Level Adjustment

Black/White Adjustment

Ilie Sebastian Stefan

333AB

Cuprins

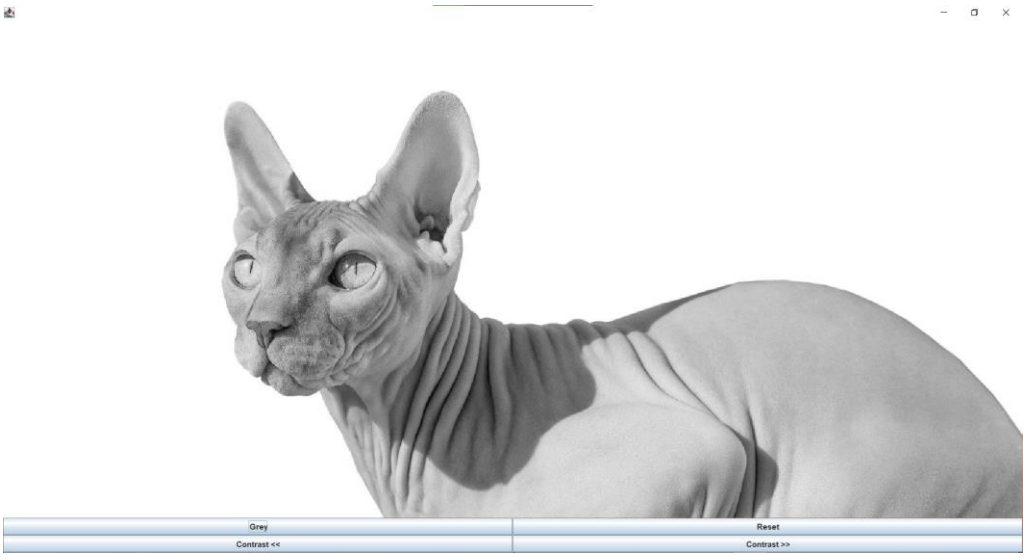
1.Introducere

Lucrarea are ca scop prelucrarea de imagini, mai exact ajustarea nivelului contrastului sau luminozității (alb/negru) acestora.

2.Exemplu functionalitate

Imagine aplicata:

Imagina obtinuta:

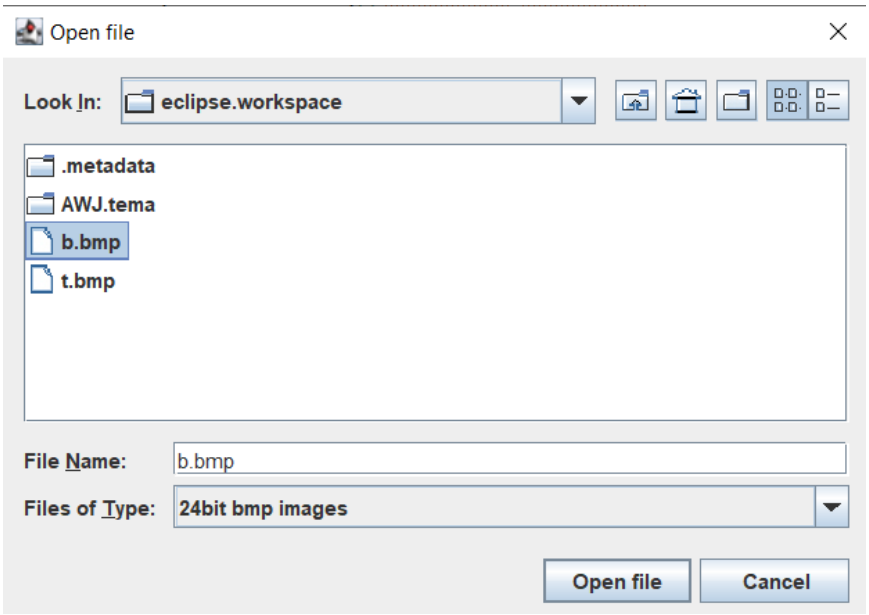


3.Descriere functionare program:

Am folosit librariile java.awt si javax.swing pentru interfata grafica a programului. De asemenea este folosita libraria java.io.File pentru a putea accesa fisierele locale din sistemul calculatoarului.

```
1  import javax.swing.*;
2  import javax.swing.filechooser.FileNameExtensionFilter;
3  import java.awt.Color;
4  import java.awt.color.ColorSpace;
5  import java.awt.*;
6  import java.awt.event.*;
7  import java.awt.image.*;
8  import java.io.File;
9
```

