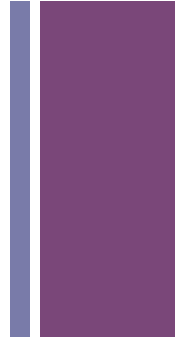




Image Processing

+ Quiz 5



- Write a recursive function `int sum(int x, int y)` that computes and returns the sum of all integers between `x` and `y`, inclusive. Remember, the function **MUST** be recursive.

+ Quiz 4



- Declare a float array, dimension it to hold 100 values, and fill it with random numbers. Print the numbers after they have been generated and stored in the array.
- Expand the previous program. After the array is fully initialized with random numbers, revisit each element in the array and multiply each by 2. Store each doubled value in the original array location.
- Modify the previous program further. Move the part of your program that doubles the values in the array to a separate function. The new function should be called `doubleIt()`. It should take a float array as its only argument and return the modified float array. The body of the function should double each element of the array. Change your program to use this new function to double the values in the original float array.

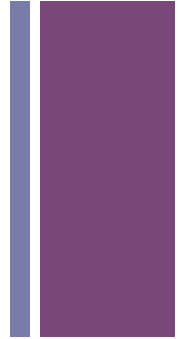
+ Quiz 4 cont.

```
int x;
int y;
color c;

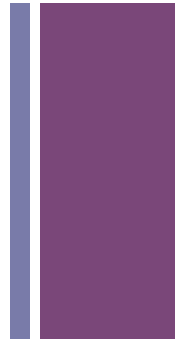
void setup() {
  size(500, 500);
  x = int(random(width));
  y = int(random(height));
  c = color(random(255), random(255), random(255));
}

void draw() {
  background(255);
  x = (x + 1) % width;
  y = (y + 1) % height;
  fill( c );
  rect(x, y, 20, 20);
}
```

The following program moves a randomly located and randomly colored square diagonally down to the right, wrapping at the boundaries. Modify the program by designing a class called Square that models a square that moves from diagonally down to the right, wrapping at the boundaries. Your program will create 50 randomly placed and randomly colored squares, storing them using arrays. Your program will then draw and animate all of them appropriately.

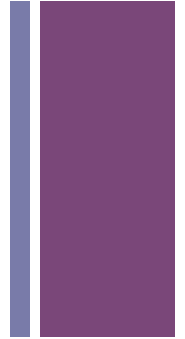


+ Quiz 3



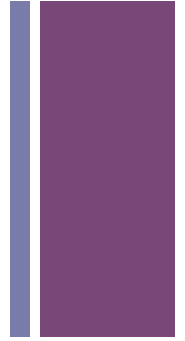
- Write two loops that paint a grid over the sketch window using horizontal and vertical lines. The resolution of the grid (i.e. number of cells in each direction) should be determined by two int variables, m and n, where m specifies the number of rows and n specifies the number of columns.
- Modify your above answer to 2) to a void function named grid that does the same, and takes m and n as parameters. In addition, add a call to your function that will draw a grid of 50x100 covering the entire sketch.
- Write a function that takes an integer x and returns true if x is prime and false otherwise.
(Reminder: A prime number is a number greater than one that is only divisible by itself and one.)

+ Quiz 2



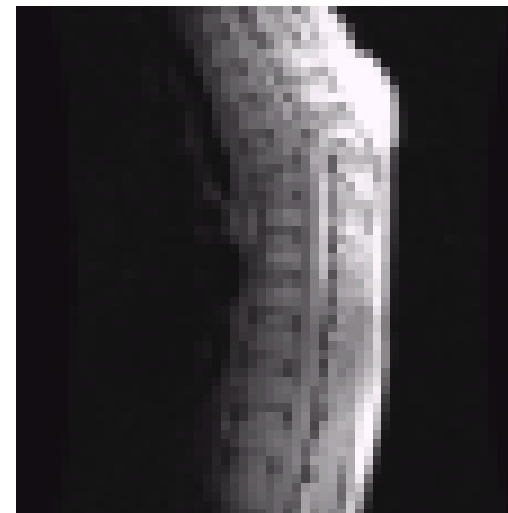
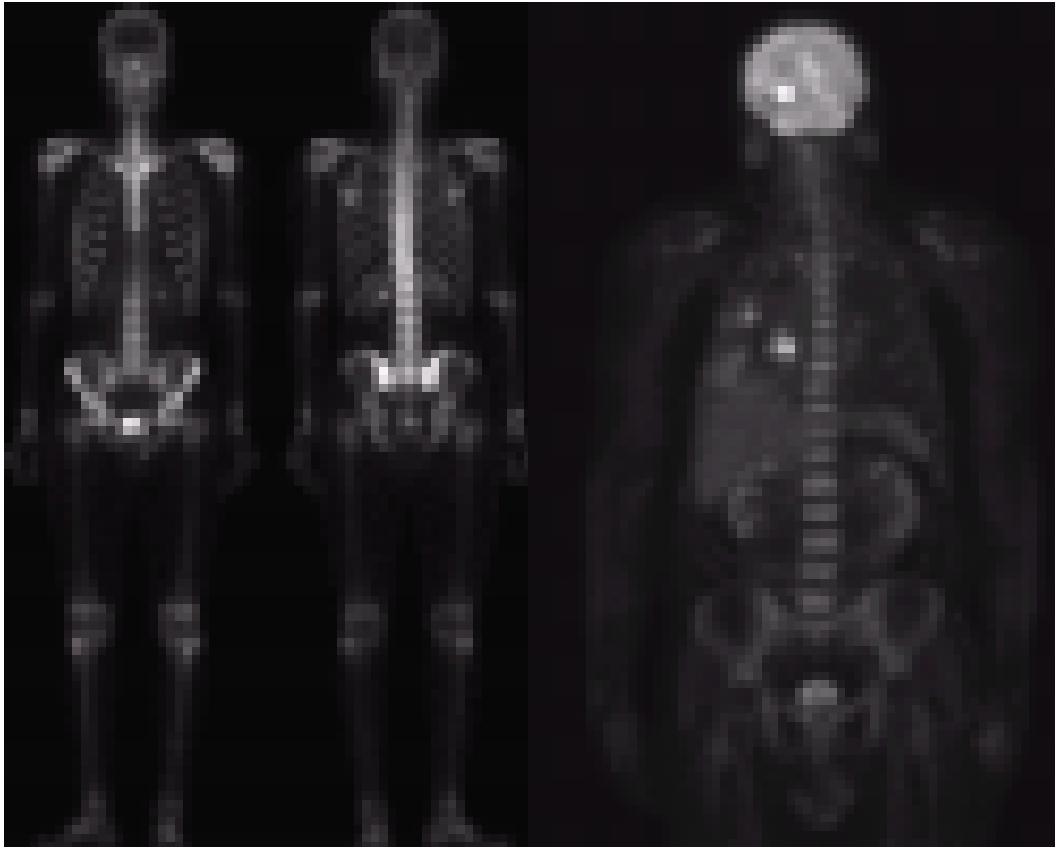
- Write a loop that prints all integers between 3 and 52 inclusive, that are divisible by 3.
(Partial credit: a loop that prints all integers between 3 and 52 inclusive)

