there are pixels which are common in neighborhoods of multiple locations in L, set them to be white).

   (e) Apply Gaussian filter (Figure 1-(b)) to locations in L in C (For overlapping regions, take average of the values from the different Gaussian filters). Follow #2-(d) on how to apply Gaussian filters.

   (f) Composite C with B, with $\alpha = 0.6$.

4. Perform segmentation of a color (input) image in the form of compositing.

   (a) Compute the brightness of each of the pixel of the input image, which is the arithmetic mean of R, G, B components.

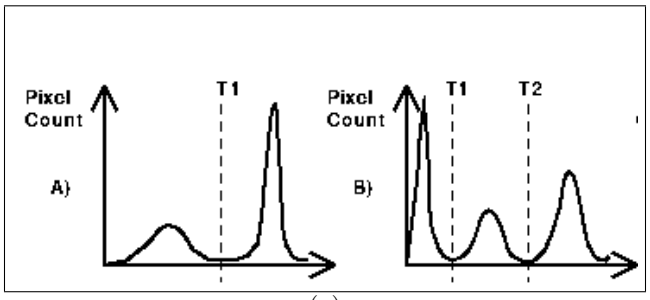
   (b) Construct histograms as shown in Figure 3-(a). Find *thresholds* at local minima in the pixel count in histogram, shown as $T1, T2$, etc.

   (c) Depending on the number of segments (made by the thresholds) along the x-axis, assign color values (red, green, blue, yellow, magenta, cyan) in the order of segments. Suppose we have two thresholds $T1$ and $T2$, we consider brightness values $[0, T1]$ to be red, $[T1, T2]$ to be green, and any pixels with brightness greater than $T2$, we associate blue, and so on. (Consider building new classes for histogram.) Color the remaining pixels, i.e. pixels which do not belong to any of the connected components, black (if your algorithm considers the "remaining pixels" to be one of the connected components, then color that component black.).

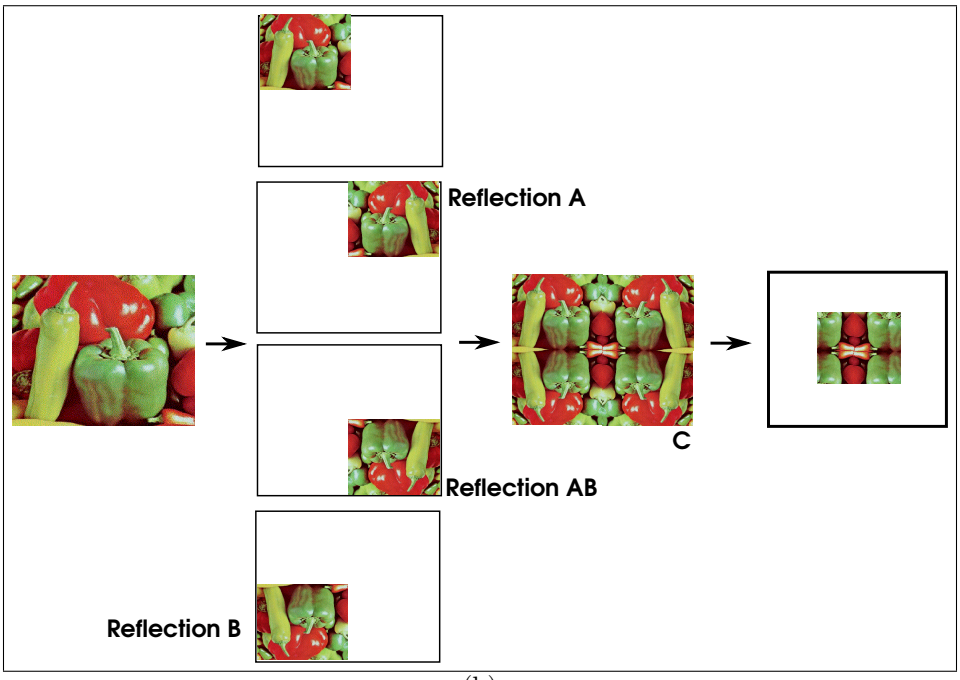
   (d) Construct an image C corresponding to location in the different segments in histogram of the input image, and its corresponding colors.

   (e) In the absence of thresholds, consider reflection A (Figure 1-(a)) of the input image as C.

   (f) Composite the input image with C, with $\alpha = 0.4$.



(a)



(b)

Figure 3: (a) Thresholds of $T1, T2$, etc. in histograms for pixels. In our case, we take x-axis to be brightness. (b) Scaling by subsampling by 0.5 (skipping every other x and y pixels) and reflection to give a final image by compositing.

