CS 259
Lab 5: Image Histograms and Transformations

Instructor:
Joel Castellanos
e-mail: joel@unm.edu
Web: http://cs.unm.edu/~joel/
Office: Farris Engineering Center (FEC) room 319

Image Histogram
Making Image Histograms

1. **Open an Image** selected by the user.
2. **Calculate** Red, Green and Blue histogram values of the image.
3. **Normalize** pixel counts.
4. **Plot** Histograms in corresponding colors.

Histogram Application: Normalize Contrast

Links on Class Website:
*Original:*
  Treebeard-John_Howe-500x340.png

*After Normalizing Contrast:*
  Treebeard-John_Howe-500x340-autoContrast.png
Summary

1. Open an Image into a Picture: (source code on the class website).
2. Auto Contrast: Normalize the image and display in a new picture window.
3. Calculate Histograms: Examine each pixel of each image and count the Red, Green and Blue values separately for each images and store them in arrays.
4. Normalize Histograms:
   - Larger images will have larger pixel counts for each brightness level.
   - Uniformly scale pixel counts so that largest count in each set of plots of one image is 100 pixels.
5. Plot Histograms in corresponding colors.
   - Create six empty Picture(256, 100) objects.
   - Draw vertical lines (histogram bars) of different lengths from the base of the plot up to the histogram value. Draw the bars in the color of the histogram channel.
6. Custom Transformation: Apply a transformation of your own choice and design to the original image and display in a new window.
Using Picture to Display an Image

// Display a file chooser dialog for the user to select an
// image file. Then paints the image in a Picture.
Picture myPic = new Picture();

System.out.println("Outside Width = " + myPic.getWidth());
System.out.println("Inside Width = " + myPic.getImageWidth());

Output: Outside Width = 418
Inside Width = 400

Using Picture to Create an Empty Frame

The Picture class is a utility class that is designed to be
instantiated by other classes. Therefore, it does not need to
have main.
The next few slides show how to use the Picture class
from other classes.

//Create and display an empty Picture JFrame that is
// 1000×500 pixels.
Picture p1 = new Picture(1000, 500);

//Picture extends JFrame. Therefore, all methods of
// JFrame (i.e. setLocation) are also methods of Picture.
p1.setLocation(100, 100);
Drawing in a Picture

1) `Picture myPic = new Picture(300,100);`
2) `myPic.setTitle("Euclid");`
3) `Graphics canvas = myPic.getOffScreenGraphics();`
4) `// Fill the off-screen buffer with light gray.`
5) `canvas.setColor(Color.LIGHT_GRAY);`
6) `canvas.fillRect(0,0,300,100);`
7) `// Draw a line in a color different from the background.`
8) `canvas.setColor(Color.BLUE);`
9) `canvas.drawLine(0,99,299,0);`
10) `// Copy the off-screen buffer to the JFrame`
11) `myPic.repaint();`

Calculate Histograms

- Create six arrays:
  ```java
  redOrg[], greenOrg[], blueOrg[],
  redMod[], greenMod[], blueMod[]
  ```
  all `int[256]` (or, if you want to get fancy, create a single 2D array: `hist[6][256]`)  
- Read the red, blue and green values at each pixel of each row and each column of the input image.
- If the red value is 7 then increment red[7]. if the red value is 127 then increment red[127]....
Normalization

- **Audio normalization** is the process of uniformly increasing (or decreasing) the amplitude of an entire audio signal so that the resulting peak amplitude matches a desired target.

- **Image normalization**: Subtract the lowest red, green and blue brightness value from all red, green and blue values. Then, multiply all red green and blue values by a scaling factor equal to 255 divided by the new maximum red, green and blue value. Note: each of the color channels must be shifted and scaled by the same values.

- **Normalization in mathematics** is a process that maps a set of real numbers, $a_i \in A$, to $b_i \in B$ such that $b_i \in [0,1]$ and for any $a_i$ and $a_k \in A$ that maps to $b_i$ and $b_k \in B$, $a_i/a_k = b_i/b_k$

Method: getMax

```
public class Project3 {
    public static void main(String[] args) {
        int[] numList = {5, 10, 20, 30, 10, 15};
        int max = getMax(numList);
        System.out.println(max);
    }

    private static int getMax(int[] array) {
        int max = array[0];
        for (int i=1; i<array.length; i++) {
            if (array[i] > max) max = array[i];
        }
        return max;
    }
}
```

Before a list of numbers can be normalized, the list must be searched for its minimum and maximum values.