+ 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

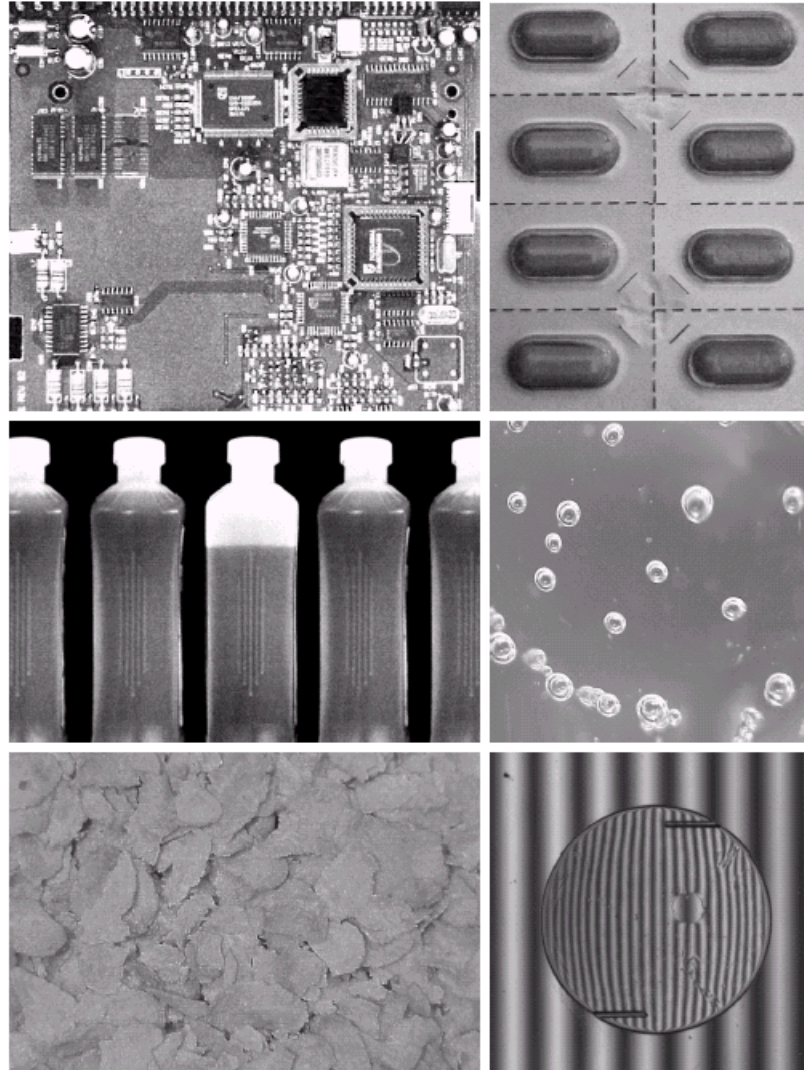
+ Medical Images



+ Image Processing in Manufacturing

a b
c d
e f

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.)





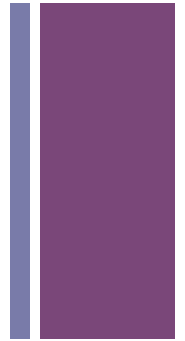
What can you do with Image Processing?

