Image Processing

Presented BY:

Murali Krishna MOPIDEVI

Arjun SINGH



Contents

[**Introduction to Quadtree:-** 2](#_Toc27083364)

[**Introduction to portable pixmap format (PPM) Image Format:-** 3](#_Toc27083365)

[**Basic Functions:-** 4](#_Toc27083366)

[ **Image Read & Process** 4](#_Toc27083367)

[ **Mirror** 4](#_Toc27083368)

[ **Color Reversal** 5](#_Toc27083369)

[ **Rotation:** 5](#_Toc27083370)

[**Performed some Filters on JPG Image:-** 6](#_Toc27083371)

[**Packages and Scala Version Used:-** 9](#_Toc27083372)

[**Advantages:-** 9](#_Toc27083373)

[**Disadvantages:-** 9](#_Toc27083374)

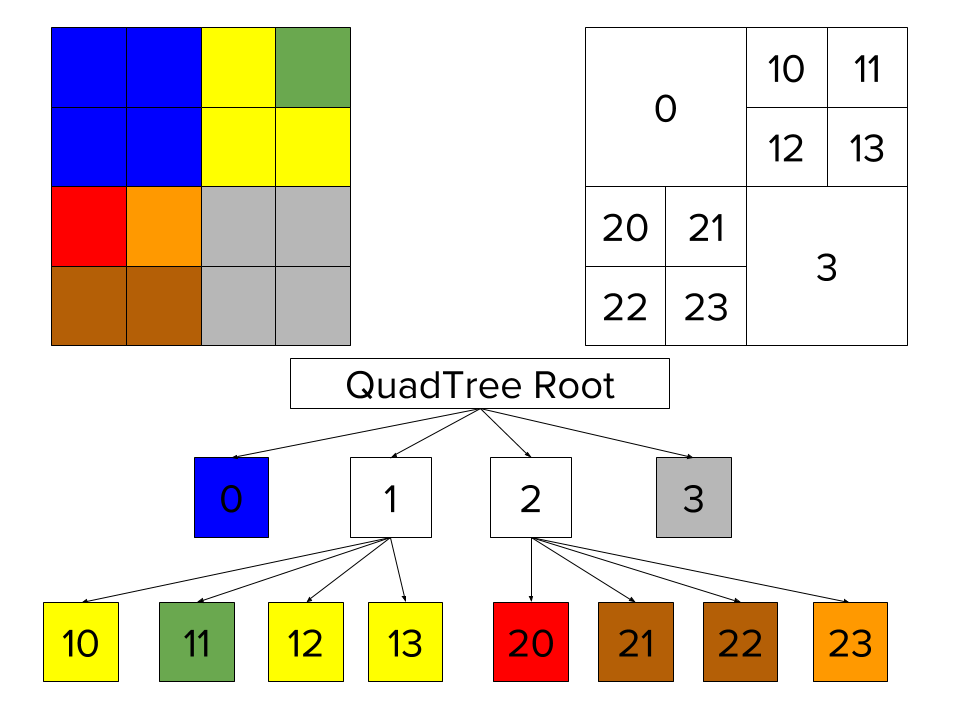
Image Processing with Quadtree

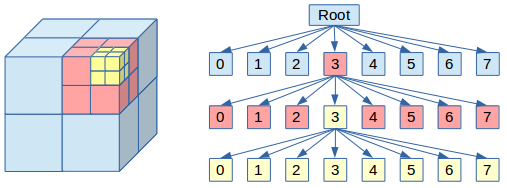
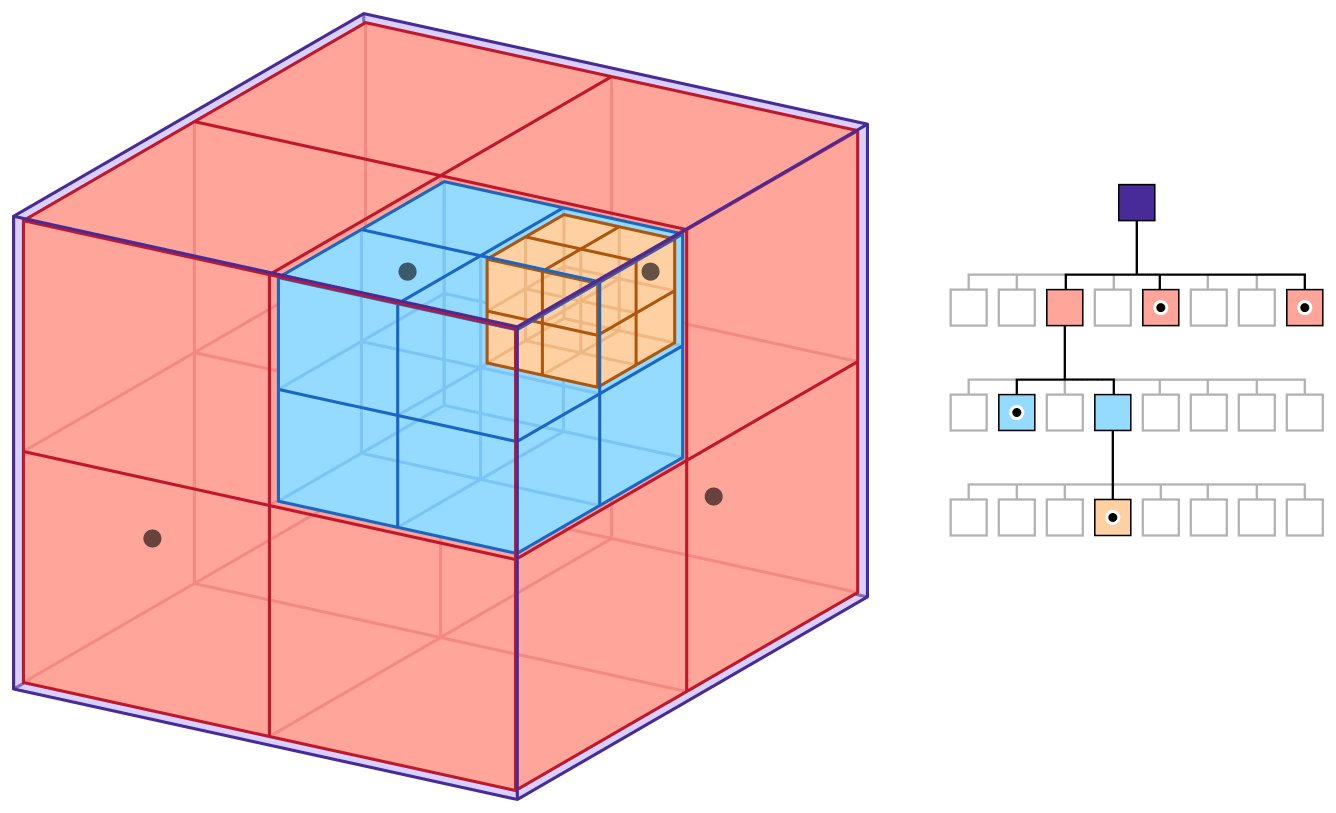
FUNTIONAL PROGRAMMING in SCALA

ADEO-II 2019-2020

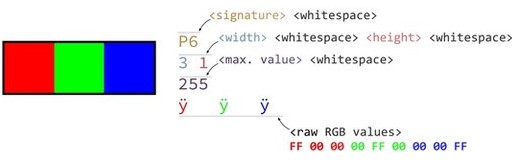
**ARJUN SINGH  
MURALI KRISHNA MOPIDEVI**

# **Introduction to Quadtree:-**

The quadtree, visually, often starts with a square spatial field. The field is then split into four smaller, consistent squares, then each of those squares is separated into four, etc. The result can be useful for data modeling. One example is in image handling, where an image can pixelate through a quadtree: first, the four largest squares get color, then the corresponding next-level set of sixteen squares each get their own color, and so on. The result is a clean and consistent way of pixelating an image that may take significant resources to load in a system. Other applications of a quadtree involve state analysis or other kinds of data analysis.



