Introdução ao Processamento de Imagens Digitais em Java com Aplicações em Ciências Espaciais

Escola de Verão do Laboratório Associado de Computação e Matemática Aplicada

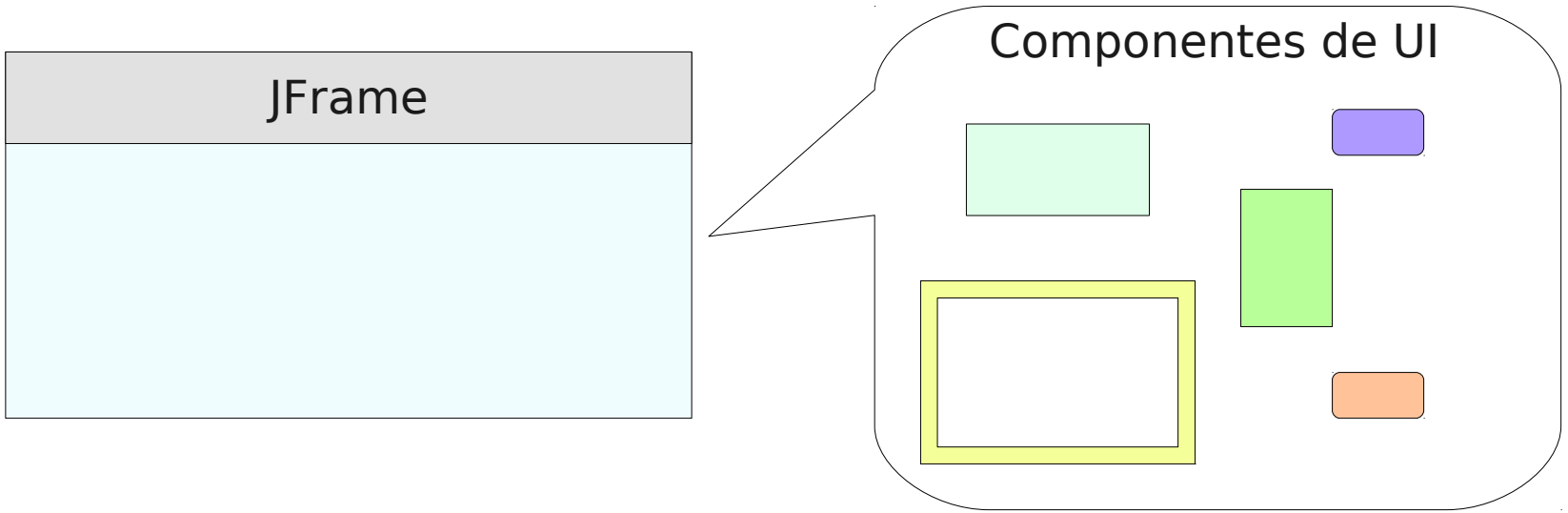
Rafael Santos

Programa

- ***Dia 1:*** Introdução e aplicações. Representação de imagens digitais. Criando e armazenando imagens.
- ***Dia 2:*** Visualização de imagens.
- ***Dia 3:*** Manipulação de pixels e regiões. Operadores da API JAI.
- ***Dia 4:*** Outros operadores da API JAI. Implementação de algoritmos.

Visualização de Imagens

- Componentes de interfaces gráficas para mostrar imagens.
- Geralmente bem simples, melhorias como interatividade, processamento, etc. ficam por conta do programador...
  - … o que é fácil de fazer graças ao mecanismo de herança!
- Conhecimentos de programação de interfaces gráficas em Java são úteis: só conhecimento de *design* não adiantam.



Display de Imagens (sem JAI)

```
public static void main(String[] args) throws IOException
  {
  BufferedImage image = ImageIO.read(new File(args[0]));
  JFrame frame = new JFrame("Display Image: "+args[0]);
  ImageIcon icon = new ImageIcon(image);
  JLabel imageLabel = new JLabel(icon);
  frame.getContentPane().add(new JScrollPane(imageLabel));
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.setSize(600,300);
  frame.setVisible(true);
  }
```

- BufferedImage → ImageIcon → JLabel (JScrollPane).