Inspect, Measure, and Count using Photos and Video

<http://www.youtube.com/watch?v=KsTtNWVhpgI>

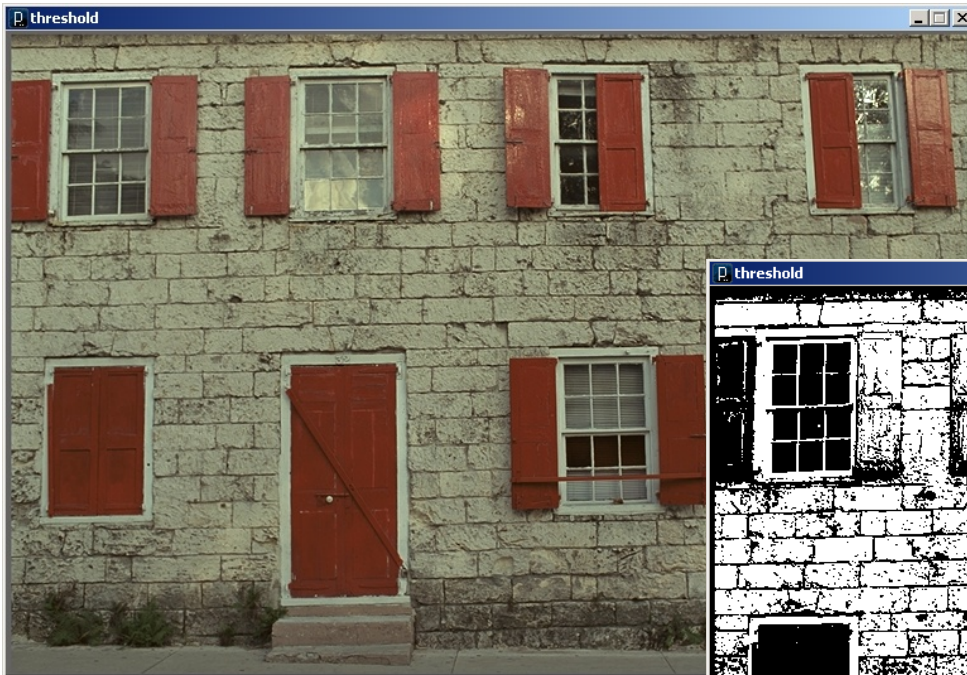
Image Processing Software

<http://www.youtube.com/watch?v=lWJp9mGnWSM>

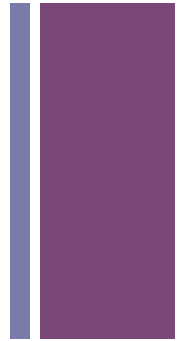


+ Thresholding for Image Segmentation

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



+ Obamicon



+ Example

- obamicon

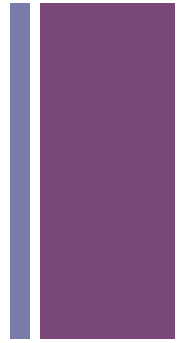
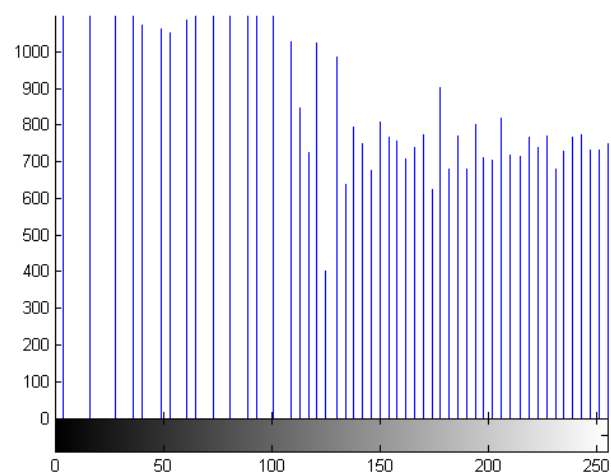
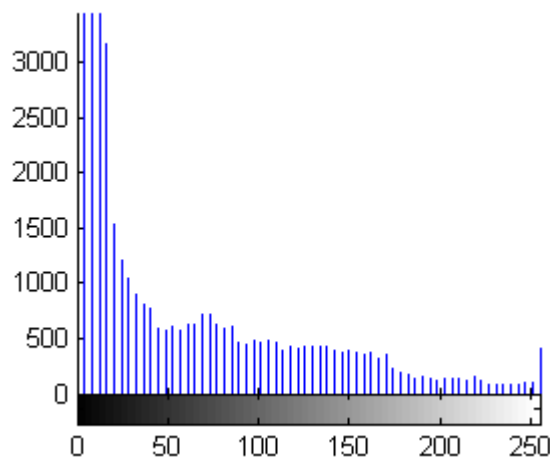


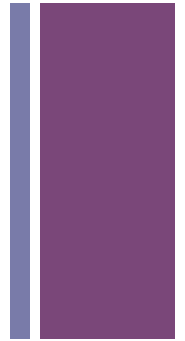
Image Enhancement

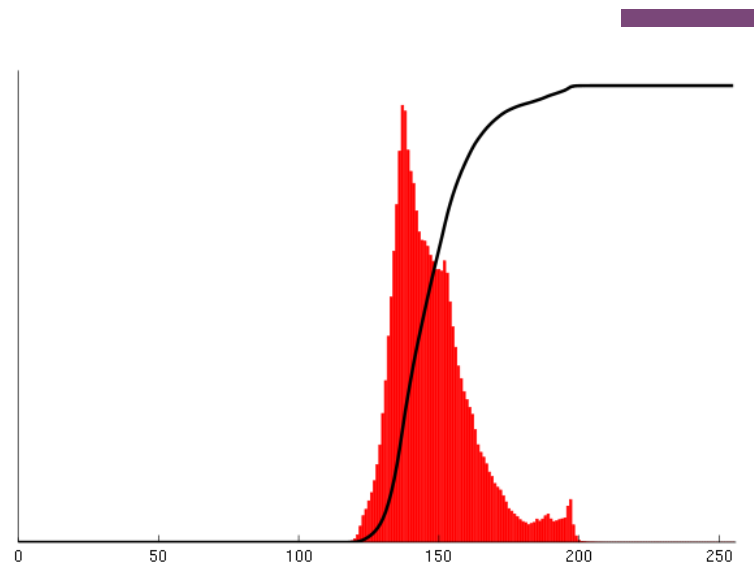
- Color and intensity adjustment
- Histogram equalization