# **Introduction to portable pixmap format (PPM) Image Format:-**

A PPM file consists of two parts, a header and the image data. The header consists of at least three parts normally delineated by carriage returns and/or linefeeds but the PPM specification only requires white space. The first "line" is a magic PPM identifier, it can be "P3" or "P6" (not including the double quotes!). The next line consists of the width and height of the image as ASCII numbers. The last part of the header gives the maximum value of the color components for the pixels, this allows the format to describe more than single byte (0..255) color values. In addition to the above required lines, a comment can be placed anywhere with a "#" character, the comment extends to the end of the line.

Example:-

4 4

15

0 0 0 0 0 0 0 0 0 15 0 15

0 0 0 0 15 7 0 0 0 0 0 0

0 0 0 0 0 0 0 15 7 0 0 0

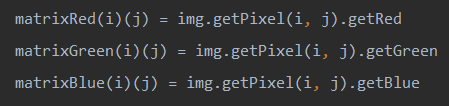
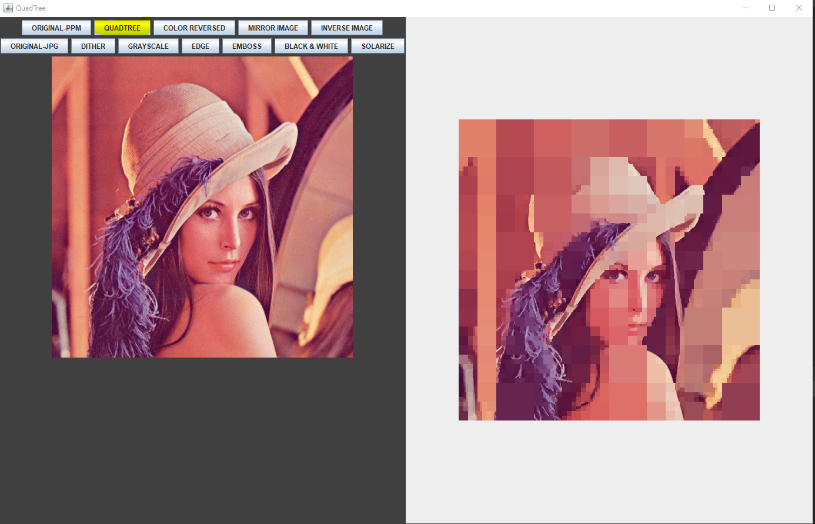
1. 0 15 0 0 0 0 0 0 0 0 0

