Review

- thresholding
- Convolution filters (area-based filters)
- blend()
- filter()

Filter class

```
class Filter{
  float[][] kernel;      // convolution kernel
  float normalizer;      // normalizing factor
  float offset;          // brightening offset
  String name;           // display label

  Filter(float[][] kernel, float normalizer, float offset,
String name) {
    this.kernel = kernel;
    this.normalizer = normalizer;
    this.offset = offset;
    this.name = name;
  } // end Filter()

  //...
```

apply()

```
 color apply(int x, int y, PImage img) {
   int halfSize = kernel.length/2;
   float r = 0.0, g = 0.0, b = 0.0;
   for (int i = 0; i < kernel.length; i++) {
     for (int j= 0; j < kernel[i].length; j++) {
       int newX = x+j-halfSize;
       int newY = y+i-halfSize;
       int idx = img.width*newY + newX;
       idx = constrain(idx, 0,img.pixels.length-1);
       r+=(red(img.pixels[idx])*kernel[i][j]/normalizer);
       g+=(green(img.pixels[idx])*kernel[i][j]/normalizer);
       b+=(blue(img.pixels[idx])*kernel[i][j]/normalizer);
     }
   }

   return color(r+offset, g+offset, b+offset);
  } // end apply()
} // end class Filter
```

Kernels

```
float[][][] ks = {{{1, 1, 1}, {1, 1, 1}, {1, 1, 1}},
                  {{1, 2, 1}, {2, 4, 2}, {1, 2, 1}},
                  {{1, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 1, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 1, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 1, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 1, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 1, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 1, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 1, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 1}},
                  {{0, -2, 0}, {-2, 11, -2}, {0, -2, 0}},
                  {{-1, -1, -1}, {-1, 9, -1}, {-1, -1, -1}},
                  {{1, 1, 1}, {1, -7, 1}, {1, 1, 1}},
                  {{0, -1, 0}, {-1, 4, -1}, {0, -1, 0}},
                  {{-1, -1, -1}, {-1, 8, -1}, {-1, -1, -1}},
                  {{1,1,0},{1,0,-1},{0,-1,-1}}};
```

Create filters

```
Filter[] filters = {new Filter(ks[0], 9, 0, "Mean"),
                    new Filter(ks[1], 16, 0,
                    "Gaussian Blur"),
                    new Filter(ks[2], 9, 0,
                    "Motion Blur"),
                    new Filter(ks[3], 3, 0, "Sharpen"),
                    new Filter(ks[4], 1, 0,
                    "Mean Removal"),
                    new Filter(ks[5], 1, 0, "Mystery"),
                    new Filter(ks[6], 1, 0,
                    "Edge Detection Horizontal/Vertical"),
                    new Filter(ks[7], 1, 0,
                    "Edge Detection with Diagonal"),
                    new Filter(ks[8], 1, 127, "Emboss")};
```

Convolution

```
PImage img, img2;

void setup() {
  img = loadImage("prinzipal.jpg");
  img2 = createImage(img.width, img.height, RGB);
  size(img.width*2, img.height);

  img.loadPixels();
  img2.loadPixels();
  applyFilter(0);    // apply first filter - Mean
} // end setup()

void draw(){}

void keyPressed() {
  if (key >= '0' && key <= '8') {
    applyFilter(key-'0');
  }
} // end keyPressed()
```

Apply filters

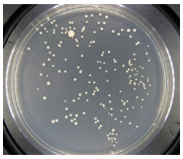
```
void applyFilter(int n) {
  for (int y=0; y<img.height; y++ ) {
    for (int x=0; x<img.width; x++) {
      img2.pixels[y*img.width+x] =
      filters[n].apply(x, y, img);
    }
  }

  img2.updatePixels();
  image(img, 0, 0);
  image(img2, width/2, 0);
  textSize(20);
  text(filters[n].name, width/2+width/25, height/25);
} // end applyFilter()
```