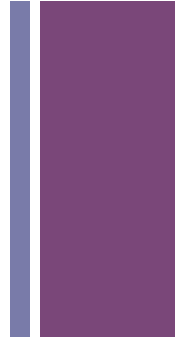
+ Histogram Equalization

- 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



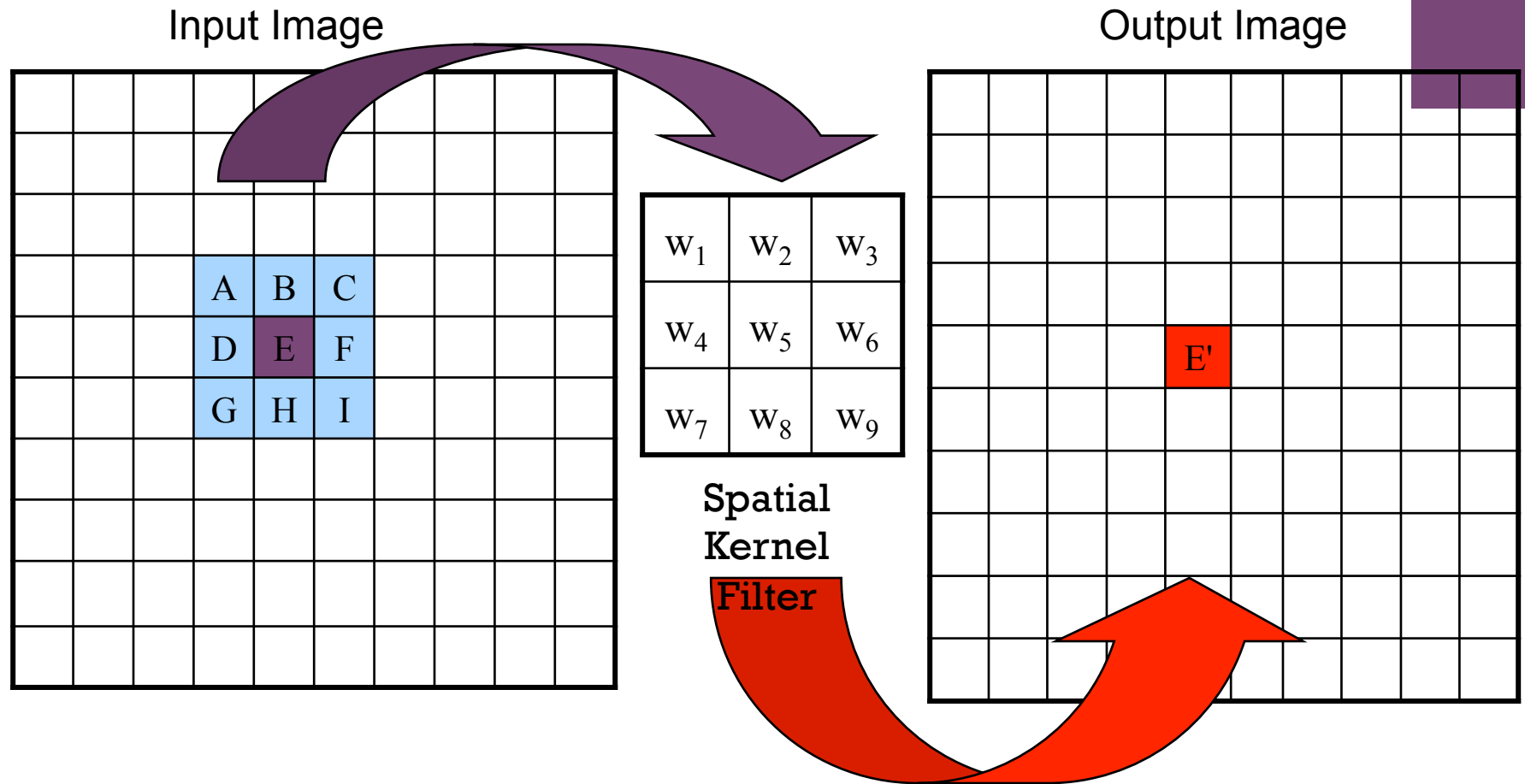


+ Histogram Equalization



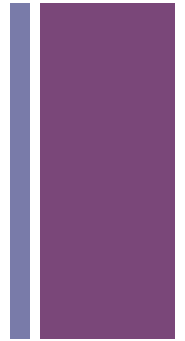
- Calculate color frequencies - count the number of times each pixel color appear in the image
 - save each value from 0-255 as an array of percentages, p
- Calculate the cumulative distribution function (cdf) for each pixel color – the number of times all smaller color values appear in the image
 - sum $p[0]$ to $p[\text{pixel color}]$ as cdf
- Normalize over (0, 255)
 - multiply cdf by 255
- store new pixel values and display the modified image.

Convolution Filters (Area-based)



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

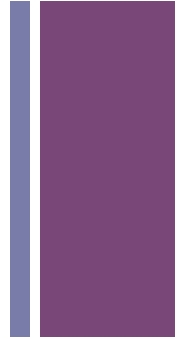
+ Identity



- No change

0	0	0
0	1	0
0	0	0

+ Random Neighbor

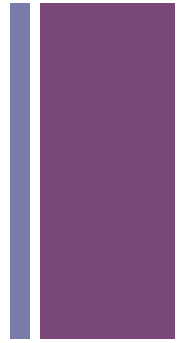


- Copies randomly from one of the 8 neighbors, and itself

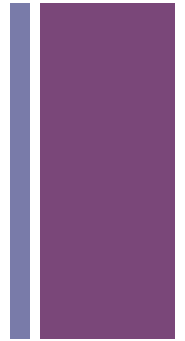


+ Example

- randomNeighbor



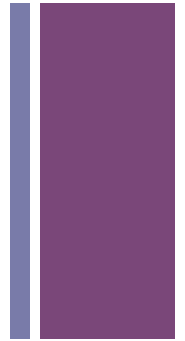
+ Average – smooth



- Set pixel to the average of all colors in the neighborhood
- Smooths 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