# **Basic Functions:-**

## **Image Read & Process**

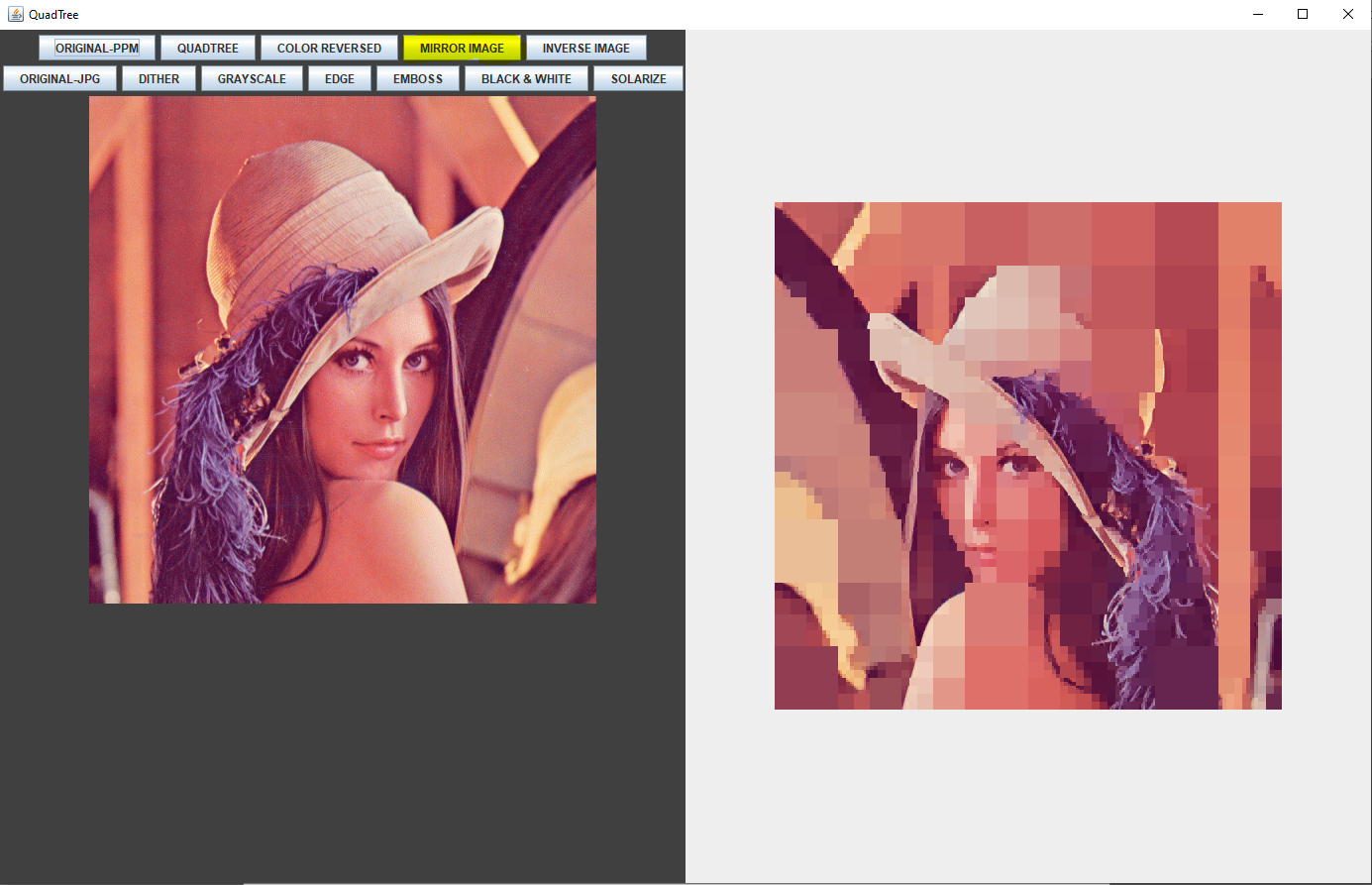
Charge an image from a .ppm file (portable pixmap file format) into a colored  
matrix. From this matrix, generate the corresponding quadtree.

* Save an image : from a quadtree and a desired image size, generate the  
  corresponding colored matrix. Save this matrix into a ppm file.