Measuring Confluency in Cell Culture Biology

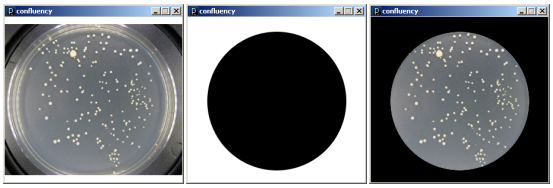
- Refers to the coverage of a dish or flask by the cells
- 100% confluency = completely covered

- Image Processing Method
  1. Mask off unimportant parts of image
  2. Threshold image
  3. Count pixels of certain color



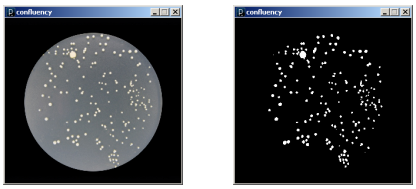
Blend: Subtract

Original Mask Subtracted



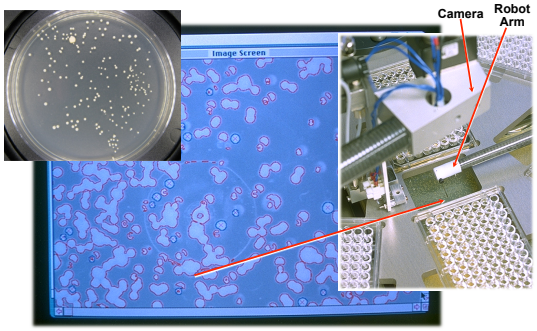
Filter: Theshold

Subtracted Threshold



Count pixels to quantitate: 5.3% confluency

Vision Guided Robotics Colony Picking



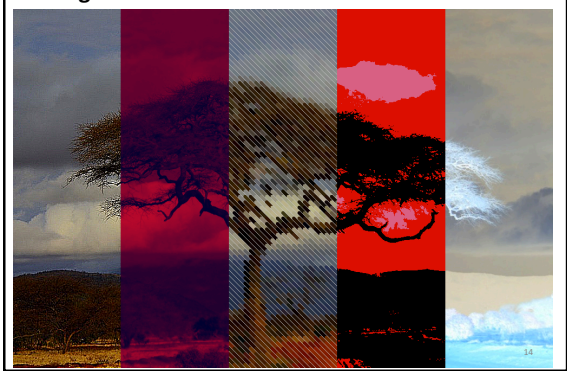
Predator algorithm for object tracking with learning
http://www.youtube.com/watch?v=1GhNXHCQGsM

Video Processing, with Processing
http://www.niklasroy.com/project/88/my-little-piece-of-privacy/
http://www.youtube.com/watch?v=rKhbUjVyKIc

Assignment 06

13

Assignment 06

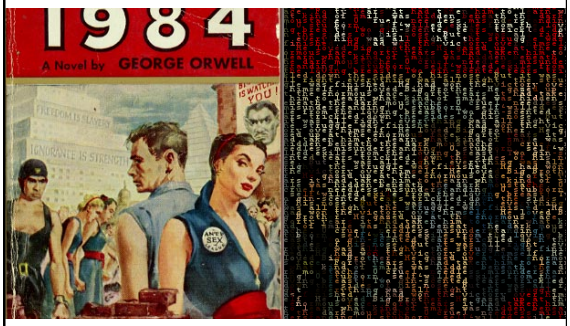


14

Assignment 06

15

Assignment 06



16

Video Processing

- Video is a sequence of still images.
- All image processing techniques apply per frame.
- Interesting effects can happen between frames.

Simple Video

- Import video library
- Create a Capture object which represents a camera
- Start camera
- Read a frame if it is available
- Draw the frame

```
import
processing.video.*;
Capture cam;

void setup() {
  size(640, 480);
  cam = new Capture(this,
        640, 480);
  cam.start();
}

void draw() {
  if (cam.available()) {
    cam.read();
  }
  image(cam, 0, 0);
}
```

Examples

- simpleVideo
- diffFrame
- textVideo