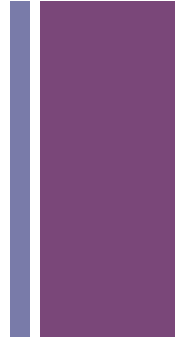


- Enhances the difference between neighboring pixels
- The greater the difference, the more change in the current pixel

-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

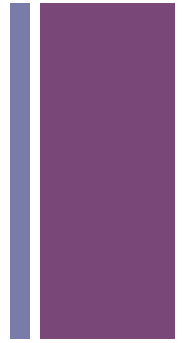


- Softens significant color changes in image
- Creates intermediate colors

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

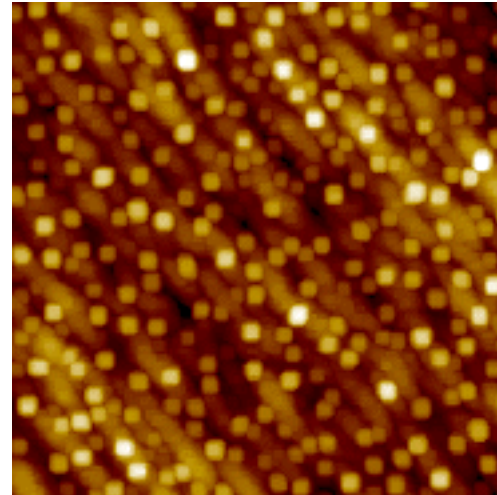
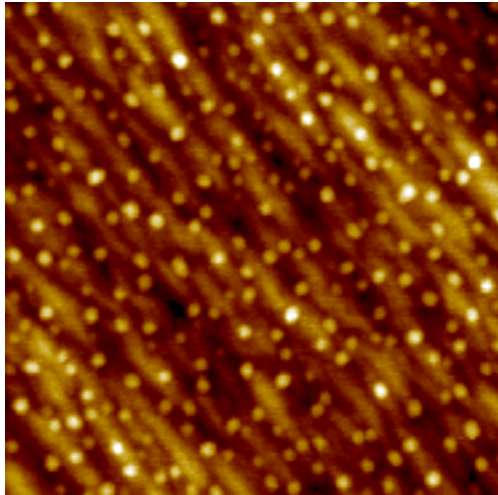
+ Example

- convolution



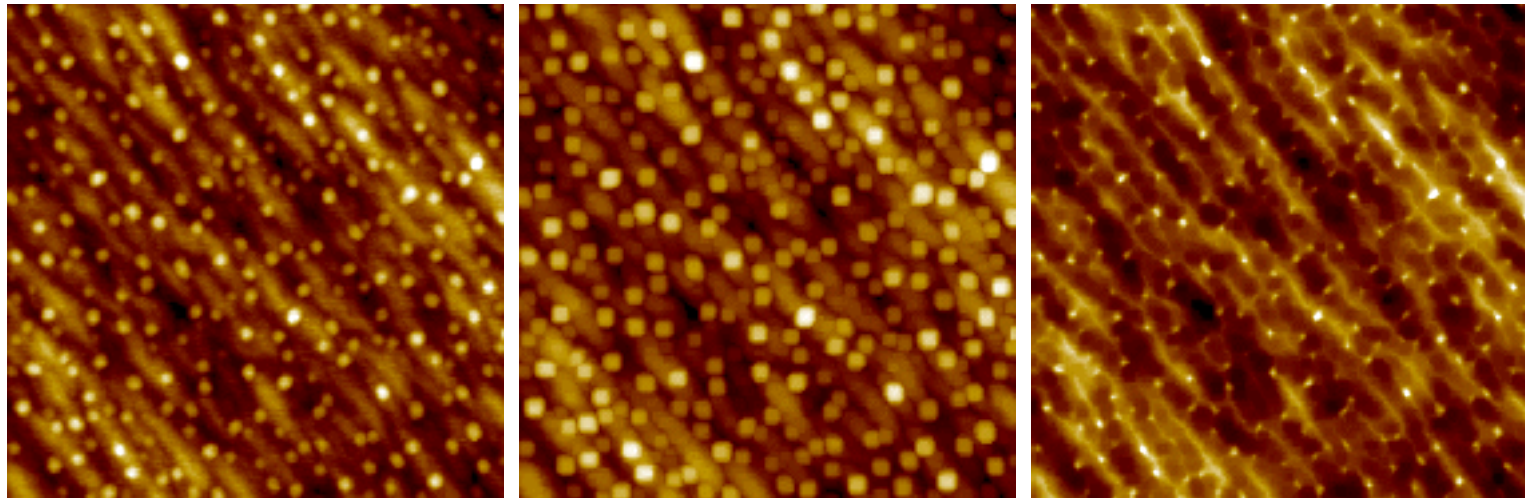
+ Dilation - Morphology

- 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

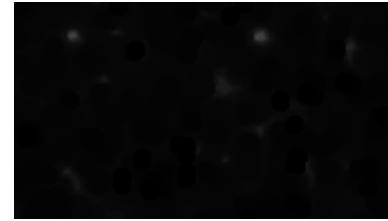
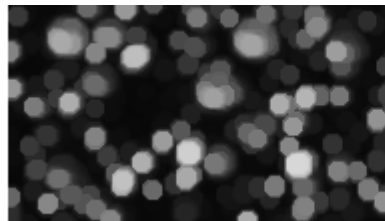
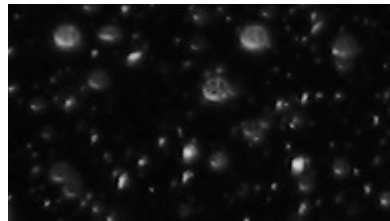