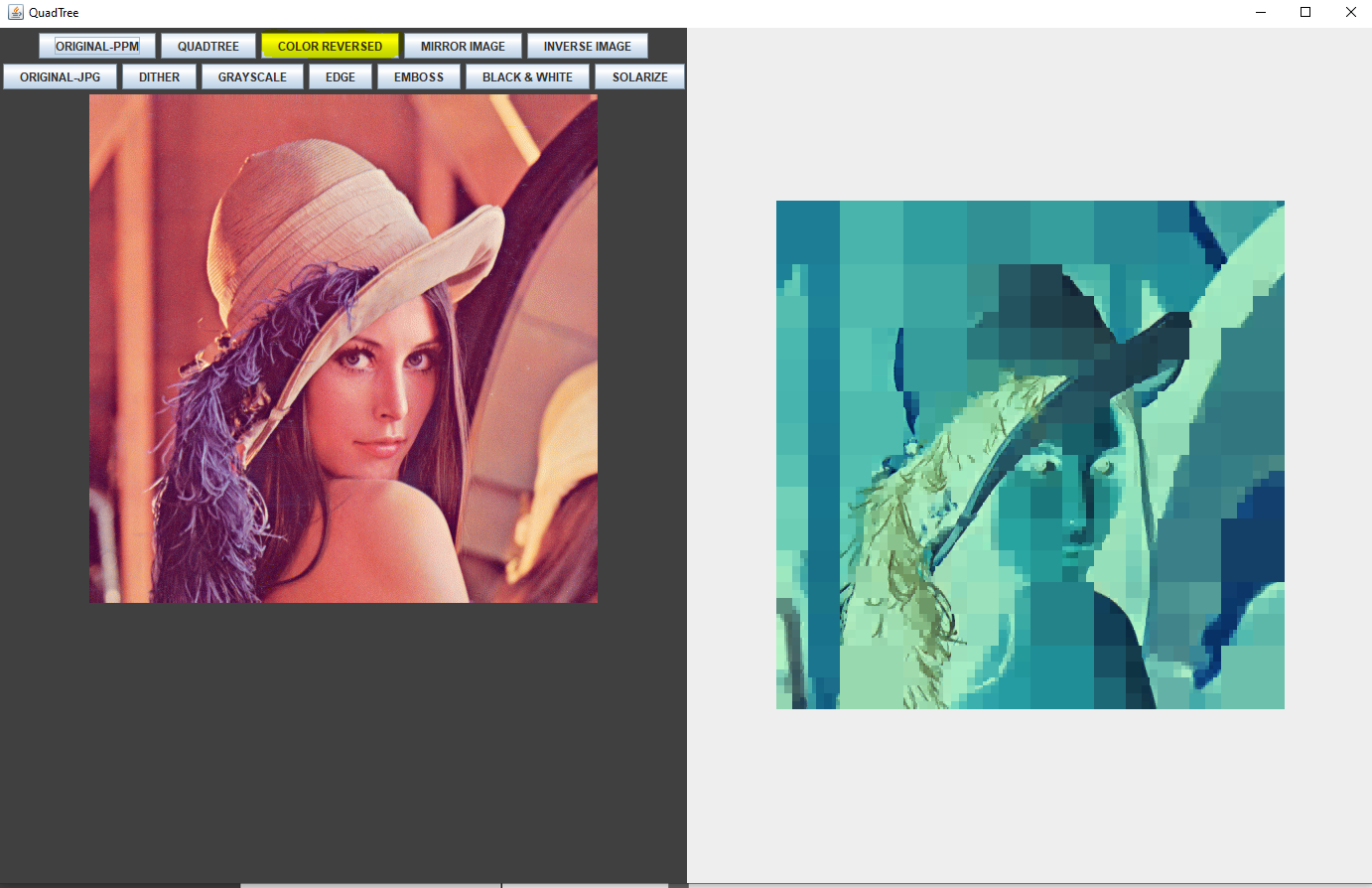
****

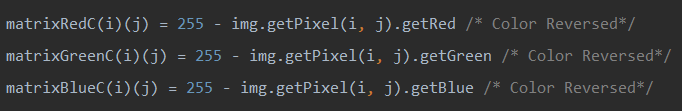
* **Mirror** :- reversal left-right and top-bottom.