- Why start with $i=1$?
- What is going on here?
A Java method can only return one value. However, that one value can be a reference to a class of data fields.

```java
public class DataMinMax {
    public int min, max;
}
```

Example of when it is okay to use public fields.

```java
public class Example {
    public static void main(String[] args) {
        // code on next slide
    }
    private static DataMinMax getExtremes(int[] array) {
        // code on next slide
    }
}
```

```java
public static void main(String[] args) {
    int[] numList = {10, 10, 20, 30, 5, 15};
    DataMinMax a = getExtremes(numList);
    System.out.println(a.min +", " + a.max);
}
```

```java
private static DataMinMax getExtremes(int[] array) {
    DataMinMax data = new DataMinMax();
    data.min = array[0];
    data.max = array[0];
    for (int i=1; i<array.length; i++) {
        if (array[i] > data.max) data.max = array[i];
        if (array[i] < data.min) data.min = array[i];
    }
    return data;
}
```
Formatting Numeric Output

```java
public static void main(String[] args) {
    double x = 2.0/3.0;
    System.out.format("$%5.2f,%9d", x, (int)x);
    // $0.67, 0
}
```

- `$""` Literal.
- "%5.2f" Format a float (or double) with 2 decimal places. If the formatted string (including decimal point) is less than 2 characters, then pad with spaces and right justify.
- "", `Literal.`
- "%9d" Format an integer. If the formatted string is less than 9 characters, then pad with spaces and right justify.

Shift and Scale to [0, 30]

```java
int[] x = {2, 5, -5, 15, -4, 7, -3};
int minX = -5, maxX = 15;
int[] y = new int[x.length];
int[] z = new int[x.length];
for (int i=0; i<x.length; i++)
    { y[i] = ((x[i]-minX)*30) / (maxX-minX); z[i] = (x[i]-minX)*(30) / (maxX-minX); System.out.format( "%2d --&gt; %2d,%3d\n", x[i], y[i], z[i]);
    }
```

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>-5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>-4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>-3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
Shift and Scale to [0.0, 1.0]

1) int[] x = {2, 5, -5, 15, -4, 7, -3};
2) int minX = -5;
3) int maxX = 15;
4) double[] y = new double[x.length];
5) double[] z = new double[x.length];
6) for (int i=0; i<x.length; i++)
7) {
8)   y[i] = (x[i]-minX) / (maxX-minX);
9)   z[i] = (x[i]-minX) / (double)(maxX-minX);
10)  System.out.format("%2d --> %5.2f,%6.2f\n",  
11)   x[i], y[i], z[i]);
12) }

Normalize: Program Structure

public class ExampleOfNormalize
{
  private static void printArray(int[] array)
  {
    private static DataMinMax getExtremes(int[] array)
    {
      //Assumes the target minimum is zero
      private static void normalize(int[] array, int targetMax)
      {
      
      public static void main(String[] args)
      {
      
      }
Method: PrintArray (formatted output)

```java
private static void printArray(int[] array)
{
    System.out.print("[");
    for (int i=0; i<array.length; i++)
    {
        if (i < array.length-1)
            System.out.format("%2d, ", array[i]);
        else
            System.out.format("%2d\n", array[i]);
    }
    System.out.format("%2d]: If integer argument is less than 2 characters, then pad with spaces and right justify. Then print the literal "}\n".
}
```

Examples: [10, 15, 99, 30, 5, 101] [0, 5, 1, 33, 17, 10]

Methods: normalize and main

```java
private static void normalize(int[] array, int targetMax)
{
    DataMinMax d = getExtremes(array);
    int range = d.max - d.min;
    for (int i=0; i<array.length; i++)
    { array[i] = ((array[i]-d.min)*targetMax)/range;
    }
    [10, 15, 20, 30, 5, 10] [1, 2, 3, 6, 0, 1]
```

```java
public static void main(String[] args)
{
    int[] numList = {10, 15, 20, 30, 5, 10};
    printArray(numList);
    normalize(numList, 6);
    printArray(numList);
    Largest normalized to targetMax.
    Smallest normalized to 0.
}
```
Normalize: What Happens in Memory

In Java, arrays and objects are passed by reference. Primitive types are passed by value.

```java
private static void normalize(int[] array, int targetMax) {
    DataMinMax d = getExtremes(array);
    int range = d.max - d.min;
    for (int i=0; i<array.length; i++)
    { array[i] = ((array[i]-d.min)*targetMax)/range;
    }
}
```

When `normalize` returns, the pointer `array` is garbage collected. However since, the data pointed to by `array`, is the same as the data pointed to by `numList` changes to that data persist.  

