Image Processing

Review

- Images – an array of colors
- Color – RGBA
- Loading, modifying, updating pixels
- pixels[] as a 2D array
- Simple filters – tinting, grayscale, negative, sepia

void setup() {
  // Load the image three times
  Image morhol = loadImage("andy-warhol2.jpg");
  size(morhol.width, morhol.height);

  // Draw modified images
  tint(255, 0, 0);
  image(morhol, 0, 0);
  // Draw modified images
  tint(0, 255, 0);
  image(morhol, 250, 0);
  tint(0, 0, 255);
  image(morhol, 500, 0);
}

Basic Filters

- Color
  - Extracting Red/Green/Blue colors
    - pixels[i] = color(red(c), 0, 0);
    - pixels[i] = color(0, 0, blue(c));
- Grayscale
  - pixels[i] = color(0.3*red(c)+ 0.59*green(c)+ 0.11*blue(c));
- Negative
  - pixels[i] = color(255-red(c), 255-green(c), 255-blue(c));
- Sepia (Technique for archiving BW photos)
  - float r = red(c)*0.393+green(c)*0.769+blue(c)*0.189;
  - float g = red(c)*0.349+green(c)*0.686+blue(c)*0.168;
  - float b = red(c)*0.272+green(c)*0.534+blue(c)*0.131;
  - pixels[i] = color(r, g, b);

Examples

- blackWhite
- negative
- sepia
- sepiaPalette
- sepiaWithPalette

A 100-pixel wide image

- First pixel at index 0
- Right-most pixel in first row at index 99
- First pixel of second row at index 100

The pixels[] array is one-dimensional
Example

```java
void setup() {
  size(400, 400);
  // Load colors into the pixels array
  loadPixels();
  // Access pixels as a 2D array
  for (int y=0; y<height; y++) {
    for (int x=0; x<width; x++) {
      // Compute distance to center point
      float d = dist(x, y, width/2, height/2);
      int idx = width*y + x;
      // Set pixel as distance to center
      pixels[idx] = color(d);
    }
  }
  // Update the sketch with pixel data
  updatePixels();
}
```

What does this program do?

```
void setup() {
  size(400, 400);
  // Load colors into the pixels array
  loadPixels();
  // Access pixels as a 2D array
  for (int y=0; y<height; y++) {
    for (int x=0; x<width; x++) {
      // Compute distance to center point
      float d = dist(x, y, width/2, height/2);
      // Set pixel as distance to center
      pixels[idx] = color(d);
    }
  }
  // Update the sketch with pixel data
  updatePixels();
}
```

PImage

```java
PImage img = loadImage("myImage.jpg");
image(img, 0, 0);
```

Methods

- `loadPixels()`: Loads the color data out of the PImage object into a 1D array of colors named `pixels[]`.
- `updatePixels()`: Copies the color data from the `pixels[]` array back to the PImage object.
- `resize()`: Changes the size of the image.

get(...) (Cont'd)

- `get(...)`: Reads the color of any pixel or grabs a rectangle of pixels.
- `set(...)`: Writes a color to any pixel or writes an image into another.
- `copy(...)`: Copies pixels from one part of an image to another.
- `mask(...)`: Masks part of the image from displaying.
- `save(...)`: Saves the image to a TIFF, Targa, PNG, or JPEG file.
- `resize(...)`: Changes the size of an image to a new width and height.
- `blend(...)`: Copies a pixel or rectangle of pixels using different blending modes.
- `filter(...)`: Processes the image using one of several algorithms.

Example

```java
blackWhite2
```

+ Example

```java
whiteLine
```

Example

```java
blackWhite2
```

+ Example

```java
whiteLine
```
Example

- crumble
- reassemble

Example

- fade
- fade2

Simple Image Visualization

- Sample pixel colors every n pixels
- Draw a grid of basic shapes (ellipse, rect, line, triangle, etc) using the sampled color as fill color or stroke color

Pointillism

Medical Images
Image Processing in Manufacturing

What can you do with Image Processing?