Display de Imagens (sem JAI)

Display de Imagens (com JAI)

```
public static void main(String[] args) throws IOException
  {
  BufferedImage image = ImageIO.read(new File(args[0]));
  JFrame frame = new JFrame("Display Image: "+args[0]);
  DisplayJAI display = new DisplayJAI(image);
  frame.getContentPane().add(new JScrollPane(display));
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.setSize(600,300);
  frame.setVisible(true);
  }
```

- `DisplayJAI` é mais flexível, permite alguma interação (não implementada).
- Não é parte da API JAI (!?).

Imagens Sincronizadas

- Componente que mostra duas imagens sincronizadas:
  - Modificação no *viewport* de uma causa modificação no *viewport* da outra.
- Composição de instâncias de `DisplayJAI`.

Imagens Sincronizadas

- Exibição de duas instâncias de `DisplayJAI` de forma sincronizada.

```
public class DisplayTwoSynchronizedImages extends JPanel
        implements AdjustmentListener
  {
  protected DisplayJAI dj1;
  protected DisplayJAI dj2;
  protected JScrollPane jsp1;
  protected JScrollPane jsp2;
```

Imagens Sincronizadas

```
public DisplayTwoSynchronizedImages(RenderedImage im1,
                                    RenderedImage im2)
  {
  super();
  // Cria componente com duas imagens com JScrollPanes
  setLayout(new GridLayout(1,2));
  dj1 = new DisplayJAI(im1);
  dj2 = new DisplayJAI(im2);
  jsp1 = new JScrollPane(dj1);
  jsp2 = new JScrollPane(dj2);
  add(jsp1);
  add(jsp2);
  // Registra listeners para os scroll bars do JScrollPanes
  jsp1.getHorizontalScrollBar().addAdjustmentListener(this);
  jsp1.getVerticalScrollBar().addAdjustmentListener(this);
  jsp2.getHorizontalScrollBar().addAdjustmentListener(this);
  jsp2.getVerticalScrollBar().addAdjustmentListener(this);
  }
```

Imagens Sincronizadas

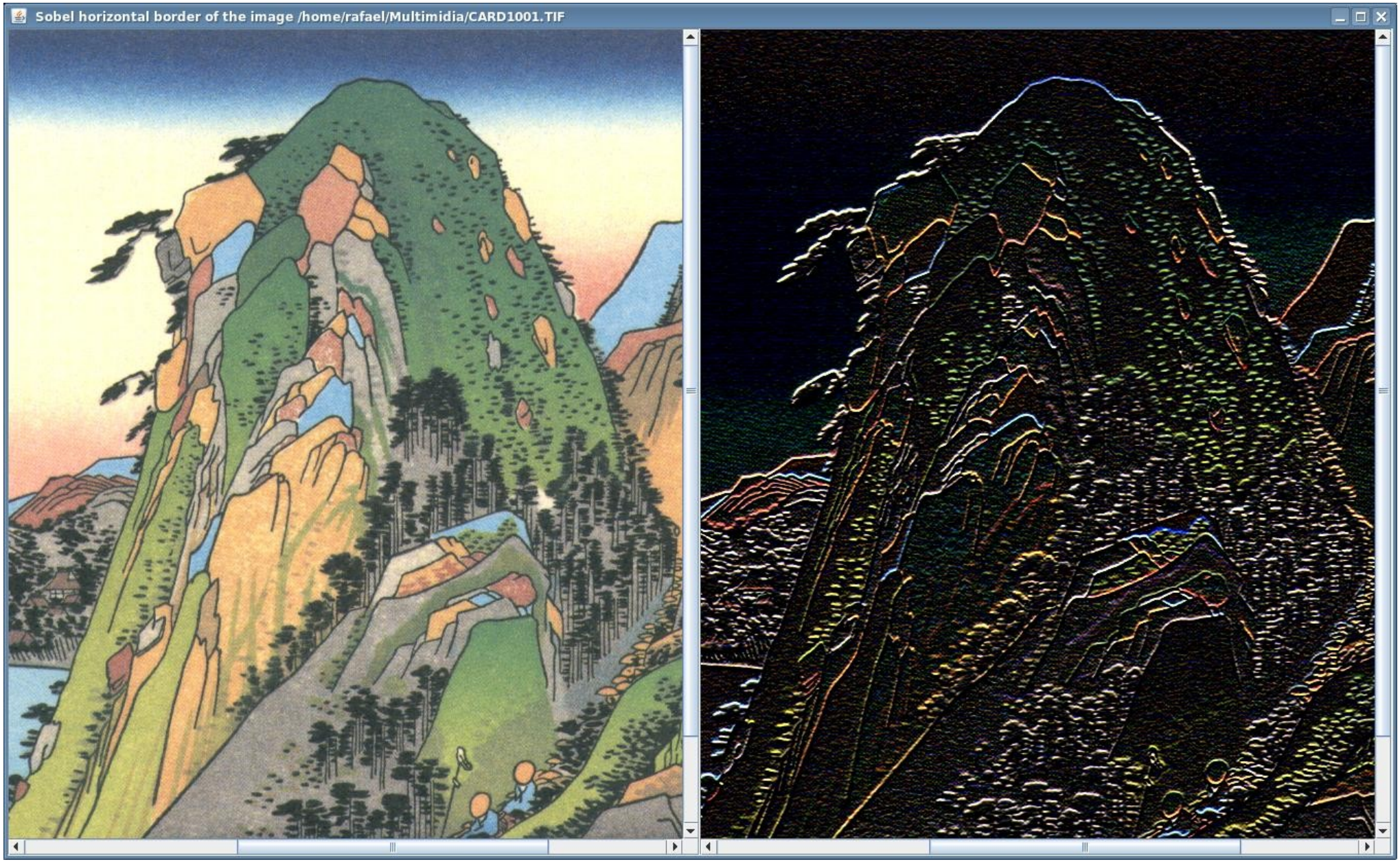
```
public void adjustmentValueChanged(AdjustmentEvent e)
  {
  if (e.getSource() == jsp1.getHorizontalScrollBar())
    jsp2.getHorizontalScrollBar().setValue(e.getValue());
  if (e.getSource() == jsp1.getVerticalScrollBar())
    jsp2.getVerticalScrollBar().setValue(e.getValue());
  if (e.getSource() == jsp2.getHorizontalScrollBar())
    jsp1.getHorizontalScrollBar().setValue(e.getValue());
  if (e.getSource() == jsp2.getVerticalScrollBar())
    jsp1.getVerticalScrollBar().setValue(e.getValue());
  }

}
```