5. (a) Construct brightness histogram of the input image, as given in #4.
   (b) Consider the first threshold $T1$ alone, construct C which is a binary of the input image, i.e. pixels with brightness $< T1$ will be colored blue, and the rest white.
   (c) Perform similar operation to construct an image D, where threshold $T2$ is considered exclusively, and blue is substituted by yellow.
   (d) In the absence of $T1$ or $T2$ or both, consider A and B reflections of the input images given in Figure 1-(a) as C and D.
   (e) For the output, composite D with C, with $\alpha = 0.2$.

6. (a) Construct brightness histogram of the input image, as given in #4 and identify thresholds.
   (b) Consider the median thresholds T (i.e. median of all thresholds $T1, T2, T3, \ldots$), construct C which is a binary of the input image, i.e. pixels with brightness $< T1$ will be colored black, and the rest white.
   (c) Find connected components labels as described in https://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm.
   (d) Number the labels in sequence from 0, and associate the labels to an array of colors, using an `enum` for (red, green, blue, cyan, magenta, yellow). Add more distinct colors, other than black, to this list to accommodate connected components more than six.
   (e) Output the final image where pixels in connected components with its associated colors based on labels, and the remaining pixels black (if your algorithm considers the "remaining pixels" to be one of the connected components, then color that component black.).

7. (a) From an input image, construct four images which can be composited by scaling and reflection as shown in Figure 3-(b). Construct derived class, ScaledImage from Image, which is to be used for generating the four images, and use $\alpha = 0.5$.
   (b) Note that all the six images in Figure 3-(b) have the same size, i.e. `(image_width x image_height)`. The quadrants not having transformed (scaled and reflected) images of the input image are all colored white.
   (c) Combine the four images taking pairwise by substituing the white pixels with the transformed images. The final image is as shown in Figure 3-(b).
   (d) Now repeat the clipping in #1, with a single square primitive of size of `min(floor(image_width/2), floor(image_height/2))` to give the final image C.

8. (a) Add noise to the input image. Generate noise image using `rand()` function.
   (b) Composite noise image with input image using $\alpha = 0.5$.
   (c) Now, find the brightness histogram and thresholds, as described in #4. Use the first threshold, $T1$ to binarize the image, as given in #6.
   (d) Find at least one connected component label, as described in https://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm.
   (e) Number the labels in sequence from 0, and associate the labels to an array of colors, using an `enum` for (red, green, blue, cyan, magenta, yellow). Add more colors in case of finding all connected components, where the number of connected components is > 6. Avoid using black as the label color.
   (f) Construct the final image where the pixels in connected components are colored as per their label color. Color the remaining pixels, i.e. pixels which do not belong to any of the connected components, black (if your algorithm considers the "remaining pixels" to be one of the connected components, then color that component black.).

The final image in each of the task must be outputted as .ppm file.

Integration: As a starting point, use the best class interface and implementation (developed in the preceding assignments) available amongst the group members.

The application takes in an image in the .ppm format (as has been done in Assignments 5 and 6). So, the program is to be run as follows:

```
./a.out input.ppm
```

Once the file reading is done, a user of the application should get a notification:

```
Our group task ID combination is: a,b,c,d,e,f
```

where, `a,b,c,d,e,f` are numerical values between 1 and 8, and together the six IDs give the unique combination of IDs of tasks for your group, as mentioned above.

Once the chosen tasks are completed (i.e. executed completely), one can exit the program. The completion of each task must output an image:

```
output_task<ID>_imt<group-member-rollnumber>.ppm
```

Ensure the output files are submitted along with your project source files. The output files must be named correctly, where the contents in angular brackets must be updated with numeric values appropriately. This output file will be evaluated along with your individual demo performances for your exclusive assessment. The file name should not contain any alphabet in capital.

One must be able to re-modify the group task ID combination to run a single task, or any other number of tasks, too. e.g.:
`Our group task ID combination is: a`

Integration must involve combining common classes required for performing more than one tasks, as well as removing redundant classes.

Submission: Submit on LMS a tar-zipped folder of the header files, implementation/source files, and individual output files. The input .ppm need not be included in the folder. The folder is to be named:

```
cpp_project_group_<number>
```

Write a README.txt indicating group and individual contributions. When mentioning individual contributions, tabulate name and roll number of student and the respective task ID. Include this file in the folder. **In the absence of the README.txt, we will not consider the individual contributions.**