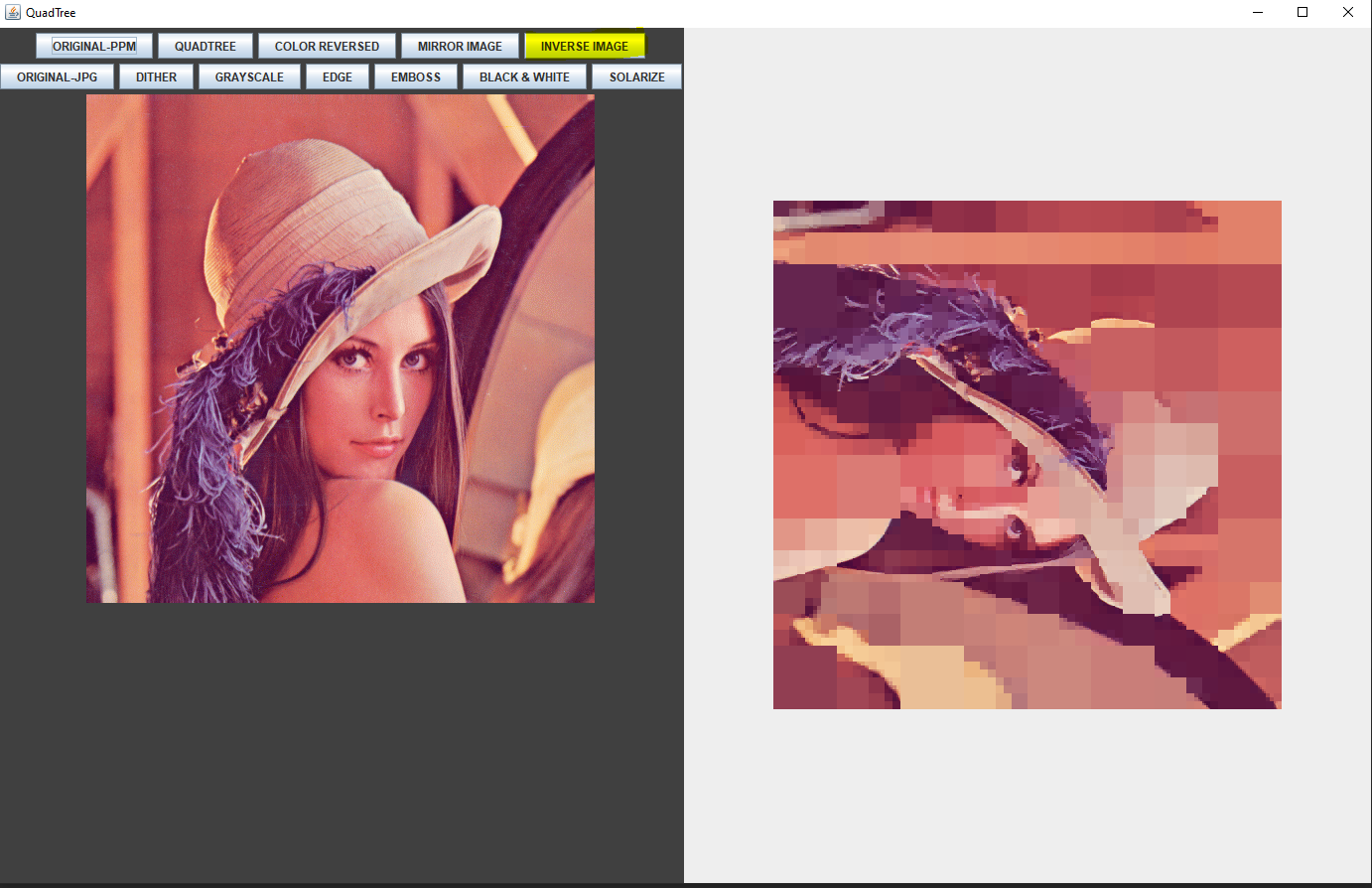


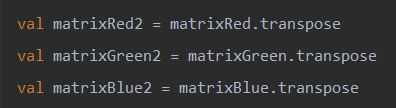
## **Color Reversal**





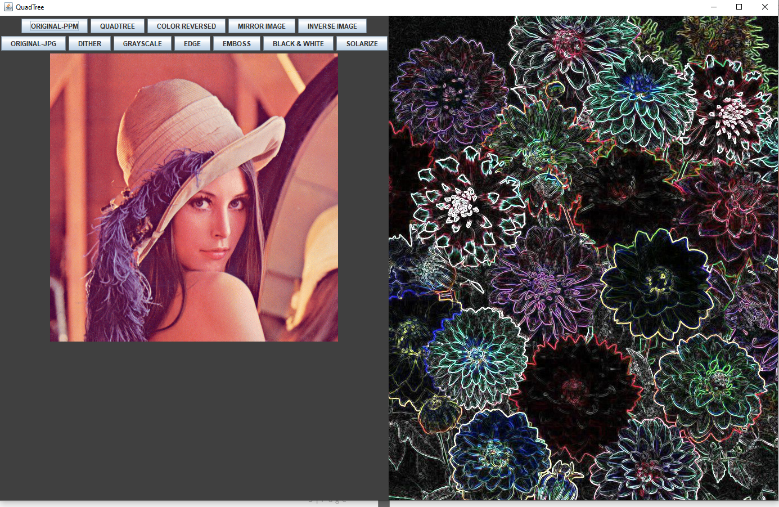
* **Rotation:** 90° clockwise or counter-clockwise