Inspect, Measure, and Count using Photos and Video
http://www.youtube.com/watch?v=KsTtNWVhpgI

Image Processing Software
http://www.youtube.com/watch?v=1WJp9mGoW1M

Thresholding for Image Segmentation
- Pixels below a cutoff value are set to black
- Pixels above a cutoff value are set to white

Obamicon

Example

- obamicon

Image Enhancement
- Color and intensity adjustment
- Histogram equalization

Kun Huang, Ohio State - Digital Image Processing using Matlab, R.C.Gonzalez, R.E.Woods, and S.L.Eddins
Histogram Equalization

- Increases the global contrast of images
- So that intensities are better distributed
- Reveals more details in photos that are over or under exposed
- Better views of bone structure in X-rays

Convolution Filters (Area-based)

\[ E' = w_1A + w_2B + w_3C + w_4D + w_5E + w_6F + w_7G + w_8H + w_9I \]

Identity

- No change

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Random Neighbor

- Copies randomly from one of the 8 neighbors, and itself
**Example**

- randomNeighbor

**Average – smooth**

- Set pixel to the average of all colors in the neighborhood
- Smooths out areas of sharp changes.

\[
\begin{array}{ccc}
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
\end{array}
\]

**Sharpen – High Pass Filter**

- Enhances the difference between neighboring pixels
- The greater the difference, the more change in the current pixel

\[
\begin{array}{ccc}
-1 & -1 & -1 \\
-1 & 9 & -1 \\
-1 & -1 & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
0 & -2/3 & 0 \\
-2/3 & 11/3 & -2/3 \\
0 & -2/3 & 0 \\
\end{array}
\]

**Blur – Low Pass Filter**

- Softens significant color changes in image
- Creates intermediate colors

\[
\begin{array}{ccc}
1/16 & 2/16 & 1/16 \\
2/16 & 4/16 & 2/16 \\
1/16 & 2/16 & 1/16 \\
\end{array}
\]

**Example**

- convolution

**Dilation - Morphology**

- Set pixel to the maximum color value within a neighborhood around the pixel
- Causes objects to grow in size.
- Brightens and fills in small holes

\[
\begin{array}{ccc}
\end{array}
\]

\[
\begin{array}{ccc}
\end{array}
\]
**Erosion - Morphology**

- Set pixel to the minimum color value within a neighborhood around the pixel.
- Causes objects to shrink.
- Darkens and removes small objects.

**Image Enhancement**

- **Denoise**
  - Averaging
  - Median

**Blend Command**

- Linear interpolation of colours:
  \[ C = A \times \text{factor} + B \]
- Additive blending with white clip:
  \[ C = \min(A \times \text{factor} + B, 255) \]
- Subtractive blending with black clip:
  \[ C = \max(B - A \times \text{factor}, 0) \]
- Darkest only the darkest colour succeeds:
  \[ C = \min(A \times \text{factor}, B) \]
- Lightest only the lightest colour succeeds:
  \[ C = \max(A \times \text{factor}, B) \]
- Difference:
  Subtract colors from underlying image.
- Exclusion similar to difference, but less extreme.
- Multiply: Multiply the colors, result will always be darker.
Filter Command

```pseudocode
Filter Command
- image(b, 0, 0);  // Threshold
- filter(THRESHOLD, 0.5);

THRESHOLD converts the image to black and white pixels depending if they are above or below the threshold defined by the level parameter. The level must be between 0.0 (black) and 1.0 (white). If no level is specified, 0.5 is used.

GRAY converts any color in the image to grayscale equivalents

INVERT sets each pixel to its inverse value

POSTERIZE limits each channel of the image to the number of values specified as the level parameter

EROSION reduces the light areas with the amount defined by the level parameter

DILATE increases the light areas with the amount defined by the level parameter.
```

```pseudocode
// Threshold
PImage img;
void setup() {
  img = loadImage("myImage.jpg");
  size(img.width, img.height);
  image(img, 0, 0);
}
void draw() {}  

void drawImg(float thresh) {
  image(img, 0, 0);
  filter(THRESHOLD, thresh);
}
void mouseDragged() {
  float thresh = map(mouseY, 0, height, 0.0, 1.0);
  println(thresh);
  drawImg(thresh);
}
```

Image Processing Applications

Manual Colony Counter

http://www.youtube.com/watch?v=7B-9Wf6pENQ

---

Measuring Confluency in Cell Culture Biology

- Refers to the coverage of a dish or flask by the cells
- 100% confluency = completely covered

Image Processing Method

1. Mask off unimportant parts of image
2. Threshold image
3. Count pixels of certain color
Filter: Threshold

Subtracted  Threshold

Count pixels to quantitate: 5.3% confluency

Vision Guided Robotics Colony Picking

Predator algorithm for object tracking with learning
http://www.youtube.com/watch?v=1GhNXHCQGsM

Video Processing, with Processing
http://www.niklasroy.com/project/BB/my-little-piece-of-privacy/
http://www.youtube.com/watch?v=rKhbUjVykKc