Feature Extraction – Region Detection

- Dilate and Erode

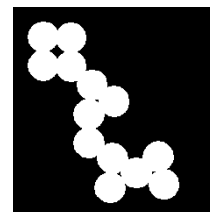
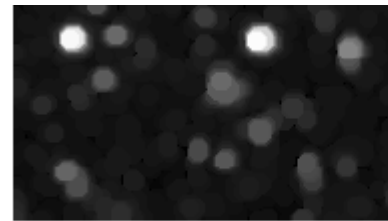
- Open



- Erode → dilate
- Removes noise

- Close

- Dilate → Erode
- Holes are closed



+ Erode + Dilate to Despeckle



Erode



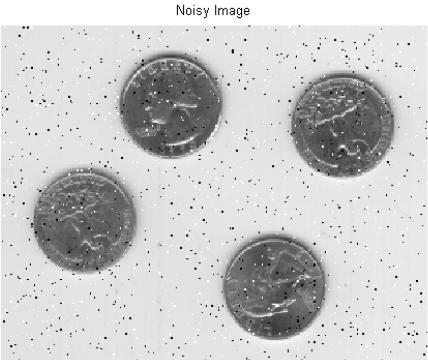
Dilate



Image Enhancement

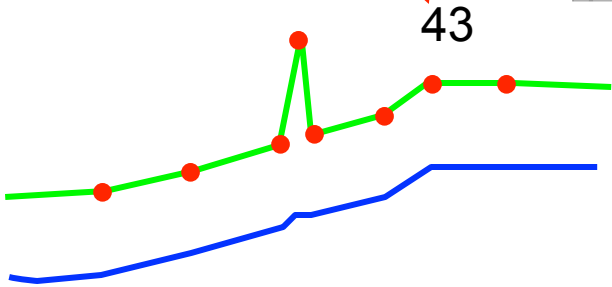
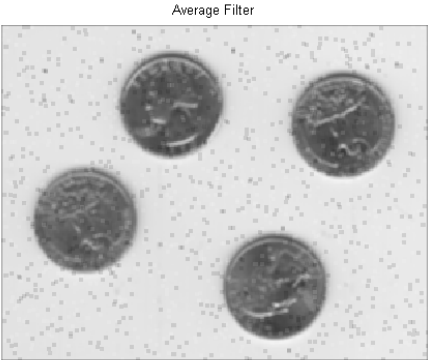
- Denoise
- Averaging

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

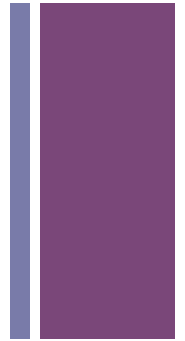


- Median filter

20	5	43
78	3	22
115	189	200



+ 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:	$C = A * \text{factor} + B$
ADD	additive blending with white clip:	$C = \min(A * \text{factor} + B, 255)$
SUBTRACT	subtractive blending with black clip:	$C = \max(B - A * \text{factor}, 0)$
DARKEST	only the darkest colour succeeds:	$C = \min(A * \text{factor}, B)$
LIGHTEST	only the lightest colour succeeds:	$C = \max(A * \text{factor}, B)$
DIFFERENCE		subtract colors from underlying image.
EXCLUSION	similar to DIFFERENCE, but less extreme.	
MULTIPLY	Multiply the colors, result will always be darker.	

+ Filter Command

```
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.

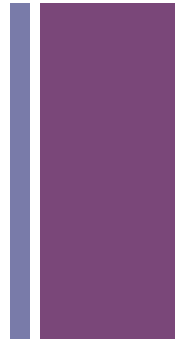
```
+ // Threshold
  PImage img;