Imagens Sincronizadas: Exemplo

```
public class Borda
  {
  public static void main(String[] args)
    {
    PlanarImage imagem = JAI.create("fileload", args[0]);
    float[] kernelMatrix = { -1, -2, -1,
                              0,  0,  0,
                              1,  2,  1 };
    KernelJAI kernel = new KernelJAI(3,3,kernelMatrix);
    PlanarImage bordas = JAI.create("convolve",imagem,kernel);
    JFrame frame = new JFrame("Bordas horizontais");
    frame.add(new DisplayTwoSynchronizedImages(imagem,bordas));
    frame.pack();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
    }
  }
```

Imagens Sincronizadas: Exemplo



Imagens Substitutas

- Uso de imagens substitutas (*surrogate images*):
- Criamos uma imagem normalizada com pixels entre valores 0-255
- Transformamos o tipo da imagem para bytes.
- Uma classe que herda de `DisplayJAI` pode executar estes passos.

Imagens Substitutas

```
public class DisplaySurrogateImage extends DisplayJAI
  {
  protected PlanarImage surrogateImage;
  protected int width,height;
  public DisplaySurrogateImage(PlanarImage image)
    {
    width = image.getWidth();
    height = image.getHeight();
    // Recuperamos valores extremos da imagem.
    ParameterBlock pbMaxMin = new ParameterBlock();
    pbMaxMin.addSource(image);
    PlanarImage extrema = JAI.create("extrema", pbMaxMin);
    double[] allMins = (double[])extrema.getProperty("minimum");
    double[] allMaxs = (double[])extrema.getProperty("maximum");
    double minValue = allMins[0];
    double maxValue = allMaxs[0];
    for(int v=1;v<allMins.length;v++)
      {
      if (allMins[v] < minValue) minValue = allMins[v];
      if (allMaxs[v] > maxValue) maxValue = allMaxs[v];
      }
```