Pentru citirea fisierului am folosit clasa deja existenta in java: JFileChooser care deschide o fereastra de dialog in care utilizatorul trebuie sa aleaga fisierul pe care va rula programul. De remarcat faptul ca se filtreaza rezultatele astfel incat sa apara doar fisierele bmp. In urma acestui proces se obtinea calea catre imaginea ce trebuie editata. Imaginea este salvata si verificata, existand de asemenea o exceptie pentru evitarea blocarii programului.



```
107    public void loadImage() {
108
109
110        JFileChooser fileopen = new JFileChooser();
111        FileNameExtensionFilter filter = new FileNameExtensionFilter("bmp images", "bmp");
112        fileopen.setFileFilter(filter);
113        int ret = fileopen.showDialog(null, "Open file");
114
115        if (ret == JFileChooser.APPROVE_OPTION)
116        {
117          File selectedFile = fileopen.getSelectedFile();
118          path = selectedFile.getAbsolutePath();
119        }
120
121        displayImage = Toolkit.getDefaultToolkit().getImage(path);
122        MediaTracker mt = new MediaTracker(this);
123        mt.addImage(displayImage, 1);
124        try {
125            mt.waitForAll();
126        } catch (Exception e) {
127            System.out.println("Exception while loading.");
128        }
129
130        if (displayImage.getWidth(this) == -1) {
131            System.out.println("No bmp file");
132            System.exit(0);
133        }
134    }
```

Este creata o variabila de tip BufferedImage deoarece cele de tip Image nu pot fi prelucrate.

```
136    public void createBufferedImages() {
137        biSrc = new BufferedImage(displayImage.getWidth(this),
138                                  displayImage.getHeight(this),
139                                  BufferedImage.TYPE_INT_RGB);
140
141        big = biSrc.createGraphics();
142        big.drawImage(displayImage, 0, 0, this);
143
144        biDest = new BufferedImage(displayImage.getWidth(this),
145                                   displayImage.getHeight(this),
146                                   BufferedImage.TYPE_INT_RGB);
147        bi = biSrc;
148    }
```

Clasa Modify care mosteneste clasa JFrame construieste butoanele necesare programului alaturi de metodele apelate la apasarea lor(ActionListener).

Pentru modificarea contrastului imaginii este folosita functia RescaleOp din metoda rescale(). Deoarece se doreste modificarea treptata a contrastului, la fiecare apasare a butonului se modifica parametrul scaleFactor cu 0.1f, reducand sau crescand contrastul imaginii.

Clasele folosite in acest proiect sunt:

- Modify care mosteneste JFrame
  + Modify();
- ButtonListener care implementeaza interfata ActionListener
  + public void ActionPerformed(ActionEvent);
- DisplayPanel care mosteneste JPanel
  + DisplayPanel();
  + public void loadImage();
  + public void createBufferedImage();
  + public void changeScaleFactor();
  + public void rescale();
  + public void update (Graphics);
  + public void paintComponent (Graphics);
  + public void grayOut ();
  + public void reset();
- WindowEvenetHandler care mosteneste WindowAdapter
  + public void windowClosing(WindowEvent);

```
150    public void changeScaleFactor() {
151        if (contrastInc) {
152            if (scaleFactor < 2)
153                scaleFactor = scaleFactor+0.1f;
154        }
155        else {
156            if (scaleFactor > 0)
157                scaleFactor = scaleFactor-0.1f;
158        }
159    }
160
161    public void rescale() {
162        rescale = new RescaleOp(scaleFactor, offset, null);
163        rescale.filter(biSrc, biDest);
164        bi = biDest;
165    }
166
```