<table>
<thead>
<tr>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

Quiz: Shift and Scale

1) `int[] x = {15, 25, 5, 17, 22, 9, 12};`
2) `int min = 5;`
3) `int max = 25;`
4) `int max2 = 100; //min2 will be 0`
5) `for (int i=0; i<x.length; i++)`
6) `{`
7) `int y = ((x[i]-min)*max2) / (max-min);`
8) `System.out.print(y + ", ");`
9) `}

The first three values to be displayed in the console are:

- a) 50, 100, 0,
- b) 75, 125, 25,
- c) 75, 125, 50,
- d) 40, 80, 10,
- e) 40, 100, 10,
Histogram Test: A SIMPLE Case

- 10x12 pixels.
- First 4 rows have RGB color: (100, 100, 200).
- Bottom 8 rows have the RGB color: (250, 150, 175).

Print each histogram array for this image.

- `redOrg[100]` should = 40
- `redOrg[250]` should = 80
- all other elements of `redOrg[]` should = 0

Getting it to Work (1 of 2)

Make SIMPLE test cases:

1. Use a very small image: i.e. 10 pixels x 12 pixels.
2. Use an image format that is **lossless** (NOT .jpg which will add slight color variations to a simple image who's original consisted of a few solid color regions).
3. Print and check every pixel value.
4. Print and check each channel's max.
5. Print and check ALL normalized values.
6. Show and check the histogram.
Getting it to Work (2 of 2)

- After you have calculated the correct histogram data, you need to draw each histogram.
- BEFORE trying to code the drawing of the histogram:
  1) Print the first 4 non-zero histogram values.
  2) Draw these first 4 lines with paper and pencil.
  3) Write the coordinates of the endpoints of the 4 lines.
  4) Look for a pattern in what you have written.
  5) If you cannot see a pattern, then draw another line on paper and write another set of endpoints.
  6) Only after you see the pattern do you have a chance of being able to write the code to continue that pattern.

Grading Rubric 30 Points Total (1 of 2)

[5 Points]: Create a class called Histogram that creates eight instances of Picture:
  - One for the user-selected image,
  - One for the auto-contrast image,
  - Three for the histograms of the user-selected image,
  - Three for the histograms of the auto-contrast image.

[6 Points]: The auto-contrast image must be the same size as the user-selected image, it must shift and scale all three color channels by the same amount, it must be the original image with color channel range from 0 to 255.

[5 Points]: Display custom transformation window showing an interesting effect.
**Grading Rubric  (2 of 2)**

[2 Points]: The histogram bars must be the color of the color channel it represents.

[2 Points]: The histogram windows must all have an inside size of 256×100 pixels.

[3 Points]: The three histograms of the user-selected image must all use the same scale with the highest bar of the three channels being 100 pixels.

[3 Points]: The three histograms of the auto-contrast image must all use the same scale with the highest bar of the three channels being 100 pixels.

[1 Points]: Each histogram must have an identifying title.

[3 Points]: The 8 Picture windows must be placed on the display so that they are not all covering each other with a layout that is aware of the screen size.

---

**Quiz: Shift and Scale**

1) `int[] x = {7, 12, 4, 17, 22, 9, 2};`  
2) `int min = 2;`  
3) `int max = 22;`  
4) `int max2 = 100; //min2 will be 0`  
5) `for (int i=0; i<x.length; i++)`  
6) `{`  
7) `int y = ((x[i]-min)*max2) / (max-min);`  
8) `System.out.print(y + ", ");`  
9) `}

The first three values to be displayed in the console are:

a) 35, 70, 25,  
b) 35, 60, 20,  
c) 35, 60, 22,  
d) 25, 60, 22,  
e) 25, 50, 10,
Quiz: Histogram

1) `int[] x = {0, 2, 3, 0, 1, 0, 2};`
2) `int[] histogram = new int[4];`
3) `for (int i=0; i<x.length; i++)`
4) `{`
5) `int y = x[i];`
6) `histogram[y]++;`
7) `}`
8) `for (int i=0; i<histogram.length; i++)`
9) `{ System.out.print(histogram[i]+" ");
10) }

The values displayed in the console are:

a) 3 1 2 1  
b) 0 2 3 0  
c) 1 3 4 1  
d) 3 4 4 3  
e) 3 2 1 3