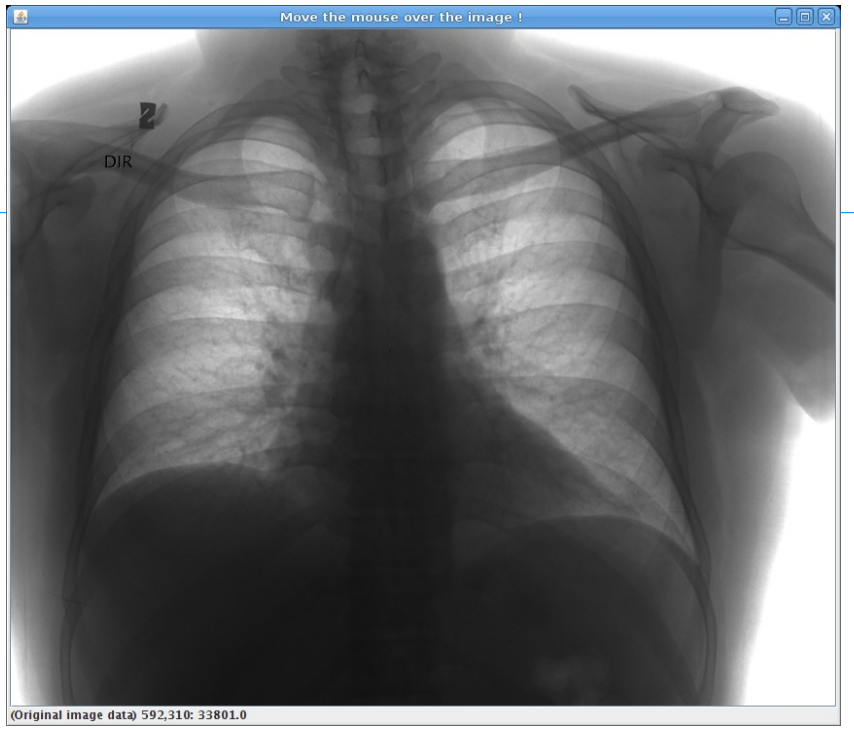
Imagens Substitutas

```
    // Reescalamos os níveis de cinza da imagem.
    double[] subtract = new double[1];   subtract[0]   = minValue;
    double[] multiplyBy = new double[1];
    multiplyBy[0] = 255./(maxValue-minValue);
    ParameterBlock pbSub = new ParameterBlock();
    pbSub.addSource(image);
    pbSub.add(subtract);
    surrogateImage = (PlanarImage)JAI.create("subtractconst",pbSub);
    ParameterBlock pbMult = new ParameterBlock();
    pbMult.addSource(surrogateImage);
    pbMult.add(multiplyBy);
    surrogateImage = (PlanarImage)JAI.create("multiplyconst",pbMult);
    // Convertemos para bytes.
    ParameterBlock pbConvert = new ParameterBlock();
    pbConvert.addSource(surrogateImage);
    pbConvert.add(DataBuffer.TYPE_BYTE);
    surrogateImage = JAI.create("format", pbConvert);
    // Usamos esta imagem para display.
    set(surrogateImage);
    }

}
```

Imagens Substitutas

```
public class DemonstraDisplaySurrogateImage
  {
  public static void main(String[] args)
    {
    PlanarImage image = JAI.create("fileload", args[0]);
    JFrame frame = new JFrame("Mostrando "+args[0]);
    frame.getContentPane().add(
       new JScrollPane(new DisplaySurrogateImage(image)));
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.pack();
    frame.setVisible(true);
    }
  }
```

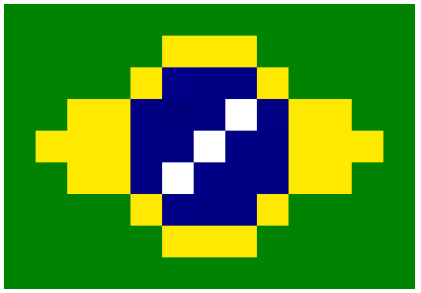


Imagens Substitutas: LUTs

- Uso de imagens substitutas (*surrogate images*) com LUTs:
- *Look-up Tables* (*LUTs*): tabela de cores.

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	1	2	2	2	1	0	0	0	0
0	0	1	1	2	2	2	3	2	1	1	0	0
0	1	1	1	2	2	3	2	2	1	1	1	0
0	0	1	1	2	3	2	2	2	1	1	0	0
0	0	0	0	1	2	2	2	1	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Índice	R	G	B
0 →	0	131	1
1 →	255	233	0
2 →	0	0	124
3 →	255	255	255