  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);
```

```
}
```

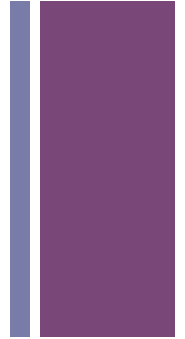




Image Processing Applications

Manual Colony Counter

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

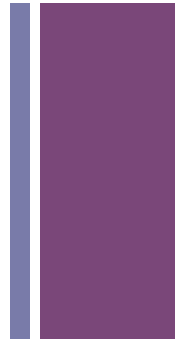
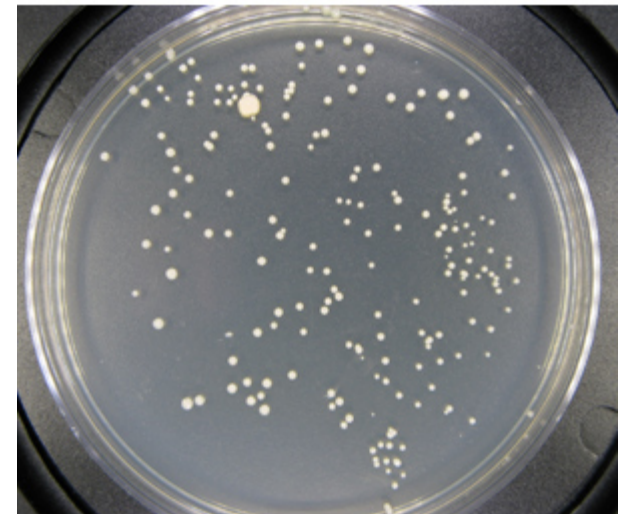




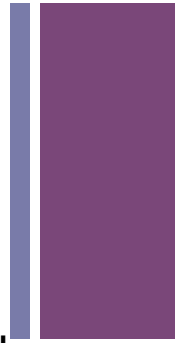
Measuring Confluency in Cell Culture Biology

- 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



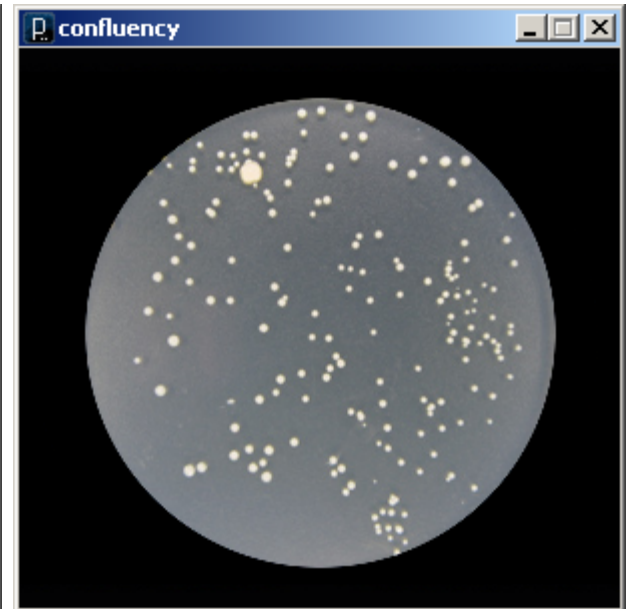
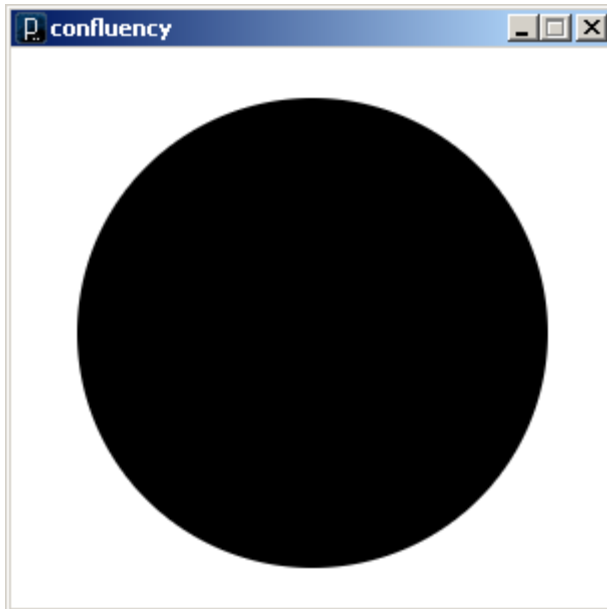