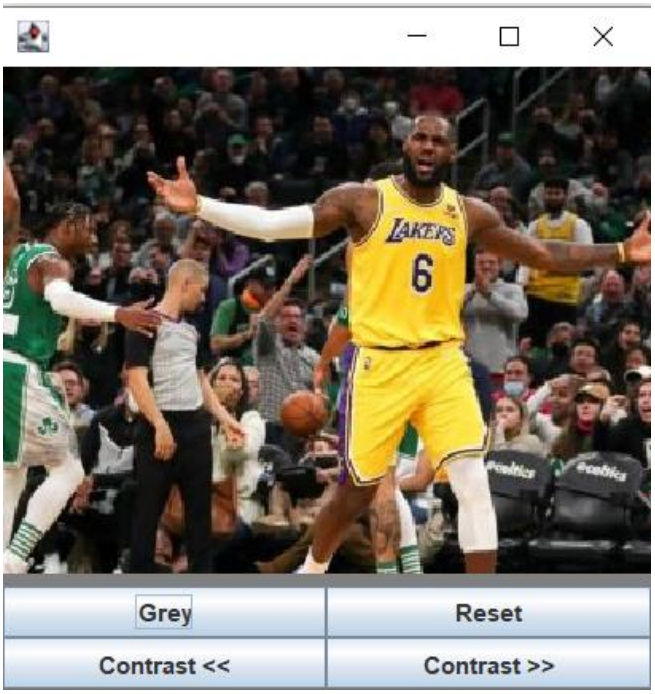
Pentru modificarea contrastului s-au folosit changeScaleFactor si rescale.

```
179    public void grayOut() {
180        ColorConvertOp colorConvert = new ColorConvertOp(ColorSpace
181            .getInstance(ColorSpace.CS_GRAY), null);
182        colorConvert.filter(bi, bi);
183      }
184
```

Pentru a face imaginea alb-negru s-a folosit grayOut.

4.Cerinte tema

1. Imaginea sursa este BMP (fisier) – 24bit BMP:



```
110         JFileChooser fileopen = new JFileChooser();
111         FileNameExtensionFilter filter = new FileNameExtensionFilter("24bit bmp images", "bmp");
112         fileopen.setFileFilter(filter);
113         int ret = fileopen.showDialog(null, "Open file");
114
115         if (ret == JFileChooser.APPROVE_OPTION)
116         {
117           File selectedFile = fileopen.getSelectedFile();
118           path = selectedFile.getAbsolutePath();
119         }
```

2. S-au folosit doar secvente de cod low level pentru a modifica contrastul

```
150     public void changeScaleFactor() {
151         if (contrastInc) {
152             if (scaleFactor < 2)
153                 scaleFactor = scaleFactor+0.1f;
154         }
155         else {
156             if (scaleFactor > 0)
157                 scaleFactor = scaleFactor-0.1f;
158         }
159     }
160
161     public void rescale() {
162         rescale = new RescaleOp(scaleFactor, offset, null);
163         rescale.filter(biSrc, biDest);
164         bi = biDest;
165     }
166
```

si ajustarea alb-negru.

```
179     public void grayOut() {
180         ColorConvertOp colorConvert = new ColorConvertOp(ColorSpace
181             .getInstance(ColorSpace.CS_GRAY), null);
182         colorConvert.filter(bi, bi);
183       }
184
```

3. Include incapsulare: imaginea se obtine prin intermediul unei metode, nu direct prin intermediul constructorului, pentru a evita introducerea de date gresite.
   Mostenire: exista clase mostenite (punctual 7).
   Polimorfism:

```
public void loadImage() {

public void loadImage(String path)
```

```
107    public void loadImage() {
108
109
110        JFileChooser fileopen = new JFileChooser();
111        FileNameExtensionFilter filter = new FileNameExtensionFilter("bmp images", "bmp");
112        fileopen.setFileFilter(filter);
113        int ret = fileopen.showDialog(null, "Open file");
114
115        if (ret == JFileChooser.APPROVE_OPTION)
116        {
117          File selectedFile = fileopen.getSelectedFile();
118          path = selectedFile.getAbsolutePath();
119        }
120
121        displayImage = Toolkit.getDefaultToolkit().getImage(path);
122        MediaTracker mt = new MediaTracker(this);
123        mt.addImage(displayImage, 1);
124        try {
125            mt.waitForAll();
126        } catch (Exception e) {
127            System.out.println("Exception while loading.");
128        }
129
130        if (displayImage.getWidth(this) == -1) {
131            System.out.println("No bmp file");
132            System.exit(0);
133        }
134    }
```

4. Codul sursa este comentat cu explicatii.
5. Operatii de lucru cu fisiere.

```
JFileChooser fileopen = new JFileChooser();
int ret = fileopen.showDialog(null, "Open file");

if (ret == JFileChooser.APPROVE_OPTION)
{
  File selectedFile = fileopen.getSelectedFile();
  path = selectedFile.getAbsolutePath();
}
```