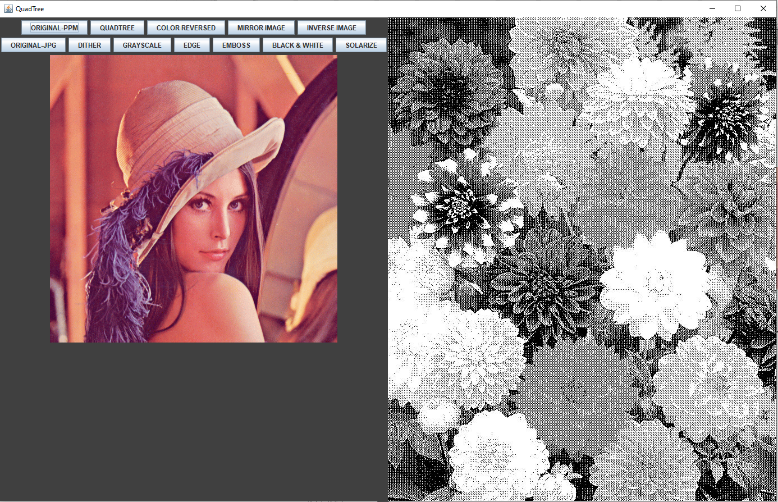


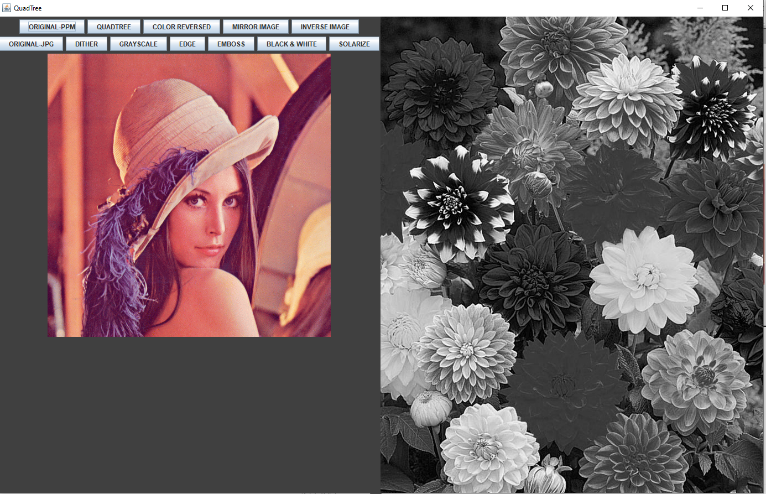


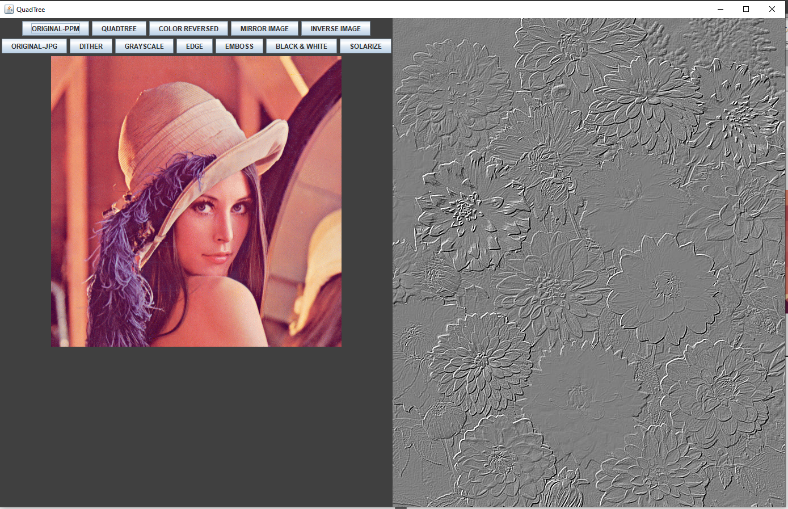
# **Performed some Filters on JPG Image:-**