Imagens Substitutas: LUTs

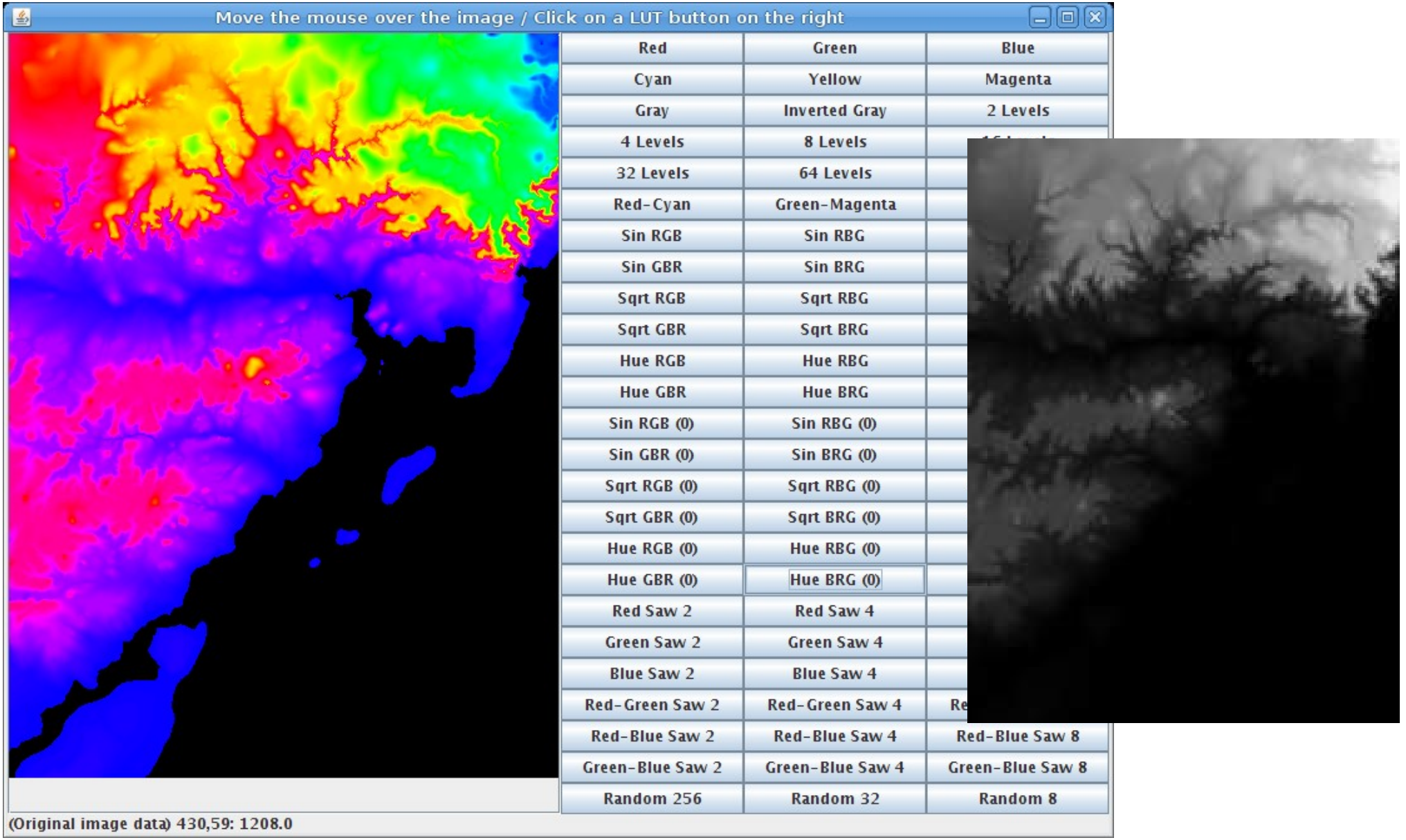
```
public void setLUT(short[][] lut)
  {
  SampleModel sampleModel = surrogateImage.getSampleModel();
  SampleModel newSampleModel =
     RasterFactory.createBandedSampleModel(DataBuffer.TYPE_BYTE,
                          sampleModel.getWidth(),sampleModel.getHeight(),3);
  byte[] reds = new byte[256];
  byte[] greens = new byte[256];
  byte[] blues = new byte[256];
  for(int i=0;i<256;i++)
    {
    reds[i] = (byte)lut[i][0];
    greens[i] = (byte)lut[i][1];
    blues[i] = (byte)lut[i][2];
    }
  ColorModel colorModel = new IndexColorModel(8,256,reds,greens,blues);
  ImageLayout layout = new ImageLayout(surrogateImage);
  layout.setColorModel(colorModel);
  HashMap<RenderingHints.Key,ImageLayout> map =
    new HashMap<RenderingHints.Key,ImageLayout>();
  map.put(JAI.KEY_IMAGE_LAYOUT,layout);
  RenderingHints hints = new RenderingHints(map);
  ParameterBlock pb = new ParameterBlock();
  pb.addSource(surrogateImage);
  PlanarImage newSurrogateImage = JAI.create("format",pb,hints);
  set(newSurrogateImage);
  }
```

