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
  PImage warhol = loadImage("andy-warhol2.jpg");
  size(warhol.width*3, warhol.height);

  // Draw modified images
  tint(255, 0, 0);
  image(warhol, 0, 0);

  tint(0, 255, 0);
  image(warhol, 250, 0);

  tint(0, 0, 255);
  image(warhol, 500, 0);
}
```

Basic Filters

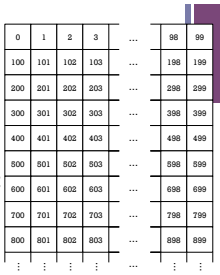
- Color
  - Extracting Red/Green/Blue colors
    - pixels[i] = color(red(c), 0, 0);
    - pixels[i] = color(0, 0, blue(c));
  - Grayscale
    - pixels[i] = color(0.3*red(c)+ 0.59*green(c)+ 0.11*blue(c));
  - Negative
    - pixels[i] = color(255-red(c), 255-green(c), 255-blue(c));
  - Sepia (Technique 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

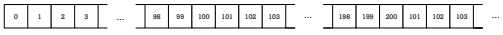
- blackWhite
- negative
- sepia
- sepiaPalette
- sepiaWithPalette

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

- whiteLine

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);

      int idx = width*y + x;

      // Set pixel as distance to center
      pixels[idx] = color(d);
    }
  }

  // Update the sketch with pixel data
  updatePixels();
}
```

PImage

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

Fields

```
width        - the width of the image
height       - the height of the image
pixels[]     - the image pixel colors
               (after a call to loadPixels())
```

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()
: Change the size of the image

Example

```
blackWhite2
```

PImage

Methods (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

get(…)

- Get a single pixel (very slow)

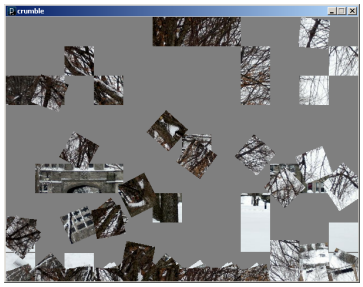
```
Color c = img.get(x, y);
```

- Get a rectangular range of pixels

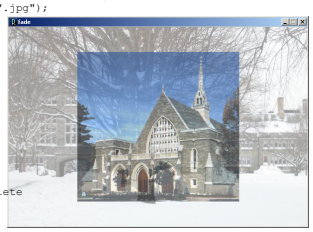
```
PImage img2 = img.get(x, y, w, h);
```

Example

- crumble
- reassemble



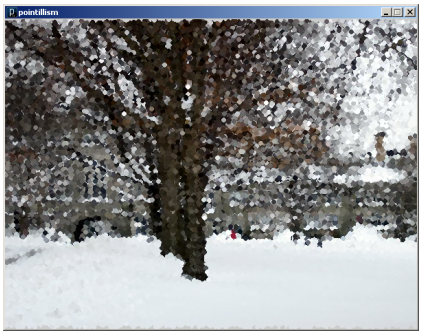

```
// fade
PImage[] img = new PImage[5];
int alpha = 255;
int i1 = 0, i2 = 1;

void setup() {
  size(600,400);
  imageMode(CENTER);
  for (int i=0; i<img.length; i++)  // Load images
    img[i] = loadImage("bmc"+i+".jpg");
}

void draw() {
  background(255);

  // Fade out current image
  tint(255, alpha);
  image(img[i1], 300, 200);

  // Fade in next image
  tint(255, 255-alpha);
  image(img[i2], 300, 200);

  // Swap images when fade complete
  alpha--;
  if (alpha < 0) {
    i1 = (i1 + 1) % img.length;
    i2 = (i2 + 1) % img.length;
    alpha = 255;
  }
}
```