6. Utilizatorul acceseaza fisierul dorit prin intermediul perifericelor (mouse, tastatura) si modifica imaginea apasand pe butoanele realizate pentru functiile respective.

```
58    class ButtonListener implements ActionListener {
59        public void actionPerformed(ActionEvent e) {
60            JButton temp = (JButton) e.getSource();
61
62            if (temp.equals(grayButton)) {
63                displayPanel.color = false;
64                displayPanel.grayOut();
65                displayPanel.repaint();
66              }
67            else if (temp.equals(resetButton)) {
68                displayPanel.contrastInc = true;
69                displayPanel.reset();
70                displayPanel.repaint();
71              }
72
73
74            else if (temp.equals(contIncButton)) {
75                displayPanel.contrastInc = true;
76                displayPanel.changeScaleFactor();
77                displayPanel.rescale();
78                displayPanel.repaint();
79            }
80            else if (temp.equals(contDecButton)) {
81                displayPanel.contrastInc = false;
82                displayPanel.changeScaleFactor();
83                displayPanel.rescale();
84                displayPanel.repaint();
85            }
86        }
87    }
```

7. Exista clase mostenite.

```
10 public class Modify extends JFrame {
48    class WindowEventHandler extends WindowAdapter {
89 class DisplayPanel extends JPanel {
```

8. Include varargs.

```
public abstract class abs {
    private static boolean varArgs(String ... arguments){
        long start = System.currentTimeMillis();
        if(arguments.length == 3) {
            String parameter1 = arguments[0].toLowerCase();
            if(!parameter1.equals("and")&&!parameter1.equals("or")&&!parameter1.equals("xor"))
                return false;
        }
        return false;
    }
}
```

9. Include interface.

```
public class Afis implements interf {
    public void intAf()
    {
        System.out.println("Succes!");
    }

}
```

10. Include clase si metode abstracte

```
public interface interf {
public void intAf();
}
```

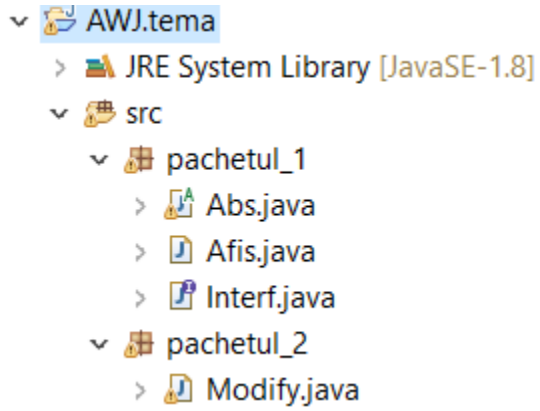
```
    class ButtonListener implements ActionListener {
```

11. Include tratarea exceptiilor.

```
        try {
            mt.waitForAll();
        } catch (Exception e) {
            System.out.println("Exception while loading.");
        }

        if (displayImage.getWidth(this) == -1) {
            System.out.println("No bmp file");
            System.exit(0);
        }
    }
```

12.Aplicatia contine 2 pachete.



5.Conlcuzii

Contrastul este diferența dintre înnegrirea maximă și cea minimă care se pot vedea într-o imagine fotografică. În percepția vizuală a lumii reale, contrastul este determinat de diferența dintre culoarea și luminozitatea unui obiect și alte obiecte din interiorul aceluiași câmp vizual.

Ajustare Alb-Negru inseamna a transforma o imagine color intr-o imagine alb-negru.

6.Bibliografie

- Java Courses – prof. A. Hossu
- Google drive
- https://stackoverflow.com/questions/14513542/how-to-convert-image-to-black-and-white-using-java
- https://docs.oracle.com/javase/7/docs/api/java/awt/image/BufferedImage.html

- https://memorynotfound.com/convert-image-grayscale-java/#:~:text=We%20use%20the%20ImageIO.,and%20color%20of%20each%20pixel.

- https://www.geeksforgeeks.org/variable-arguments-varargs-in-java/#:~:text=Variable%20Arguments%20(Varargs)%20in%20Java%20is%20a%20method%20that%20takes,a%20variable%20number%20of%20arguments.