Review
- Images – an array of colors
- Color – RGBA
- Loading, modifying, updating pixels
- pixels[] as a 2D array
- Simple filters – tinting, grayscale, negative, sepia
- PImage class, fields and methods
- get() method and crumble
- tint() function – color and alpha filtering
- Creative image processing – Pointillism, other shapes

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

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)

Input Image   Output Image

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Spatial Kernel
Filter

$$E' = w_1A + w_2B + w_3C + w_4D + w_5E + w_6F + w_7G + w_8H + w_9I$$

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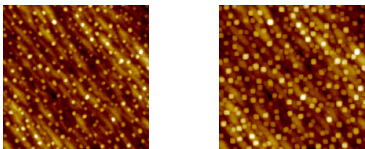
1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

Example

- convolution
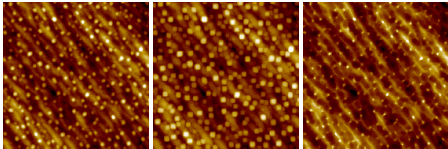
Dilation - Morphology

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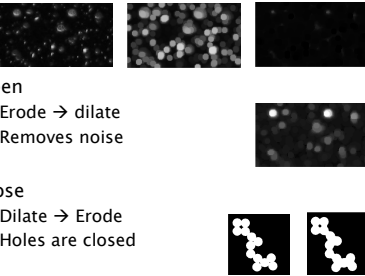- Brightens and fills in small holes

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

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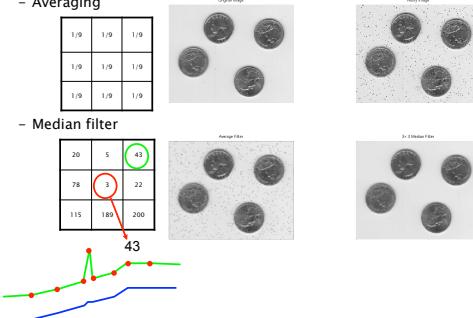
Erode + Dilate to Despeckle

Erode   Dilate

Image Enhancement

– Denoise
  – Averaging

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

Original Image   Noisy Image

– Median filter

20	5	43
78	3	22
115	189	200

43

Average Filter   3× 3 Median Filter

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

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.	
SCREEN	Opposite multiply, uses inverse values of the colors.	
OVERLAY	A mix of MULTIPLY and SCREEN. Multiplies dark values, and screens light values.	
HARD_LIGHT	SCREEN when greater than 50% gray, MULTIPLY when lower.	
SOFT_LIGHT	Mix of DARKEST and LIGHTEST. Works like OVERLAY, but not as harsh.	
DODGE	Lightens light tones and increases contrast, ignores darks.	
BURN	Darker areas are applied, increasing contrast, ignores lights.	

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

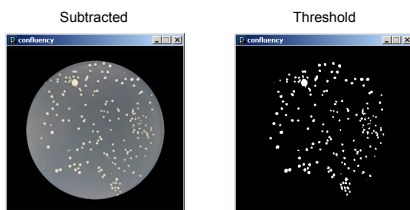Measuring Confluency in Cell Culture Biology

- 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
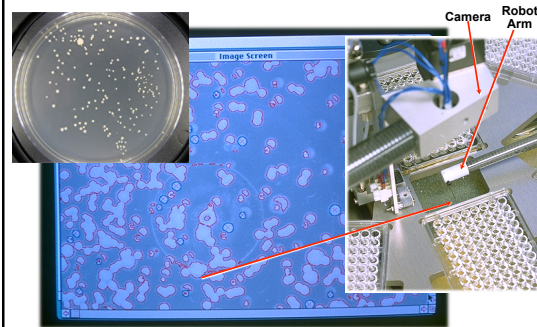 3. Count pixels of certain color

---

Blend: Subtract

Original   Mask   Subtracted
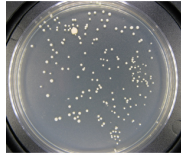
Filter: Theshold
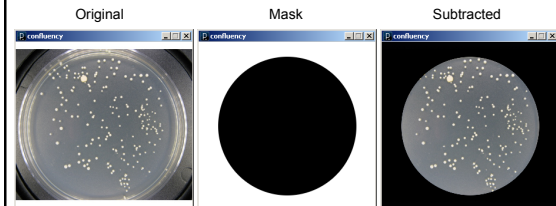
Subtracted   Threshold

Count pixels to quantitate: 5.3% confluency

**Vision Guided Robotics
Colony Picking**

Camera   Robot Arm

Image Screen

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