+ Blend: Subtract



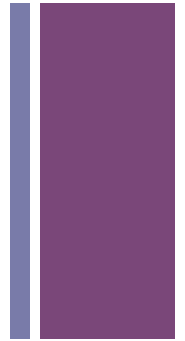
Original

Mask

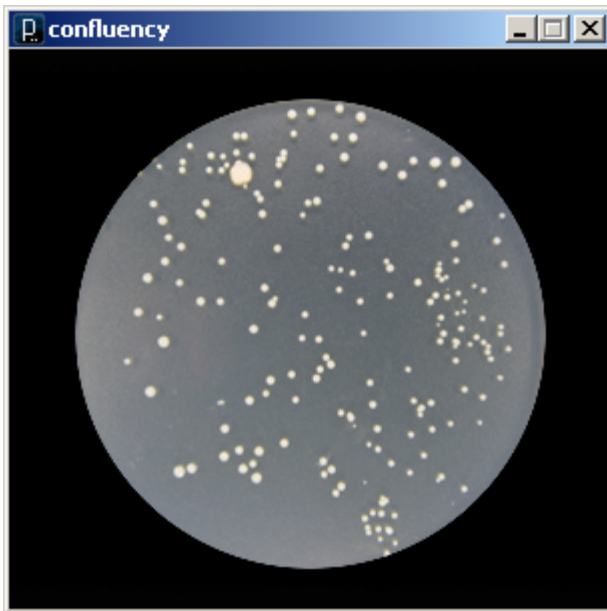
Subtracted



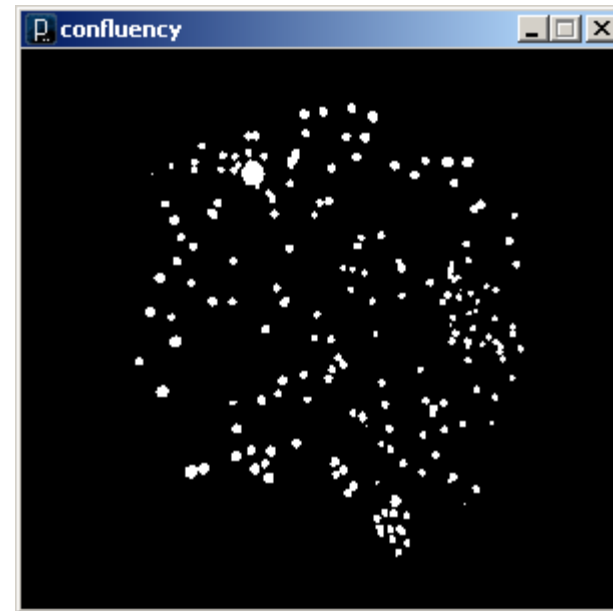
+ Filter: Theshold



Subtracted



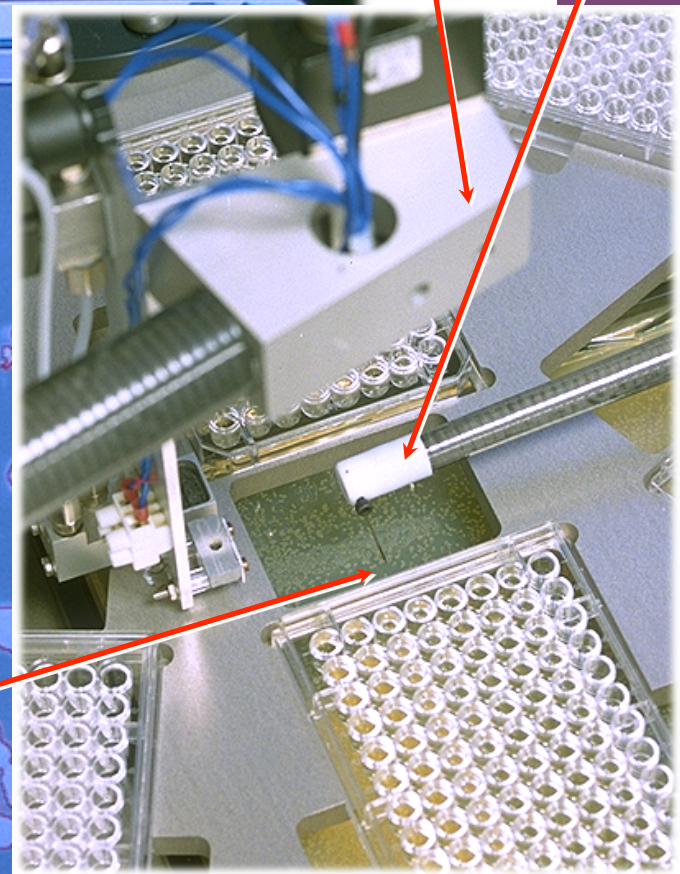
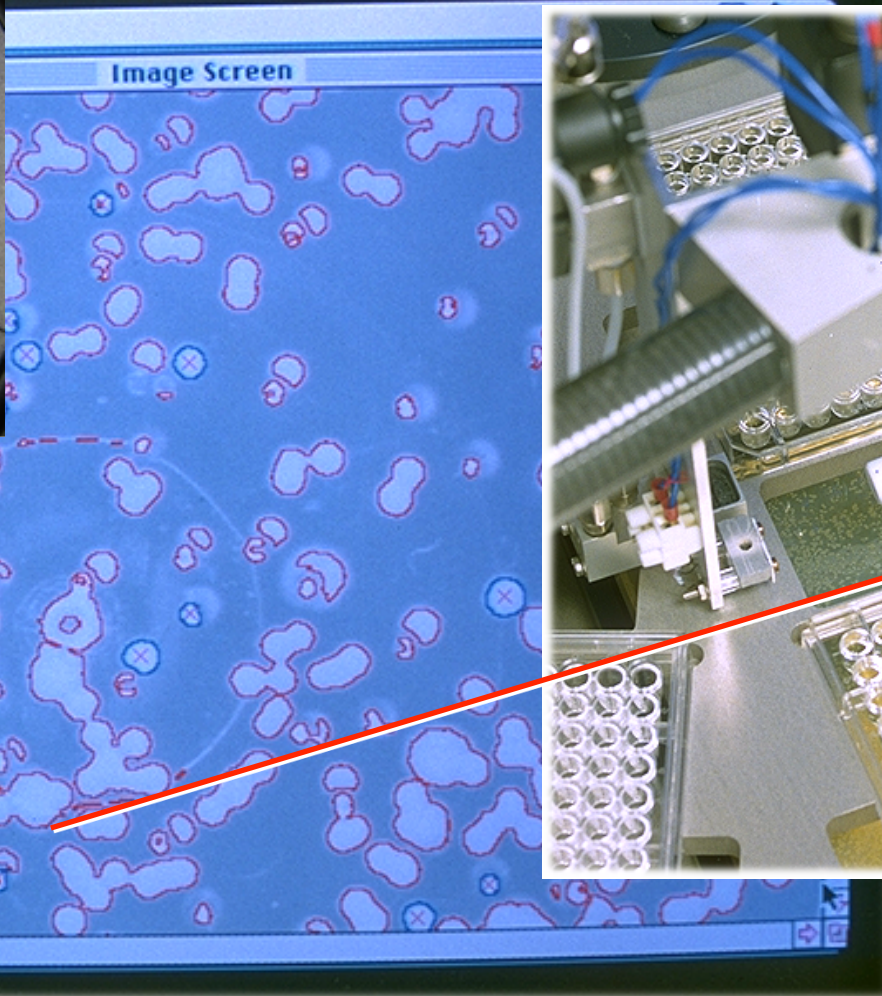
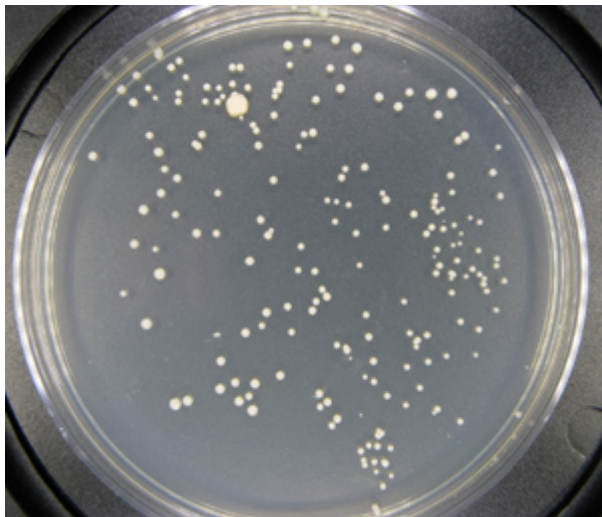
Threshold



Count pixels to quantitate:

5.3% confluency

+ Vision Guided Robotics Colony Picking



Camera
Robot Arm

- + 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=rKhbUjVyKlc>