Imagens Substitutas: LUTs

```
/** The inverted gray lut */
public final static short[][] invGray()
  {
  short[][] lut = new short[256][3];
  for(short i=0;i<256;i++)
    {
    lut[i][0] = (short)(255-i);
    lut[i][1] = (short)(255-i);
    lut[i][2] = (short)(255-i);
    }
  return lut;
  }
```

```
/** The sin lut (rgb order) */
public final static short[][] sin_rgb()
  {
  short[][] lut = new short[256][3];
  for(short i=0;i<256;i++)
    {
    lut[i][0] = (short)(127*(1+Math.sin(Math.PI*(i-127)/255)));
    lut[i][1] = (short)(127*(1+Math.sin(Math.PI*(i    )/255)));
    lut[i][2] = (short)(127*(1+Math.sin(Math.PI*(i+127)/255)));
    }
  return lut;
  }
```

Imagens Substitutas: LUTs



Desenhando em Imagens

- Podemos obter contextos gráficos de `BufferedImages` e `PlanarImages`..
  - .. e usá-los para desenhar sobre a imagem.
- As imagens são modificadas (na memória) e podem ser visualizadas e/ou armazenadas com os gráficos.

Desenhando em Imagens

```
BufferedImage baseImage = ImageIO.read(new File("sjc_region.png"));
int[][] coords = new int[][] {
    {714,219},
    {822,256},
    {797,329},
    {710,300},
    {711,293},
    {666,271}};
Path2D.Float regionOfInterest = new Path2D.Float();
boolean isFirst = true;
double firstX=0,firstY=0;
for(int[] coord:coords)
  {
  int x = coord[0]; int y = coord[1];
  if (isFirst)
    {
    regionOfInterest.moveTo(x,y);
    firstX = x;
    firstY = y;
    isFirst = false;
    }
  else { regionOfInterest.lineTo(x,y); }
  }
regionOfInterest.lineTo(firstX,firstY);
```

Desenhando em Imagens

```
Path2D.Float pathForWholeImage = new Path2D.Float();
pathForWholeImage.moveTo(0,0);
pathForWholeImage.lineTo(baseImage.getWidth(),0);
pathForWholeImage.lineTo(baseImage.getWidth(),baseImage.getHeight());
pathForWholeImage.lineTo(0,baseImage.getHeight());
pathForWholeImage.lineTo(0,0);
Area wholeImage = new Area(pathForWholeImage);
wholeImage.subtract(new Area(regionOfInterest));
```

```
Graphics2D g2d = (Graphics2D)baseImage.getGraphics();
g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                     RenderingHints.VALUE_ANTIALIAS_ON);
g2d.setColor(new Color(255,255,255,100));
g2d.fill(wholeImage);
g2d.setStroke(new BasicStroke(5f));
g2d.setColor(new Color(255,0,0,200));
g2d.draw(regionOfInterest);
```

```
JFrame frame = new JFrame("Highlighting image regions");
ImageIcon icon = new ImageIcon(baseImage);
JLabel label = new JLabel(icon);
frame.getContentPane().add(new JScrollPane(label));
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.pack();     frame.setVisible(true);
```

Desenhando em Imagens

Programa

- ***Dia 1:*** Introdução e aplicações. Representação de imagens digitais. Criando e armazenando imagens.
- ***Dia 2:*** Visualização de imagens.
- ***Dia 3:*** Manipulação de pixels e regiões. Operadores da API JAI.
- ***Dia 4:*** Outros operadores da API JAI. Implementação de algoritmos.

Mais informações em...

- http://www.lac.inpe.br/~rafael.santos
  - http://www.lac.inpe.br/~rafael.santos/piapresentacoes.jsp
  - http://www.lac.inpe.br/JIPCookbook/index.jsp
- http://www.lac.inpe.br/ELAC/index.jsp