Examples

- fade
- fade2

Pointillism



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

Medical Images

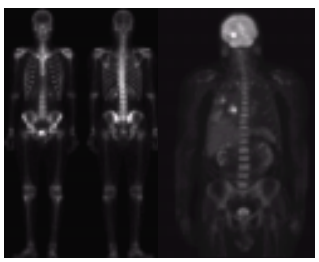


Image Processing in Manufacturing

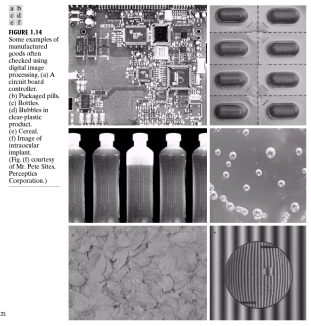


FIGURE 1.14 Some examples of manufactured goods often checked using digital image processing. (a) A circuit board controller. (b) Packaged pills. (c) Bottles. (d) Bubbles in clear-plastic product. (e) Cereal. (f) Image of intraocular implant. (Fig. (f) courtesy of Mr. Pete Sites, Perceptics Corporation.)

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=1WJp9mGnWSM

Thresholding for Image Segmentation

- Pixels below a cutoff value are set to black
- Pixels above a cutoff value are set to white

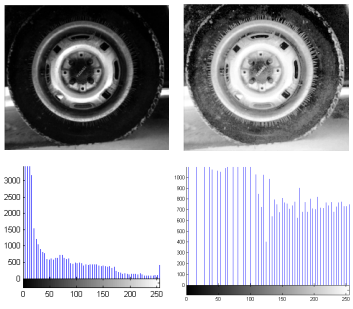
Obamicon

Example

- obamicon

Image Enhancement

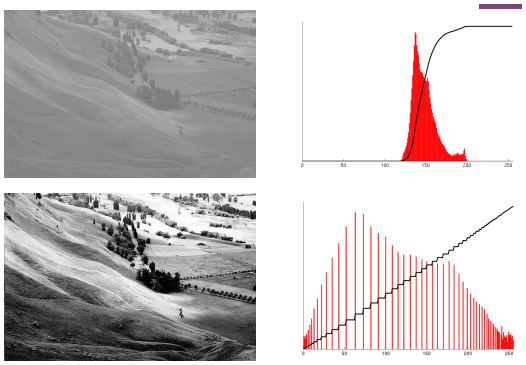
- Color and intensity adjustment
  - Histogram equalization



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

Histogram Equalization

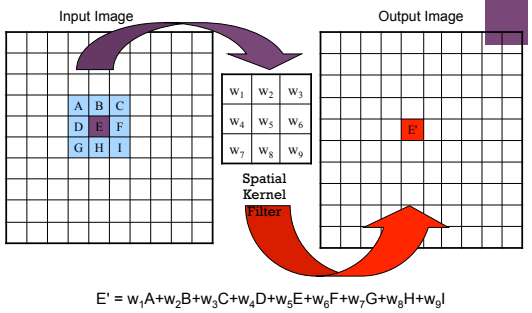
- Increases the global contrast of images
- So that intensities are better distributed
- Reveals more details in photos that are over or under exposed
- Better views of bone structure in X-rays



Histogram Equalization

- Calculate color frequencies - count the number of times each pixel color appear in the image
- Calculate the cumulative distribution function (cdf) for each pixel color – the number of times all smaller color values appear in the image
- Normalize over (0, 255)

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

0	0	0
0	1	0
0	0	0

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
- Smoothes out areas of sharp changes.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Sharpen – High Pass Filter

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

-1	-1	-1
-1	9	-1
-1	-1	-1

0	-2/3	0
-2/3	11/3	-2/3
0	-2/3	0

Blur – Low Pass Filter

- Softens significant color changes in image
- Creates intermediate colors

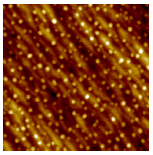
1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

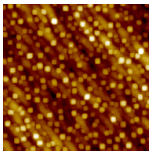
Example

- convolution

Dilation - Morphology

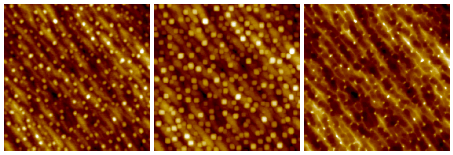
- Set pixel to the maximum color value within a neighborhood around the pixel
- Causes objects to grow in size.
- Brightens and fills in small holes