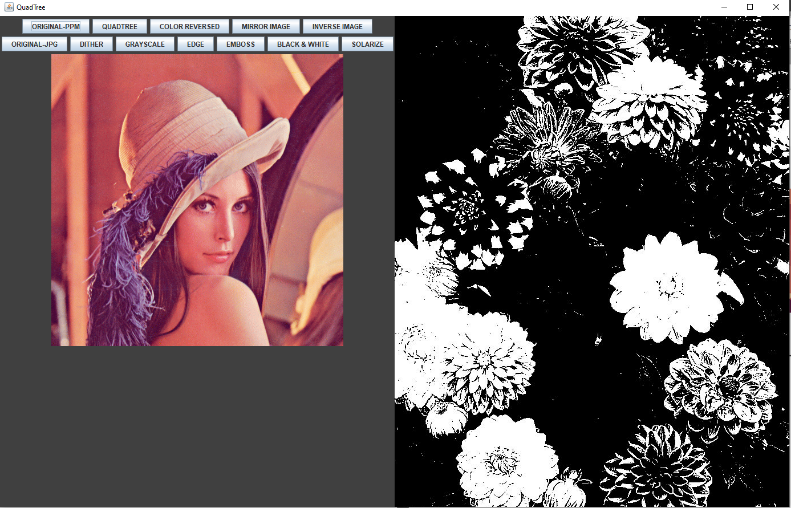
Here we performed some filters on an image using scala packages.  
Filters like GRAYSCALE, DITHER, BLACK and WHTE, SOLARIZE, EDGE and EMBOSS.

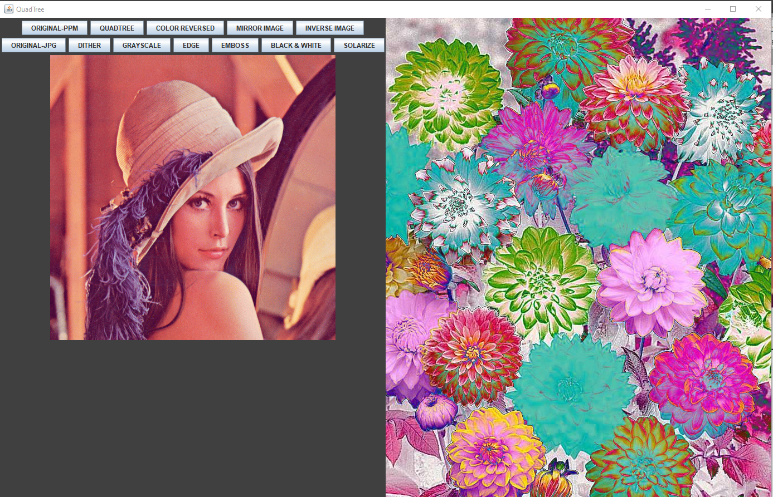












# **Packages and Scala Version Used:-**

* Scala 2.12.0
* SBT 1.3.4
* IntelliJ IDE (Optional)
* Scrimage package 2.1.8
* Java 1.8.

# **Advantages:-**

* GUI design for user friendly access to the project.
* Supporting (N X N) size of PPM image.
* Observing the neighboring pixel to same Pixel type
* Combining same pixel together.

# **Disadvantages:-**

* Its not supporting other image formats
* We can process Quadtree to compressing the size of image
* We can implement GUI to select an image and perform operations.
* The main disadvantage of Quadtrees is that it is almost impossible to compare two images that differ only in rotation or translation.

Example :-