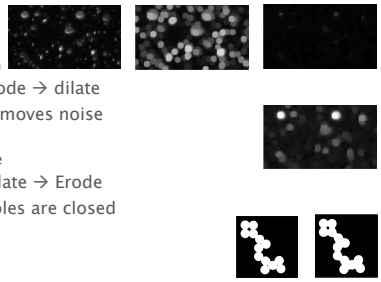
Erosion - Morphology

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



Feature Extraction – Region Detection

- Dilate and Erode
- Open
  - Erode → dilate
  - Removes noise
- Close
  - Dilate → Erode
  - Holes are closed



Erode + Dilate to Despeckle

Image Enhancement

- Denoise
  - Averaging
  - Median filter

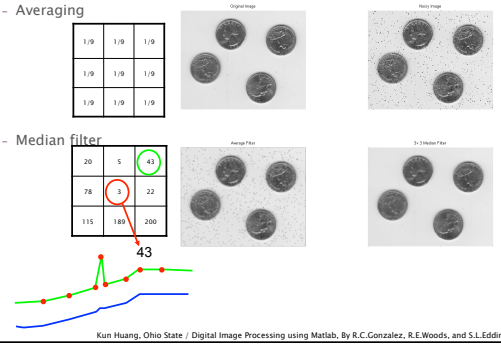


Image Processing in Processing

- `tint()`
  - modulate individual color components
- `blend()`
  - combine the pixels of two images in a given manner
- `filter()`
  - apply an image processing algorithm to an image

Blend Command

```
img = loadImage("colony.jpg");
mask = loadImage("mask.png");
image(img, 0, 0);
blend(mask, 0, 0, mask.width, mask.height,
            0, 0, img.width, img.height, SUBTRACT);
```

Draw an image and then blend with another image

BLEND	linear interpolation of colours:	C = A*factor + B
ADD	additive blending with white clip:	C = min(A*factor + B, 255)
SUBTRACT	subtractive blending with black clip:	C = max(B - A*factor, 0)
DARKEST	only the darkest colour succeeds:	C = min(A*factor, B)
LIGHTEST	only the lightest colour succeeds:	C = max(A*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

```
PImage b;
b = loadImage("myImage.jpg");
image(b, 0, 0);
filter(THRESHOLD, 0.5);
```

Draw an image and then apply a filter

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 colors 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 colors specified as the level parameter
BLUR	executes a Gaussian blur with the level parameter specifying the extent of the blurring. If no level parameter is used, the blur is equivalent to Gaussian blur of radius 1.
OPAQUE	sets the alpha channel to entirely opaque.
ERODE	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.

```
// Threshold
PImage img;

void setup() {
  img = loadImage("kodim01.png");
  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);
}
```

```
// Posterize
PImage img;

void setup() {
  img = loadImage("andy-warhol2.jpg");
  size(img.width, img.height);
  image(img, 0, 0);
}

void draw() {}

void drawImg(float val  {
  image(img, 0, 0);
  filter(POSTERIZE, val);
}

void mouseDragged() {
  float val = map(mouseY, 0, height, 2, 10);
  val = constrain(val, 2, 10);
  println(val);
  drawImg(val);
}
```

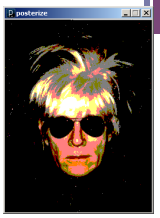
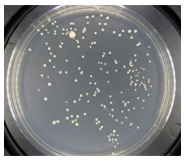


Image Processing Applications

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



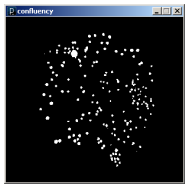
Blend: Subtract

Original | Mask | Subtracted

+ Filter: Theshold

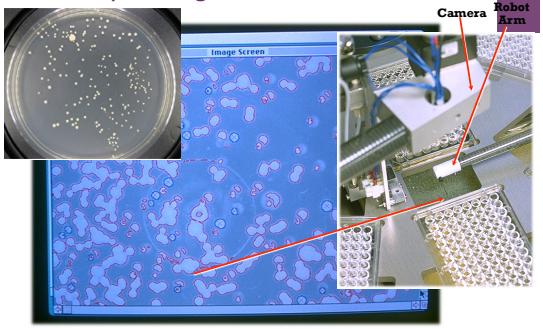
Subtracted

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/88/my-little-piece-of-privacy/

http://www.youtube.com/watch?v=rKhbUjVyKIc